Learning a Family of Optimal State Feedback Controllers

Christopher Iliffe Sprague[a]
sprague@kth.se

Dario Izzo[b]
dario.izzo@esa.int

Petter Ögren[a]
petter@kth.se

***Abstract*— Solving optimal control problems is well known to be very computationally demanding. In this paper we show how a combination of Pontryagin's minimum principle and machine learning can be used to learn optimal feedback controllers for a parametric cost function. This enables an unmanned system with limited computational resources to run optimal feedback controllers, and furthermore change the objective being optimised on the fly in response to external events. Thus, a time optimal control policy can be changed to a fuel optimal one, in the event of e.g., fuel leakage. The proposed approach is illustrated on both a standard inverted pendulum swing-up problem and a more complex interplanetary spacecraft orbital transfer.**

***Index Terms*— Neural Networks, Machine Learning, Optimal Control, Online Planning**

I. Introduction

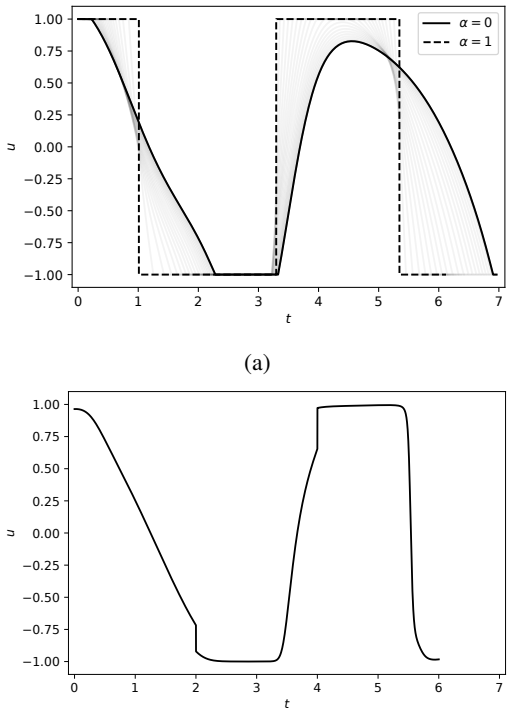
Applying an optimal control policy is often desirable, but sometimes not feasible for many unmanned systems. The reason for this is that optimal control algorithms are known to be very computationally intensive [1], and many unmanned systems have both limited computational resources onboard and limited access to offline resources. This is true for systems operating in remote areas, such as space or maritime environments, as well as other systems working under varying conditions in terms of network connectivity, such as search and rescue robots.

One way to address this problem is to pre-compute optimal trajectories, and then apply a feedback controller to track those trajectories. However, this is not viable when unforeseen events such as disturbances might bring the vehicle away from the given trajectory. Another way to make optimal control feasible is to apply a Model Predictive Control (MPC) approach where the optimisation horizon is shortened to make the computations feasible, and then performed in an iterative fashion to provide reactivity in response to unforeseen events. This approach often performs well, but not as good as the optimal solution.

In this paper we address the problem of optimal control by solving a very large set of optimal control problems offline, and then use supervised learning to train an artificial neural network (ANN) to reproduce these control policies. The result is a feedback controller that runs on low performance hardware, and produces trajectories that are optimal with respect to exactly the criteria that corresponds to the given parameter, see Figure 1.

[a] Robotics, Perception and Learning Lab., School of Electrical Engineering and Computer Science, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden

[b] Advanced Concepts Team, European Space Technology Center (ESTEC), Noordwijk, The Netherlands



(a)

(b)

Fig. 1: (a) A family of optimal control solutions of the inverted pendulum problem, ranging from a quadratic control cost ($\alpha = 0$) to time optimal cost ($\alpha = 1$). (b) Running the trained ANN with $\alpha = 0$ initially, switching to $\alpha = 0.5$ at $t = 2$ and then to $\alpha = 1.0$ at $t = 4$. Note how the trajectory starts out similar to the solid line in (a) and finishes similar to the dashed, but with a slight smoothing of the discontinuities. Note also that it is a state feedback control, and not just switching between pre-computed open loop control sequences. The corresponding state trajectories can be seen in Figure 2.

This enables the on-board software to change the optimality criterion in the middle of a mission. Imagine a vehicle that started off on a time optimal trajectory towards a given destination but suddenly finds itself having less energy than expected. Perhaps due to a battery failure, a broken solar panel or a leaking fuel tank. It can then instantly switch to a fuel optimal controller. Conversely, a vehicle that was saving fuel for future transitions might suddenly be in a hurry to reach a destination and can then instantly switch to a controller that reaches the goal in the shortest possible time.

The main contribution of this paper is that we learn a family of optimal controllers for a given system and task, in such a way that the resulting feedback controller is executable on a system with limited computational resources, and the optimality criterion can be instantly changed.

We illustrate the proposed approach on two different problems. First we choose the classical inverted pendulum swing up problem, where a cart has to move along a line to swing up and then balance a pendulum hanging from the cart. Then we address a more complex, higher dimensional, interplanetary low-thrust orbit transfer problem, where a spacecraft chooses the orientation of its ion propulsion thruster to transfer from one orbit around the Sun to another one. In both cases we show that our ANNs achieve near-optimal performance with respect to the chosen optimality criterion.

The outline of the paper is as follows. In Section II we describe related work. Then, in Section III, we explain the homotopy continuation method we use for optimal control. In Section IV, we explain our method of dataset generation, our machine learning setup, and the results we achieved. Finally, in Section V, we reflect upon our results and discuss what they entail.

II. Related Work

Nonlinear optimal control problems are important for many robotic systems [2]–[4], but they are notoriously difficult to solve [1]. While it is intractable for most nonlinear systems to find an exact solution to the optimal control problem (closed-loop), numerical approximations to the alternative infinite-time horizon trajectory optimisation problem (open-loop) can be obtained through variational methods [5], such as Pontryagin's minimum principle [6]. The control sequence resulting from this open loop solution can be applied directly; however, inaccuracies in the model can result in deviations. Although such a solution could serve as a motion guideline for simpler embedded controllers, the guiding trajectory may become irrelevant if the environment changes or if the system ventures away from it.

An alternative approach is to rely on MPC methods, where the finite-time horizon optimal control problem is solved to some extent on a receding time interval. With advances in MPC's efficiency [7] and increasing computation capabilities, MPC approaches have become popular in a variety of applications [4], [8]–[10]. While MPC can be quite effective in implementation, the need to employ a receding horizon for computational tractability results in sub-optimal trajectories. Furthermore, discrepancies between the dynamics model and reality may have adverse effects in the trajectory [11]. A key difficulty of the MPC method is that it requires a model.

For many complex systems, generating an accurate analytical dynamics model is impossible. As a result, some works [12]–[15] have used machine learning methods to learn a dynamics model to use in conjunction with MPC. While these methods can give good results, the need to solve the finite-horizon trajectory optimisation problem still stands. Considering that the problem becomes more difficult as the time horizon increases, there is an evident trade off between computational tractability and trajectory optimality [11], [16]. The major distinction to make between MPC approaches and the imitation learning approach of this work is that our method does not require knowledge of the underlying system, we only need expert (optimal) demonstrations, i.e. state-control pairs. Hence, we avoid the need to learn the dynamics and employ the computationally burdensome MPC.

With recent advances in machine learning, the problem of determining control policies has been reconsidered using artificial neural networks [17]. A number of approaches considered learning reactive control policies directly through imitation learning [18]–[20], circumventing MPC's need for iterative optimisations. Contrary to what was commonly believed [21], it was shown in [22] that such networks can, in deterministic systems, effectively learn control policies by imitating optimal demonstrations originating from the solution to Pontryagin's minimum principle[1]. In [23] the study of these networks was extended to higher state space dimensions and more complicated boundary conditions, shown in the case of an interplanetary spacecraft trajectory.

In [22]–[24], the method of homotopic continuation [25] was used to solve trajectory optimisation problems with particularly adverse cost functions. In these works, a trajectory cost function, parameterised by a homotopy parameter $\alpha \in [0,1]$, was used to describe the trade off between two objectives. With homotopic continuation, the trajectory optimisation problem was solved with increasing α in order to obtain solutions at both $\alpha = 0$ and $\alpha = 1$. This method was employed to independently learn the optimal state feedback controllers corresponding to two seperate objectives.

We go beyond these works by letting the ANN learn the complete homotopy path between different optimal control policies, with the homotopy parameter deciding the objective function as input to the network. Thus we enable vehicles with limited computational resources to reactively change between e.g., fuel and time optimal control on the fly.

III. Optimal control

In this section we generate a very large database of optimal trajectories that will be used for learning in Section IV.

Throughout the paper we consider nonlinear autonomous dynamical systems of the form $\dot{\boldsymbol{s}} = \boldsymbol{f}(\boldsymbol{s},\boldsymbol{u})$, with a scalar trajectory cost function of the form $\mathscr{J} = \int_0^{t_f} \mathscr{L}(\boldsymbol{u}(t))dt$. Through Pontryagin's minimum principle [6], we are able to determine the expression of the system's optimal control $\boldsymbol{u}^\star$, in terms of the state $\boldsymbol{s}$ and costate $\boldsymbol{\lambda}$, as well as the costate dynamics $\dot{\boldsymbol{\lambda}}$. By solving the resulting two-point boundary value problem, we are able to determine an optimal trajectory $[\boldsymbol{s}(t),\boldsymbol{u}(t)]^\intercal$, i.e. and open loop solution. In this work, we state this problem as a nonlinear programme and solve it

[1]It should be noted that, although Pontryagin's minimum principle requires an analytical dynamics model, the alternative set of direct methods [5] can be used in the case of black-box dynamics, giving similar expert demonstrations.

using sequential quadratic programming, where we must choose the proper optimisation variables $\boldsymbol{z}^\star$, to solve the optimal trajectory. With the shooting method of trajectory optimisation, the nominal set of optimisation variables is $\boldsymbol{z}^\star = [T, \boldsymbol{\lambda}(t_0)]$, where the initial costate variables $\boldsymbol{\lambda}(t_0)$ are chosen and the state dynamics $\dot{\boldsymbol{s}}$ and costate dynamics $\dot{\boldsymbol{\lambda}}$ are numerically integrated from an initial state $\boldsymbol{s}(t_0)$ between the times t_0 and t_f. The optimisation algorithm determines $\boldsymbol{z}^\star$, which satisfies the problems equality constraints, e.g. reaching some final state $\boldsymbol{s}(t_f) = \boldsymbol{s}_f$. For more detail on this process, the reader is referred to [5].

A. Homotopy

Under certain trajectory cost functions $\mathscr{J}$, computing an optimal trajectory can be quite difficult, one such is example is when the optimal control profile $\boldsymbol{u}^\star(t)$ is discontinuous, so-called bang-bang control. In order to decrease the burden of finding a solution, homotopy methods [26] can be employed to achieve convergence.

To elaborate further, consider two different trajectory costs, $\mathscr{L}_0(\boldsymbol{u}(t))$ and $\mathscr{L}_1(\boldsymbol{u}(t))$. One can define a combined homotopic trajectory cost function

$$\mathscr{J} = \alpha \int_{t_0}^{t_f} \mathscr{L}_0(\boldsymbol{u}(t))dt + (1-\alpha)\int_{t_0}^{t_f} \mathscr{L}_1(\boldsymbol{u}(t))dt, \qquad (1)$$

where the parameter $\alpha \in [0,1]$ characterises the balance of the objectives' priorities. Assuming $\mathscr{L}_1(\boldsymbol{u}(t))$ is the more difficult cost to deal with, we can solve the trajectory optimisation problem first with $\alpha = 0$, then use the solution to resolve the problem with a slightly increased homotopy parameter. With this process, we iteratively solve the trajectory optimisation problem until $\alpha = 1$, at which point we will have converged to a solution under the difficult cost function. If there are several different trajectory cost functions, one could extend this approach to multiple homotopy paths, as we do with the addition of β in Section III-C.

The outline of this processes is explained more programmatically in Algorithm 1, where $\boldsymbol{z}$ and α are the decision vector and homotopy parameter, respectively, and $\boldsymbol{z}^\star$ and $\alpha^\star$ are their best values so far in the homotopy loop. For every loop, the homotopy parameter is increased if a solution was found under its value within some tolerance. The loop stops once an admissible solution $\boldsymbol{z}^\star$ is found at $\alpha = 1$, at which point the set of decision vectors and homtopy parameters describing the homotopy path is returned.

Similarly, we can also perform this process of path homotopy upon the system's state. In the aim of building a large database of optimal trajectories from different initial states $\boldsymbol{s}(t_0)$, we can perform the same process by perturbing the initial state by a certain amount and resolve the trajectory optimisation problem with the previous solution as an initial guess.

This process is outlined in Algorithm 2, where $\boldsymbol{T}$ is the set of solutions paired by state $\boldsymbol{s}^\star$ and decision vector $\boldsymbol{z}^\star$, δ is the amount by which the states are perturbed, and n is the number of serial perturbations performed. The processes of perturbing the state and changing the perturbation size are expressed as functions, as one can perform these processes in multiple ways. Once n optimal trajectories are found, the perturbation loop terminates, and the set of solutions is returned. In this work, we solve for each state homotopy path in parallel from various points along an initial trajectory.

Algorithm 1: Policy homotopy

Function `homotopy_policy(`$\boldsymbol{z}^\star$, $\alpha^\star$`)`**:**
- $\boldsymbol{T} \leftarrow \emptyset$
- $\alpha \leftarrow \alpha^\star$
- **while** $\alpha \leq 1$ **do**
  - $\boldsymbol{z} = \text{solve}(\boldsymbol{z}^\star, \alpha)$
  - **if** successful($\boldsymbol{z}$) **then**
    - $\boldsymbol{z}^\star \leftarrow \boldsymbol{z}$
    - $\alpha^\star \leftarrow \alpha$
    - $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{(\alpha^\star, \boldsymbol{z}^\star)\}$
    - $\alpha \leftarrow \alpha \in [\alpha^\star, 1]$
  - **else**
    - $\alpha \leftarrow \alpha \in [\alpha^\star, \alpha]$
- **return** $\boldsymbol{T}$

Algorithm 2: State Homotopy

Function `homotopy_state(`$\boldsymbol{s}^\star$, $\boldsymbol{z}^\star$, n`)`**:**
- $\boldsymbol{T} \leftarrow \emptyset$
- **while** length($\boldsymbol{T}$) $< n$ **do**
  - $\boldsymbol{s} \leftarrow \text{perturb}(\boldsymbol{s}^\star, \delta)$
  - $\boldsymbol{z} \leftarrow \text{solve}(\boldsymbol{s}, \boldsymbol{z}^\star)$
  - **if** successful($\boldsymbol{z}$) **then**
    - $\boldsymbol{s}^\star \leftarrow \boldsymbol{s}$
    - $\boldsymbol{z}^\star \leftarrow \boldsymbol{z}$
    - $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{(\boldsymbol{s}^\star, \boldsymbol{z}^\star)\}$
    - $\delta \leftarrow \text{increase}(\delta)$
  - **else**
    - $\delta \leftarrow \text{decrease}(\delta)$
- **return** $\boldsymbol{T}$

B. Inverted pendulum swing up

For our first test case, we consider the simple first order dynamics of the nondimensional inverted pendulum implemented in [27],

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v \\ u \\ \omega \\ \sin(\theta) - u\cos(\theta) \end{bmatrix} = \boldsymbol{f}(\boldsymbol{s}, u), \qquad (2)$$

where x is the cart position, v is the cart velocity, θ is the pole angle (clockwise from upright), ω is the pole angular velocity, and $u \in [-1, 1]$ is the control input to the cart's horizontal motion.

We define a homotopic trajectory cost function

$$\mathscr{J} = \alpha \int_{t_0}^{t_f} dt + (1-\alpha) \int_{t_0}^{t_f} u^2 dt, \tag{3}$$

where the homotopy is defined from quadratically optimal control ($\alpha = 0$) to time optimal ($\alpha = 1$).

Using Pontryagin's minimum principle we define the Hamiltonian

$$\mathscr{H} = \lambda_x v + \lambda_v u + \lambda_\theta \omega + \lambda_\omega (\sin(\theta) - u\cos(\theta)) + \alpha + (1-\alpha) u^2 \tag{4}$$

We then minimise the Hamiltonian with respect to the control input u to find the optimal control as a function of both states and costates

$$u^\star = \frac{\lambda_v - \lambda_\omega \cos(\theta)}{2(\alpha - 1)}. \tag{5}$$

The quadratically optimal control ($\alpha = 0$) becomes

$$u^\star = \frac{\lambda_\omega \cos(\theta)}{2} - \frac{\lambda_v}{2} \tag{6}$$

As we drive the homotopy parameter to unity, the optimal switching function becomes

$$\sigma = \lambda_v - \lambda_\omega \cos(\theta), \tag{7}$$

which defines the optimal control with bound considerations as

$$u^\star = \begin{cases} -1 & \text{if } \sigma < 0 \\ 1 & \text{if } \sigma > 0 \end{cases}, \tag{8}$$

where it should be noted that the edge case $\sigma = 0$ is only encountered instantaneously if at all, so singular control analysis is not necessary here.

Lastly, we compute the costate equations of motion

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{s}} \mathscr{H} = \begin{bmatrix} 0 \\ -\lambda_x \\ -\lambda_\omega (u\sin(\theta) + \cos(\theta)) \\ -\lambda_\theta \end{bmatrix}. \tag{9}$$

Considering the task of swinging up the pendulum, we nominate initial and terminal state constraints

$$\boldsymbol{s}(t_0) = \begin{bmatrix} 0 & 0 & \pi & 0 \end{bmatrix}^\intercal \tag{10}$$

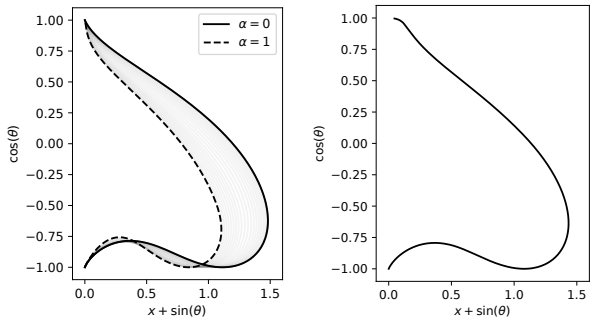
$$\boldsymbol{s}(t_f) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\intercal, \tag{11}$$

as well as the free-time condition

$$\mathscr{H}(\boldsymbol{s}(t), \boldsymbol{\lambda}(t), u(t)) = 0 \tag{12}$$

to form a two-point boundary value problem. With the shooting method parameterisation, the decision vector is $\boldsymbol{z} = [T, \boldsymbol{\lambda}(t_0)]^\intercal$, where $T = t_f - t_0$ is the trajectory duration and $\boldsymbol{\lambda}(t_0)$ is the inital costate vector.

Following Algorithm 1, we compute our initial path homotopy between quadratic and time optimal control, as shown in Figure 2.



(a) (b)

Fig. 2: The state trajectories corresponding to the controls in Figure 1. Each trajectory shows the path of the tip of the pendulum, starting from straight down $(0,-1)$ to straight up $(0,1)$. The homotopy path between quadratically optimal control ($\alpha = 0$) and effort/time optimal control ($\alpha = 1$) is shown in (a), and the solution of running the trained ANN with $\alpha = 0$ initially, switching to 0.5 and then to 1.0 is shown in (b).

C. Spacecraft orbit transfer

As an increase in complexity of boundary conditions and dimensionality, we now consider the problem of optimising a spacecraft's trajectory from Earth to the orbit of Mars. Its heliocentric dynamics are given by the following system of ordinary differential equations

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \dot{\boldsymbol{r}} \\ \dot{\boldsymbol{v}} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \frac{Tu}{m}\hat{\boldsymbol{u}} - \frac{\mu}{r^3}\boldsymbol{r} \\ -\frac{Tu}{I_{sp} g_0} \end{bmatrix} = \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{u}), \tag{13}$$

where the state variables are: position $\boldsymbol{r} = [x, y, z]^\intercal$, velocity $\boldsymbol{v} = [v_x, v_y, v_z]^\intercal$, and mass m. The control inputs are the thrust throttle $u \in [0,1]$ and direction $\hat{\boldsymbol{u}} = [\hat{u}_x, \hat{u}_y, \hat{u}_z]^\intercal$. The constant parameters governing the dynamics are: maximum thrust capability $T = 0.2$ $[N]$, specific impulse $I_{sp} = 2500$ $[s]$, gravitational acceleration at sea level $g_0 = 9.81$ $[m \cdot s^{-1}]$, and standard gravitational parameter of the sun $\mu = 1.3271 \times 10^{20}$ $[m^3 \cdot s^{-2}]$.

We again construct a homotopic trajectory cost function,

$$\mathscr{J} = \int_0^{t_f} \beta(\alpha + u(1-\alpha))\, dt + \int_0^{t_f} u^2 (1-\beta)\, dt, \tag{14}$$

where we have added a secondary homotopy parameter $\beta \in [0,1]$ as a means to feasibly arrive to a solution of the problem at $\alpha \in [0,1], \beta = 1$ through homotopy continuation. The homotopy paths are defined as

- quadratic to effort: $\alpha = 0, \beta \in [0,1]$
- quadratic to time: $\alpha = 1, \beta \in [0,1]$
- effort to time: $\alpha \in [0,1], \beta = 1$

Using Pontryagin's minimum principle we then define the Hamiltonian

$$\mathscr{H} = \boldsymbol{\lambda}_r \cdot \boldsymbol{v} + \boldsymbol{\lambda}_v \cdot \left(\frac{Tu}{m}\hat{\boldsymbol{u}} - \frac{\mu}{r^3}\boldsymbol{r} \right) + \lambda_m \left(-\frac{Tu}{I_{sp} g_0} \right) + \beta(\alpha + u(1-\alpha)) + u^2(1-\beta) \tag{15}$$

Firstly, to find the optimal thrust direction $\hat{\boldsymbol{u}}^\star$, we isolate the portion of the Hamiltonian $\mathscr{H}$ that depends on it: $\frac{Tu}{m}(\hat{\boldsymbol{u}}\cdot\boldsymbol{\lambda}_v)$. Considering that T, m, and u are positive numbers, the optimal thrust direction must be directed opposite of the velocity costate $\boldsymbol{\lambda}_v$ to minimise the Hamiltonian:

$$\hat{\boldsymbol{u}}^\star = -\frac{\boldsymbol{\lambda}_v}{\lambda_v}. \tag{16}$$

Substituting the optimal thrust direction back into the original Hamiltonian we obtain

$$\mathscr{H} = \boldsymbol{\lambda}_r\cdot\boldsymbol{v} - \lambda_v\frac{Tu}{m} - \boldsymbol{\lambda}_v\cdot\left(\frac{\mu}{r^3}\boldsymbol{r}\right) - \lambda_m\left(\frac{Tu}{I_{sp}g_0}\right) + \beta\left(\alpha + u(1-\alpha)\right) + u^2(1-\beta). \tag{17}$$

Minimising the new Hamiltonian with respect the control throttle u we find

$$u^\star = \frac{1}{2(1-\beta)}\left(\frac{T(\boldsymbol{\lambda}_v\cdot\boldsymbol{\lambda}_v)}{\lambda_v m} + \frac{T\lambda_m}{I_{sp}g_0} + \beta(\alpha-1)\right). \tag{18}$$

In the case of quadratic control ($\beta = 0$) the optimal control becomes

$$u^\star = \frac{T}{2}\left(\frac{\boldsymbol{\lambda}_v\cdot\boldsymbol{\lambda}_v}{m\lambda_v} + \frac{\lambda_m}{I_{sp}g_0}\right). \tag{19}$$

Driving the homotopy parameter β to unity, we obtain the optimal control switching function

$$\sigma = \frac{\boldsymbol{\lambda}_v\cdot\boldsymbol{\lambda}_v}{m\lambda_v} + \lambda_m + 1 - \alpha. \tag{20}$$

With a choice of homotopy parameter $\alpha\in[0,1]$, describing the tradeoff between effort optimal control ($\alpha = 0$) and time optimal control ($\alpha = 1$), the optimal control is defined with bound constraints as

$$u^\star = \begin{cases} 0 & \text{if } \sigma < 0 \\ 1 & \text{if } \sigma > 0 \end{cases}. \tag{21}$$

Finally, we compute the costate equations of motion

$$\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{s}}\mathscr{H} = \begin{bmatrix} \frac{\mu}{r^3}\boldsymbol{\lambda}_v - \frac{3\mu}{r^5}(\boldsymbol{\lambda}_v\cdot\boldsymbol{r}) \\ -\boldsymbol{\lambda}_r \\ \frac{Tu}{m^2}(\boldsymbol{\lambda}_v\cdot\hat{\boldsymbol{u}}) \end{bmatrix}. \tag{22}$$

As the progression of that spacecraft's mass m is affected by the control input u, we nominate the transversality condition on it

$$\lambda_m(t_f) = 0. \tag{23}$$

We enforce transversality on the mean anomaly of the spacecraft's final orbit so that the optimiser can choose the best state along Mars's orbit with

$$\frac{r^3(\boldsymbol{\lambda}_v\cdot\boldsymbol{v}) - \mu(\boldsymbol{\lambda}_r\cdot\boldsymbol{r})}{\sqrt{\mu^2(\boldsymbol{r}\cdot\boldsymbol{r}) + (\boldsymbol{v}\cdot\boldsymbol{v})(\boldsymbol{r}\cdot\boldsymbol{r})^3}} = 0, \tag{24}$$

see [23] for details. With these boundary constraints and the free-time horizon condition, the culminating two-point boundary value problem is formed. With the shooting method parameterisation, the decision vector is $\boldsymbol{z} = [T, M(t_f), \boldsymbol{\lambda}(t_0)]^\intercal$, where $T = t_f - t_0$ is the time of flight, $M(t_f)$ is the mean anomaly of the final orbit, and again $\boldsymbol{\lambda}(t_0)$ is the initial costate vector.

Following Algorithm 1 again, we compute each homotopy path, as shown in Figure 3 and 4, for an interplanatary transfer from Earth's position and velocity to somewhere along Mars's orbit[2]

IV. Learning

In this section we will first describe the data sets generated in the previous section in detail, then we outline the neural network architecture used and finally evaluate the result of the learning process.

A. Datasets

In order to learn the homotopy path of the control policies examined, we first start by assembling databases from which to learn. We use Algorithm 2 to generate an initial database of optimal control trajectories from various initial states with a specific homotopy parameter configuration. We then compute the homotopy path of the control policy for each trajectory in the database in parallel, using Algorithm 1. Finally, we conflate this set of trajectories into a single unified dataset of state-control pairs $[\boldsymbol{s}(t), \alpha(t), \boldsymbol{u}(t)]^\intercal$, where the state is augmented by the homotopy parameter α.

For the inverted pendulum swingup task, we first generate an initial database of quadratically optimal control trajectories, $\alpha = 0$. We then compute the homotopy path of the control policy for each trajectory until effort/time optimal control is achieved, $\alpha = 1$, keeping the intermediate trajectories of the path, $\alpha\in(0,1)$. We generate a database of 945 trajectories, equating to $577,480$ state-control pairs.

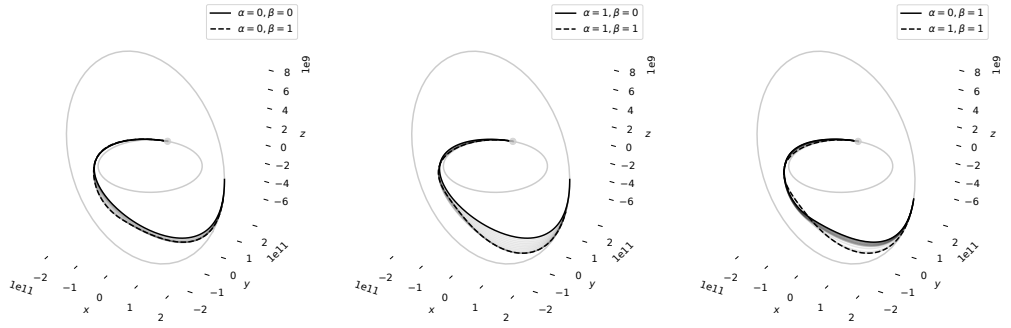
For the spacecraft orbit transfer, we generate and initial database of quadratically optimal control trajectories, $\alpha = \beta = 0$. We then compute the homotopy path between quadratically optimal control and effort optimal control, $\alpha = 0, \beta\in[0,1]$, for every trajectory in the database. With the effort optimal trajectories of the resulting database, we then compute the homotopy path between effort and time optimal control, $\alpha\in[0,1], \beta = 1$, keeping the intermediate trajectories. We generate a database of 5525 trajectories, equating to $2,124,668$ state-control pairs.

B. Neural network architecture

We extend the state of each system to include the homotopy parameter as an input to a feedfoward ANN, as reflected in Figure 6, in order to learn the mapping between the chosen homotopy parameter and the resulting optimal control policy $[\boldsymbol{s}, \alpha]^\intercal \mapsto \boldsymbol{u}^\star$.

For the output layers of our networks, we apply the hyperbolic tangent activation function. For the pendulum, the single output is left as it is, $u\in[-1,1]$. For the spacecraft, three outputs are generated; the first is scaled to the thrust bounds $u_1 \mapsto u\in[0,1]$, the second to an inclination angle $u_2\mapsto\phi\in[0,\pi]$, and the third to a azimuthal angle $u_3\mapsto\theta\in[0,2\pi]$. The two spherical angles are then

[2]We pick the starting time to be 0 $MJD2000$ (using the so-called Modified Julian Date notation).



(a) (b) (c)

Fig. 3: The state trajectories of the homotopy paths between (a) quadratic and effort optimal control, (b) quadratic and time optimal control, (c) effort and time optimal control.

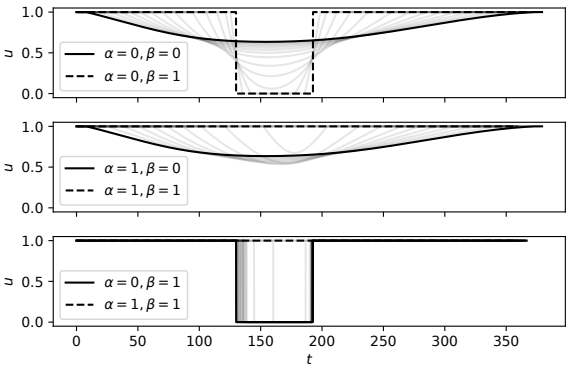


Fig. 4: The three different control policy homotopies from top to bottom: quadratically optimal control to effort optimal control ($\alpha = 0, \beta \in [0,1]$), quadratically optimal control to effort optimal control ($\alpha = 1, \beta \in [0,1]$), quadratically optimal control to effort optimal control ($\alpha \in [0,1], \beta = 1$)

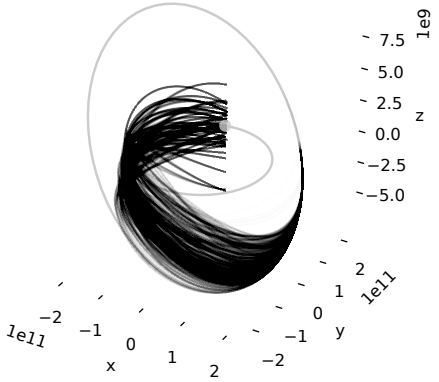


Fig. 5: Database of $5,525$ optimal control trajectories, corresponding to $2,124,668$ state-control pairs, for a variety of α for the spacecraft orbit transfer problem.

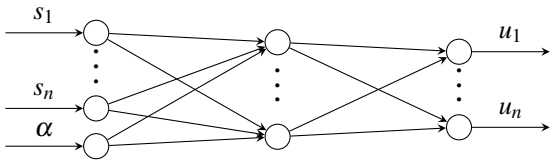


Fig. 6: Homotopic state feedback controller represented by a multi-layer ANN with arbitrarily many hidden units, mapping a system's state $\boldsymbol{s} = [s_1, \ldots, s_n]^\intercal$ to a control $\boldsymbol{u} = [u_1, .., u_n]^\intercal$, given a particular objective priority $\alpha \in [0,1]$.

mapped to a unit vector describing the thrust direction $\hat{u} = [\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)]^\intercal$.

For our ANN architectures we restrict our attention to fully connected multi-layer perceptrons with varying numbers of hidden layers and a uniform amount of nodes per layer. We express the networks' hidden architectures in the format Nodes $\times$ Layers, where $n \times m$ would express a network of m hidden layers, each having n nodes. For every layer except the last, we apply the softplus activation function instead of the more conventionally used rectified linear units, as it results in smoother and more meaningful control strategies, as indicated in [24]. For training we use a mean squared error (*MSE*) loss function coupled with the Adam training algorithm [28] at a learning rate of 1×10^{-3} with a decay rate of 1×10^{-5} to train our networks.

C. Evaluation

To evaluate our method, we consider neural networks of varying numbers of hidden layers and number of nodes per layer. We train these networks for $4,000$ episodes independently with a randomised subset of $20,000$ state-control pairs from each dataset, reserving 10% as a testing set. We judge the performance of each network in the state-control regression task by the training and testing loss it has achieved. Since success in the regression task does not necessarily guarantee that the system controlled by the trained network will behave optimally as expected, we also evaluate the performance of our networks in the simulated tasks with the policy trajectory optimality metric given in [24], reflecting how well the networks learn the optimal behaviour.

We find that our networks are able to achieve good *MSE* in the regression task (Tables Ia and Ib) which results in near-optimal trajectories (Tables IIa and IIb) when the ANN is used to control the system directly. We observe that both

increasing the node breadth for a given number of layers and increasing the number of layers for a given node breadth generally increases both the regression performance of the networks and the optimality of their resulting trajectories. Interestingly we find that, for the implemented ANNs, the pendulum swingup task, although having a mathematically simpler description, proves to be a more difficult problem, possibly due to its greater number of control switches. In the spacecraft orbit transfer case, our ANNs are able to learn the mapping between the two discontinuous control profiles of fuel and time optimal control, as reflected in Figure 10. We observe, however, that employing the networks to predict a full vector output in the action space may cause adverse interactions between the individual controls, as can be seen in the sudden spikes of the thrust direction $\hat{\boldsymbol{u}}$ around the throttle u switching points in Figures 8 and 10. Despite this, the networks still perform well given that the control switching is instantaneous.

Nodes × Layers	Training Loss	Testing Loss
50×2	0.043434	0.049832
50×4	0.034315	0.043264
100×2	0.038298	0.044142
100×4	0.029807	0.036758

(a) Inverted pendulum swingup

Nodes × Layers	Training Loss	Testing Loss
50×2	0.014710	0.013711
50×4	0.014066	0.012860
100×2	0.018373	0.018608
100×4	0.012590	0.011232

(b) Spacecraft orbit transfer

TABLE I: Regression performances on the datasets.

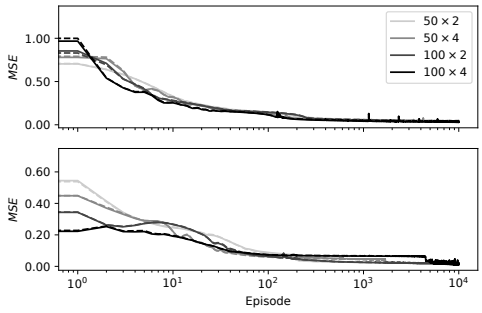


Fig. 7: The training (solid lines) and validation (dashed line) history of the networks by architecture, reflected by Tables I, for the inverted pendulum swingup (top) and the interplanatary spacecraft orbit transfer (bottom).

V. Conclusions

We presented a novel approach to learning a family of optimal state feedback control policies and evaluated it on two common dynamical systems. We've shown that the continuation between objectives for optimal state feedback policies can be learnt by extending the input state with a homotopy parameter, describing the trade off in priority between different objectives. We've shown that our networks

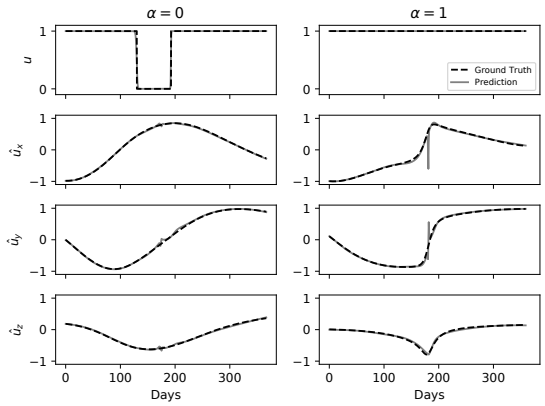


Fig. 8: The approximation of the nominal control profiles for the spacecraft orbit transfer in Figure 3c by the best ANN in Table Ib.

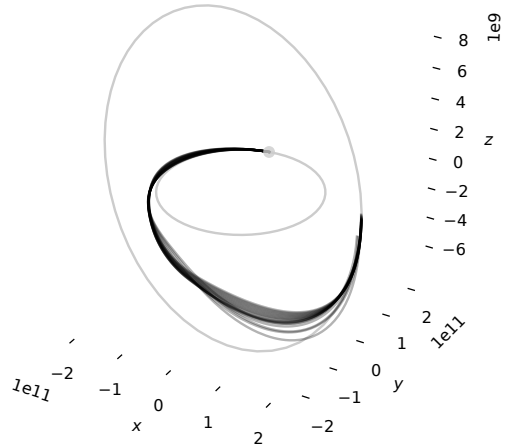


Fig. 9: The spacecraft orbit transfer as controlled by the best ANN in Table Ib.

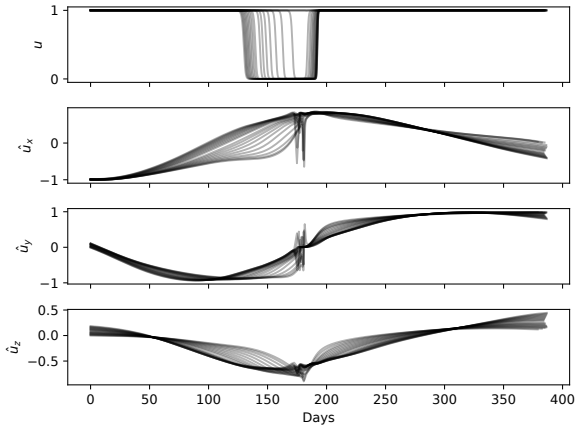


Fig. 10: The control profiles of the thrust magnitude u and direction $\hat{\boldsymbol{u}} = [\hat{u}_x, \hat{u}_y, \hat{u}_z]^\intercal$ of the spacecraft orbit transfer in Figure 9 as controlled by the best ANN in Table Ib.

	Nodes × Layers			
α	50×2	50×4	100×2	100×4
0.0	8.366217	0.201963	6.911897	0.625839
0.1	2.310301	2.667611	2.722436	2.669494
0.2	1.270388	4.418089	1.152010	3.045441
0.3	3.473601	4.555533	3.798529	5.253961
0.4	4.832105	3.444665	4.429260	4.477721
0.5	3.260343	1.645225	2.438281	2.761684
0.6	1.209908	0.616588	0.729584	1.150946
0.7	0.669214	0.430044	0.279920	0.144534
0.8	0.497213	0.514005	0.356340	2.509390
0.9	0.904339	0.797179	0.485849	0.836367
1.0	2.309510	1.775131	1.027916	1.200968
Mean	2.645740	1.915094	2.212002	2.243304

(a) Inverted pendulum swingup

	Nodes × Layers			
α	50×2	50×4	100×2	100×4
0.0	5.097919	0.163055	0.264219	0.094422
0.1	3.551325	0.057574	0.109408	0.073708
0.2	2.160594	0.355176	0.468579	0.353486
0.3	0.954027	0.665326	0.754457	0.675646
0.4	0.117137	0.979295	0.975896	1.031487
0.5	0.519775	0.344638	0.242022	0.455965
0.6	0.832056	0.210354	0.353348	0.051248
0.7	0.940100	0.622716	0.765109	0.439597
0.8	1.231413	1.173717	1.259935	1.005639
0.9	0.799833	1.096328	1.055267	1.012649
1.0	1.103462	0.189937	0.768724	0.044418
Mean	1.573422	0.532556	0.637906	0.476206

(b) Spacecraft orbit transfer

TABLE II: Optimality gap (%) between the ANN controlled trajectory and the optimal trajectory for each neural network architecture (columns) and homotopy parameter (rows) in the (a) pendulum swing-up problem and (b) spacecraft orbit transfer problem.

achieve good *MSE* values in the database regression task and that near-optimal control can be achieved across a spectrum of objective priority trade offs. We observe that both increasing the node breadth for a given number of layers and increasing the number of layers given a node breadth increases both the regression and trajectory optimality of the networks. We suspect that better performance would be achieved by training an ANN for each of the given system's actions, to avoid adverse interactions around discontinuities, or by increasing both the number of layers and nodes per layer.

ACKNOWLEDGMENT

This work was supported by Stiftelsen for Strategisk-Forskning (SSF) through the Swedish Maritime Robotics Centre (SMaRC) (IRC15-0046).

REFERENCES

[1] I. M. Ross and F. Fahroo, "Issues in the real-time computation of optimal control," *Mathematical and computer modelling*, vol. 43, no. 9-10, pp. 1172–1188, 2006.

[2] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[3] D. Izzo, D. Hennes, L. F. Simões, and M. Märtens, "Designing complex interplanetary trajectories for the global trajectory optimization competitions," in *Space Engineering*. Springer, 2016, pp. 151–176.

[4] C. Shen, Y. Shi, and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5796–5805, 2018.

[5] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam, 2010, vol. 19.

[6] L. S. Pontryagin, V. G. Boltyanshii, R. V. Gamkrelidze, and E. F. Mishenko, *The Mathematical Theory of Optimal Processes*. New York: John Wiley and Sons, 1962.

[7] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2010.

[8] R. R. Nair and L. Behera, "Robust adaptive gain higher order sliding mode observer based control-constrained nonlinear model predictive control for spacecraft formation flying," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 367–381, 2018.

[9] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6753–6760.

[10] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[11] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[12] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[13] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control." in *Robotics: Science and Systems*. Rome, Italy, 2015.

[14] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3223–3230.

[15] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[16] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, no. 3, pp. 683–694, 2014.

[17] S. Levine, "Exploring deep and recurrent architectures for optimal control," *arXiv preprint arXiv:1311.1761*, 2013.

[18] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile off-road autonomous driving using end-to-end deep imitation learning," *arXiv preprint arXiv:1709.07174*, 2017.

[19] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," in *ASME 2018 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2018, pp. V003T37A012–V003T37A012.

[20] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3342–3349.

[21] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization." in *Robotics: Science and Systems*, 2014, pp. 5–32.

[22] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: study on landing problems," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.

[23] D. Izzo, C. Sprague, and D. Tailor, "Machine learning and evolutionary techniques in interplanetary trajectory design," *arXiv preprint arXiv:1802.00180*, 2018.

[24] D. Tailor and D. Izzo, "Learning the optimal state-feedback via supervised imitation learning," *arXiv preprint arXiv:1901.02369*, 2019.

[25] I. C. W. Wampler *et al.*, *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.

[26] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[27] C. I. Sprague and P. Ögren, "Adding neural network controllers to behavior trees without destroying performance guarantees," *CoRR*, vol. abs/1809.10283, 2018. [Online]. Available: http://arxiv.org/abs/1809.10283

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.