Oracle for Guidance with Deep Neural Networks in Reusable Launch Vehicle Landing*

J.M. Igreja
Mechanical Engineering Department
INESC-ID/ISEL-IPL
Lisboa, Portugal
jigreja@deea.isel.ipl.pt

J.M. Lemos
Electrotechnical Eng. and Computers Department
INESC-ID/UL-IST
Lisboa, Portugal
jlml@inesc-id.pt

***Abstract*—Oracles are of paramount importance for Deep Neural Networks training. In this paper, an oracle developed for landing reusable launch vehicles is created from a linearizing feedback control law that can perform a prescribed landing trajectory tracking. The oracle is then used to train a Deep Neural Network that can be used as a guidance system for landing maneuvers. Verification is performed by Monte-Carlo.**

***Index Terms*—Neural networks, Oracles for training, Feedback linearizing, Reusable Launch Vehicle**

I. Introduction

About: DNN, Oracles for training, RLVs and landing trajectories.

The objective of this work is to find landing trajectories for a reusable launch vehicle (RLV), and then train an Deep Neural Network (DNN) to perform the task. Landing is understood to be the last part of reentry, when the RLV is at a few kilometers from the ground in an off-set position in relation to the landing site, traveling with a know descending velocity and showing a small inclination angle in relation to the site vertical. The problem is solved, in a vertical landing plane, for a RLV with lateral thrusters in addition to the gimbaled thrust engine. Using a vertical landing model a control solution provides smooth descending trajectories, including dynamical states and actuators responses. Moreover, an ANN is trained with validated solutions for different flight conditions and control parameters. Verification is done by Monte-Carlo techniques. Guidance is obtained through a feedback linearizing transformation and the use of simple lead-lag feedback controllers, that can be tuned to generate all signals needed for the planned landing maneuver. Constrains to the problem satisfaction are used for ruling out from training undesired or unfeasible maneuvers.

An oracle is, in general, an entity capable of solving some problem, which for example may be a decision problem or a function problem. A test oracle (or just oracle) is a mechanism for determining whether a test has passed or failed. In this context, a landing oracle is a computational procedure able to generate all the signals needed to guide a vehicle during landing. This trajectory will be validated by the oracle and then used to supervise the training of a *digital twin*, in this case, a suitably trained DNN.

*financed by FCT, pluriannual INESC-ID funding UIDB/50021/2020

In the literature there is a considerable number of relevant papers and thesis dedicated to the RLVs landing problem, about ANN training for guidance during landing and V&V for supervised learning of ANNs acting as controllers. For instance, in [7] a real-time successive convexification algorithm for a generalized free-final-time 6-degree-of-freedom powered descent guidance problem is presented. It represents a novel formulation in the optimal control, and enables a number of interesting applications, including velocity-triggered angle of attack constraints and range-triggered line of sight constraints. The algorithm converts the resulting generalized powered descent guidance problem from a non-convex free-final-time optimal control problem into a sequence of tractable convex second-order cone programming subproblems. With the aid of virtual control and trust region modifications, these subproblems are solved in succession until convergence is attained. In another paper [6], the effectiveness of optimizing a deep recurrent neural network with gated recurrent unit modules to control a sophisticated and highly nonlinear flight vehicle is demonstrated. An optimization procedure that leverages ideas from Lyapunov funnels and robust nonlinear control to create a robust and high performance controller that tracks time-varying trajectories is described. The controller is trained to negate uncertainties in the aerodynamic tables which leads to significant error reduction compared to typical baseline controllers for flight control systems.

A supervised learning framework to approximate a model predictive controller (MPC) with reduced computational complexity and guarantees on stability and constraint satisfaction is proposed in [4]. The framework can be used for a wide class of nonlinear systems. Any standard supervised learning technique (e.g. neural networks) can be employed to approximate the MPC from samples. In order to obtain closed-loop guarantees for the learned MPC, a robust MPC design is combined with statistical learning bounds. The MPC design ensures robustness to inaccurate inputs within given bounds, and Hoeffding's Inequality is used to validate that the learned MPC satisfies these bounds with high confidence. The result is a closed-loop statistical guarantee on stability and constraint satisfaction for the learned MPC. In [9] a new method to design the controller for Mars capsule atmospheric entry using deep neural networks and flight-proven Apollo

entry data is presented. The controller is trained to modulate the bank angle with data from the Apollo entry simulations. The neural network controller reproduces the classical Apollo results over a variation of entry state initial conditions. The deep neural network is only trained with data from Apollo reentry simulation in an Earth model and works in both Earth and Mars environments. It achieves the desired landing accuracy for a Mars capsule. This method works with both linear and nonlinear integration and can generate the bank angle commands in real-time without a pre-stored trajectory.

This paper is organized as follows, after the introduction, in section 2, the RLV nonlinear model is presented and a simplified version for small angles is obtained, the made assumptions are also discussed. Feedback linearizing control is briefly discussed in section 3, furthermore, the exact linear model and controls for the RLV are derived. In section 4 DNN training is explained and results are validated using Monte Carlo. Conclusions are drawn in section 5.

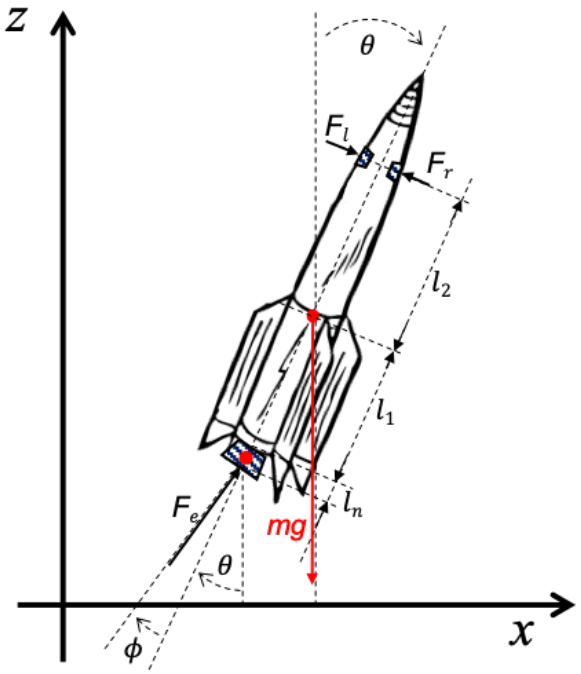

Fig. 1. RLV schematics. The center of gravity and center of pressure are marked with red dots.

II. RLV Landing Model

In this section after introducing a simplified nonlinear model for landing, a suitable exact linearized model is obtained in order to be used later to generate landing trajectories from a control like strategy.

Some simplifying assumptions must be made to write a low complexity landing RLV dynamic model:

- the model is developed in a vertical plane and the vehicle is trimmed [2], [8];
- A simplified formulation of the aerodynamics forces has been considered, and the effects of the wind are not embedded into the guidance problem;
- The center of pressure is considered constant in the body-fixed frame, and the center of mass is considered constant during flight;
- The vehicle is equipped with a rocket engine that can be gimballed symmetrically about two axes up to a maximum gimbal angle, and side Nitrogen gas thrusters mounted in top of the vehicle;
- The engine can be throttled between minimum and maximum thrust values and once the engine is ignited, it cannot be shut off until the terminal condition is reached;
- The side gas thursthers can be used during the final landing maneuver for some time;
- The launcher is modelled as a rigid-body;
- The moment of the inertia of the rocket is considered constant during the flight.
- Mechanical higher order terms like flexible modes and sloshing are neglected for the guidance model;
- The time of flight is fixed;
- The planetary rotation effects are neglected due to the relatively short duration of the studied problem.

It is worth mentioning that at least one RLV is equipped with a total of 8 nitrogen cold gas thrusters that are mounted towards the top of the first stage. Like the gimbaled main engines, the cold gas thrusters are used to control the orientation of the rocket. In falcon 9, they are particularly useful for the flip maneuver after stage separation because of the large lever arm between the thrusters and the rocket's center of mass. They are also used to control the rocket at times during the flight when the gimbaled main engines are shut off.

Considering all the assumptions and Fig. I, the vertical plane XZ landing model is given by

$$\begin{aligned} m\dot{v}_x &= F_T \sin(\theta+\phi) + F_s \cos\theta + D(z,V)\sin\theta \\ m\dot{v}_z &= F_T \cos(\theta+\phi) + F_s \sin\theta - mg + D(z,V)\cos\theta \\ J\dot{\omega} &= -F_T \sin\phi(l_1 + l_n\cos\phi) + l_2 F_s \\ J_T\ddot{\phi} &= \tau, \end{aligned} \tag{1}$$

where θ is a pitch inclination angle, with the angular rate ω, the vehicle should be aligned with the vertical Z axis, ϕ is the gimbal angle, if $\phi > 0$ the vehicle should tilt to the left, about the center of gravity. Velocities v_x and v_z are the velocities components in X and Z directions, x and z are the position in Z and X. The thrust force is F_T and $F_s = F_l - Fr$ is the lateral thrust. Drag is given by $D(z,V)$. Solving for translational and rotational forces in X, Z directions, and as well for (rotational) pitch torque with respect to the rocket's COG leads these equations. Additionally

$$\dot{m} = -\frac{F_T}{I_s g_0} - \alpha_s F_s - A_n P_{amb} \tag{2}$$

is the mass depletion, where I_s (specific impulse in vacuum), g_0 (Earth's gravity), A_n (noozle exit area) and P_{amb} (atmo-

spheric pressure) are parameters, details can be found in [7]. The atmospheric drag force component is obtained using:

$$D(z,V) = \frac{1}{2} C_D \rho(z, T_{amb}) V^2 S_D, \tag{3}$$

being $V^2 = v_x^2 + v_z^2$ the velocity magnitude in XZ, and $\rho(z, T_{amb})$ the atmospheric density. A simplified model can be obtained for small angle deviations considering $\cos\alpha \approx 1$ and $\sin\alpha \approx \alpha$, for $\alpha << 1$

$$\begin{aligned} m\dot{v}_x &= F_T(\theta + \phi) + F_s + D(z,V)\theta \\ m\dot{v}_z &= F_T(1 - \theta\phi) + F_s\theta - mg + D(z,V) \\ J_y\dot{\omega} &= -F_T\phi(l_1 + l_n) + l_2 F_s \\ J_T\ddot{\phi} &= \tau. \end{aligned} \tag{4}$$

This simplified model should be useful for designing a nonlinear landing control law in the next section.

III. Control Strategy for the Oracle

A control strategy devised for the RLV model is twofold. Firstly, an exact linearizing transformation is derived. Secondly, for the linear and decoupled dynamics, three single-input single-output controllers are tuned to produce smooth closed-loop trajectories. More information about feedback linearizing control can be found in [5].

The vehicle landing model is nonlinear, and it is affine by considering the actuators input vector $\mathbf{v} = [\ F_T \quad F_T\phi \quad F_s\ ]^T$, and so

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{v}, \tag{5}$$

with $\mathbf{x} = [\ \theta \quad \omega \quad z \quad v_z \quad x \quad v_x\ ]^T$. Using a feedback linearizing approach for writing the control law with input u_θ, it yields for pitch

$$\begin{aligned} \dot{\omega} &= u_\theta \\ \phi &= \frac{l_2 F_s - J_y u_\theta}{F_l(l_1 + l_n)}. \end{aligned} \tag{6}$$

Using the same procedure for altitude and distance it yields

$$\begin{aligned} \dot{v}_z &= u_z \\ F_T &= \frac{(u_z + g)m + F_s - D}{1 - \theta\phi}, \end{aligned} \tag{7}$$

and

$$\begin{aligned} \dot{v}_x &= u_x \\ F_s &= u_x - F_T(\theta + \phi) - D. \end{aligned} \tag{8}$$

Defining the transformation $W(\theta)E = B(\mathbf{u}, \mathbf{x})$, where

$$\begin{bmatrix} 0 & l_1 + l_n & -l_2 \\ 1 & -\theta & -1 \\ \theta & 1 & 1 \end{bmatrix} \begin{bmatrix} F_T \\ F_T\phi \\ F_s \end{bmatrix} = \begin{bmatrix} -J_y u\theta \\ m(u_z + g) - D \\ mu_x - D\theta \end{bmatrix}, \tag{9}$$

with determinant $\Delta = \det(W(\theta))$,

$$\Delta = -(\theta + 1)l_1 - (\theta^2 + 1)l_2 - (\theta + 1)l_n \neq 0, \ \ \forall\theta. \tag{10}$$

Further, the transformation is well-posed except when F_T approaches zero, in that case, the controllability from the gimbal angle will be lost. The assumption needed is that the vehicle will not touch the ground and by so $F_T \geq F_{Tmin} > 0$. If the simplified model with the transformation are considered, the model is exact linearized with relative degree of one [5], and

$$\dot{\mathbf{x}} = A_f\mathbf{x} + B_f\mathbf{u}, \tag{11}$$

where $\mathbf{x} = [\ \theta \quad \omega \quad z \quad v_z \quad x \quad v_x\ ]^T$, $\mathbf{u} = [\ u_\theta \quad u_z \quad u_x\ ]^T$ are respectively the states and control inputs. Furthermore,

$$A_f = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B_f = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{12}$$

Note that mass depletion nonlinear state is not observable in (11) and remains in the dynamics [5].

The feedback controllers can be chosen as lead-lag filters given by

$$\begin{aligned} \dot{\xi} &= -\frac{1}{\tau_p}\xi + \frac{k_p}{\tau_p}e \\ c &= \left(1 - \frac{\tau_z}{\tau_p}\right)\xi + k_p\frac{\tau_z}{\tau_p}e, \end{aligned} \tag{13}$$

where $e = r - y$ is the error between the reference and the output, c is the control action, and $\tau_p, \tau_z > 0$ are time constants. The outputs are, as already mentioned, pitch θ, descending velocity v_z, and distance x.

The reference value for the pitch is $\theta_r = 0$, for the distance $x_r = x(0)(1 - \frac{t}{T_f})$, and for velocity $v_{zr}(t) = -\lambda_d z(0) e^{-\lambda_d t}$, where $\lambda_d = \frac{\ln z(0) - \ln z(T_f)}{T_f}$, and T_f the landing time duration (final time).

The landing maneuver can now be calculated after tuning the three controllers, given the initial conditions and references, for a fixed final time. All the trajectories are computed using the nonlinear model eq. (1).

One example, after reentry, the last stage of the descent starts at 2 kilometers high ad finishes at the landing location, hovering at approximately 1 meter from the soil. Fig. 2 depicts the trajectory found for a 100 meters offset distance, and a 5 m/s descending velocity. Table I contains the numerical values for the RLV parameters.

As mentioned before, the landing trajectories must be tested and validated against a *landing envelope*, which includes safety, operational and physical constraints. Passing the validation they should be in the oracle for training. When a vehicle is pushed, for instance by diving it at high speeds, it is said to be flown "outside the envelope", something considered rather dangerous.

The oracle can produce many different trajectories from a smooth stable nominal one, just by considering initial conditions dispersion in a certain interval, different tuning parameters, and different final times. The result is collected or discharged by the landing envelope test. In this case,

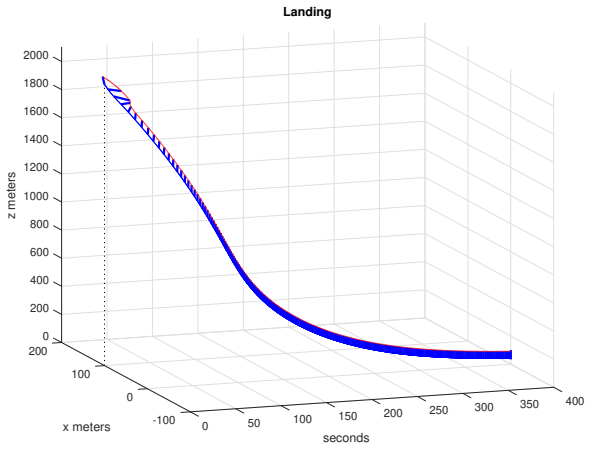

Fig. 2. Landing trajectory at XZ vertical plane.

TABLE I
RLV PARAMETERS.

	RLV parameters	
	Value	***Units***
m	10.0×10^3	kg
J_y	8.4×10^4	$kg\,m^2$
g	9.8	$m\,s^{-1}$
l_1	3.5	m
l_2	5.0	m
l_n	0.5	m
D	1.5	m
S_D	1.8	m^2
C_D	0.075	$adm.$

only typical constraints for the actuators along with the pitch constraint were included in the landing envelope, given by

$$\begin{aligned} 0\ N \leq\ & F_T & \leq 2 \times 10^5\ N \\ -1000\ N \leq\ & F_s & \leq 1000\ N \\ -15^\circ \leq\ & \phi & \leq 15^\circ \\ -10^\circ \leq\ & \theta & \leq 10^\circ. \end{aligned} \tag{14}$$

IV. DNN TRAINING

In this section a DNN is trained using the landing oracle as supervisor. Once trained and a V&V performed, the neural network can act as a guidance block, or high-level control depicted in Fig. 3.

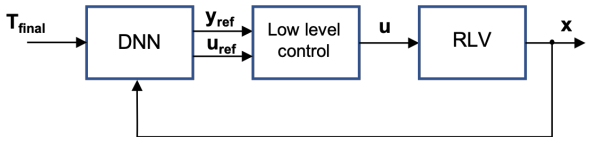

Fig. 3. Guidance block diagrams. The neural network acts as a high-level control able to send command signals to low-level control.

A feedforward neural network (FNN) is a composition of many functions, more precisely $h+1$, where h is the number of hidden layers or functions, known as depth, given by

$$y = f_y \circ f_h \circ \ldots \circ f_1(u), \tag{15}$$

in each hidden layer ($i = 1, \ldots, h$) following operation is performed

$$z_i = f_i(v_i) = f_i(w_i z_{i-1} + b_i) \tag{16}$$

where z_i collects in a vector all the outputs of layer i, and w_i is a matrix of heights. When $i = 1$, $z_0 \equiv u$, the first layer is the input layer. where f_h is the element-wise activation function for the hidden layers

$$f_i(z) = \begin{bmatrix} \sigma(v_{i1}) & \sigma(v_{i2}) & \ldots & \end{bmatrix}, \tag{17}$$

where $\sigma(v_{ij})$ is the sigmoidal logistic function and v_{ij} the activation signal for neuron j in layer i. For the output layer

$$y \equiv z_{h+1} = f_y(w_h z_h + b_{h+1}), \tag{18}$$

again f_y is the activation function for the output layer, usually linear.

The FNN goal is to approximate some function f^*, that maps $u \mapsto y$, $y = f^*(u)$, according to [1]. An FNN defines a mapping $y_{NN} = f_{NN}(u; W)$ using the values of the weights in W that results in the 'best' function approximation ($y_{NN}(u) \approx y(u)$). The weights W are obtained using the Backpropagation algorithm that solves numerically the following optimal problem, [3]

$$\begin{aligned} &\min_W J(y_0, u_0, W_0) = \frac{1}{2}(y - y_0)^T(y - y_0) \\ &\quad s.t. \\ &\quad y(u_0; W) = f_y \circ f_h \circ \ldots \circ f_1(u_0) \end{aligned} \tag{19}$$

where y_0, u_0 are training vectors presented to the algorithm in a batch. These batches of data are typically arranged in sequences containing the corresponding maps from the inputs to the outputs. In each cycle of training, an epoch, all data is used. Training is done in several hundreds of epochs, allowing the weights to converge.

A deep feedforward neural network, with two hidden layers of 10 units, 8 inputs (states $\mathbf{x}$), and 3 outputs (commands $\mathbf{u}$), was trained using the oracle data for landing described in the previous section as a supervisor. The oracle produced 200 different trajectories from a nominal one, just by randomizing uniformly the initial conditions up to $\pm 10\%$ of their nominal values, collect in pairs ($\mathbf{x}_\mathcal{O}$,$\mathbf{u}_\mathcal{O}$).

Fig. 4 to Fig. 6 show the output results of the DNN for a single trajectory provided from the oracle developed and described in the previous section. The rest of the figures (Fig. 7 to Fig. 10) depicted the Monte-Carlo test for the trained DNN.

Looking into the results, it can be concluded that the oracle and the DNN training is possible using the procedure proposed in the previous sections.

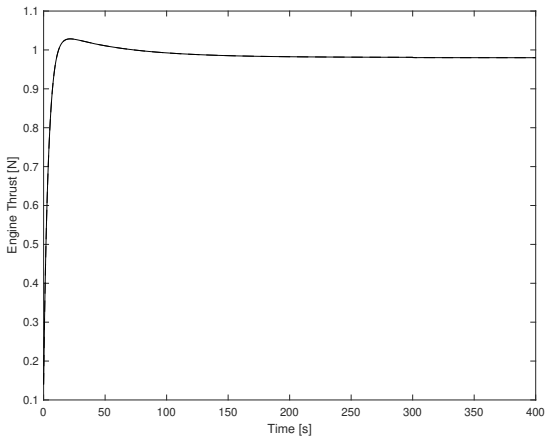

Fig. 4. Thruster $F_T/1 \times 10^5$ $[N]$.

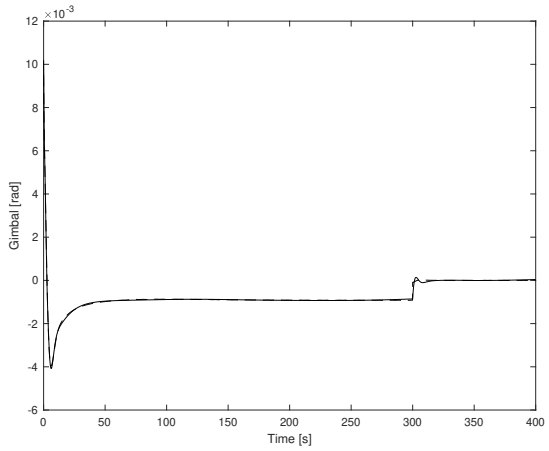

Fig. 5. Pitch angle ϕ $[Rad]$.

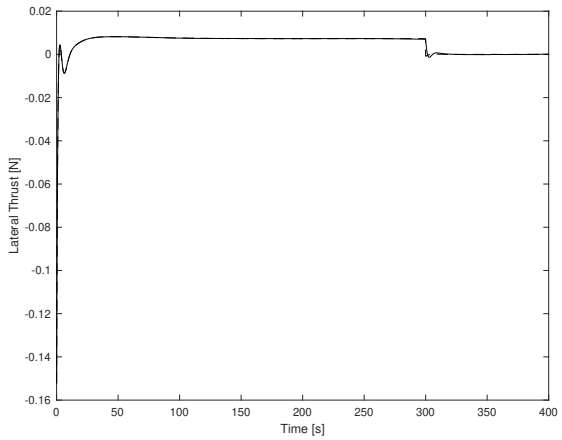

Fig. 6. Lateral actuators $F_s/1 \times 10^4$ $[N]$.

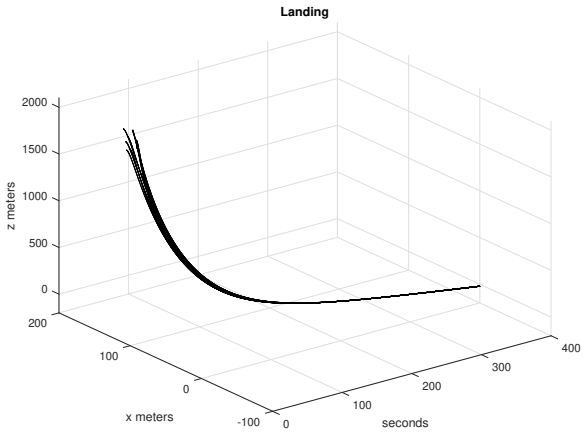

Fig. 7. Multiple landing trajectories provided by the oracle.

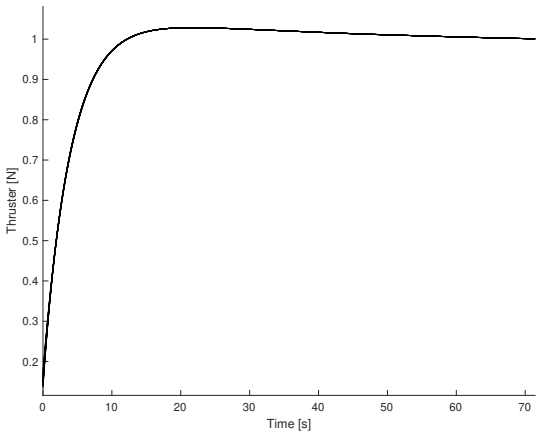

Fig. 8. Monte-Carlo for the Engine Thruster [N].

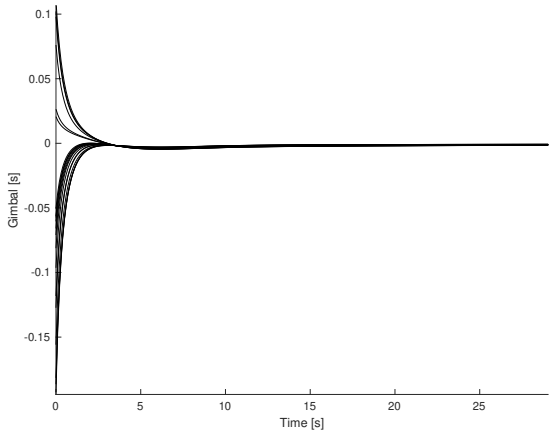

Fig. 9. Monte-Carlo for the Gimbal angle [Rad].

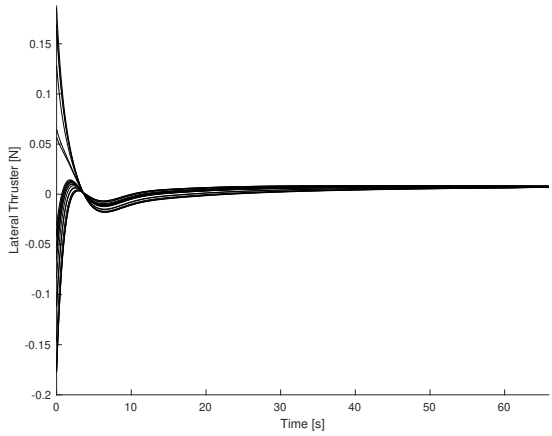

Fig. 10. Monte-Carlo for the Lateral Thruster [N].

V. Conclusions

In this work landing trajectories for a reusable launch vehicle (RLV) were computed, and then an artificial neural network was trained to perform the task. Landing is understood to be the last part of reentry, when the RLV is at a few kilometers from the ground in an off-set position in relation to the landing site, traveling with a know descending velocity and showing a small inclination angle in relation to the site vertical. The problem is solved for a RLV with lateral thrusters in addition to the gimbaled thrust engine. Moreover, the ANN was trained with validated solutions for different flight conditions and control parameters, the so-called oracle for landing. Validation and verification is done by Monte-Carlo techniques. Finally two important comments can be done. This approach shows that the lateral thrusters can be very useful to control the vehicle up, at least during the last part of the landing. The creation of an oracle for RLV landing is not limited to a unique way of finding valid trajectories, any kind of solution can be included, the training do not depend in how these trajectories were obtained.

References

[1] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[2] Reuben Ferrante. A robust control approach for rocket landing. Master's thesis, School of Informatics, University of Edinburgh, 2017.

[3] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[4] Michael Hertneck, Johannes Koehler, Sebastian Trimpe, and Frank Allgower. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, PP:1–1, 06 2018.

[5] Alberto Isidori. *Nonlinear Control Systems*. Communications and Control Engineering. Springer, London, 1995.

[6] Scott A. Nivison and Pramod P. Khargonekar. Development of a robust deep recurrent neural network controller for flight applications. In *2017 American Control Conference (ACC)*, pages 5336–5342, 2017.

[7] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(8):1399–1413, 2020.

[8] A. Tewari. *Advanced Control of Aircraft, Spacecraft and Rockets*. Aerospace Series. Wiley, 2011.

[9] Hao Wang and Tarek A. Elgohary. A simple and accurate apollo-trained neural network controller for mars atmospheric entry. *International Journal of Aerospace Engineering*, 2020:15, 9 2020.