Online guidance updates using neural networks

Cristian Filici[a], Ricardo S. Sánchez Peña[b,*,1]

[a]Comisión Nacional de Actividades Espaciales and Universidad de Buenos Aires, Paseo Colón 751, (1063), Buenos Aires, Argentina
[b]CONICET & Instituto Tecnológico de Buenos Aires (ITBA), Av. Eduardo Madero 399, (C1106ACD) Buenos Aires, Argentina

Keywords:
Trajectory optimization
Online guidance
Neural network
Training

A B S T R A C T

The aim of this article is to present a method for the online guidance update for a launcher ascent trajectory that is based on the utilization of a neural network approximator. Generation of training patterns and selection of the input and output spaces of the neural network are presented, and implementation issues are discussed. The method is illustrated by a 2-dimensional launcher simulation.

1. Introduction

The problem of computing optimal guidance commands to shape the trajectory of a launcher vehicle during its flight has been a subject of study for decades. Early methods relied on simplified dynamic equations in order to be able to determine analytic expressions, *Laws*, of the guidance commands. One of these laws is the bilinear tangent law [1], which can be used to compute the optimal pitch angle by an expression of the form $\tan(c_1 t + c_2)/(c_3 t + c_4)$. This law has been used successfully in launcher vehicles and is near optimal during the non-atmospheric part of the flight.

Modern methods try to avoid simplifying the dynamics at the expense of more complicated expressions and the need of an iterative process in order to achieve convergence to the desired optimal guidance commands [2–4]. The current approaches can be classified as indirect methods or direct ones [5]. Indirect approaches [4] are based on the theory of calculus of variations. They are usually fast in convergence, but it is sometimes difficult to provide an initial value for the iteration. On the other hand, direct methods [6] are based on the transcription of the infinite dimensional optimal problem to a non-linear programming one; however this transcription is made at the expense of a discretization of the original problem.

Recently, the use of neural network approximators has been proposed as an alternative to the solution of this problem. The advantage of this approach is that the neural approximator can be trained before the flight using the dynamic equations without any simplifications and with as many test scenarios as desired [7].

When considering an actual ascent flight of a launcher vehicle, an optimal nominal trajectory and its corresponding guidance parameters are usually computed off-line before takeoff. However, perturbations on the trajectory such as winds, unmodelled dynamics, for instance in aerodynamic forces, or non-nominal behavior of the vehicle, for instance in the thrust profile, make the introduction of changes in nominal guidance necessary during the flight [6].

This online re-optimization shall be performed periodically during the flight and in general it cannot rely on the same methods and dynamic modelling as the ones used for the off-line nominal optimization. This is because their

* Corresponding author.
E-mail addresses: cristian@conae.gov.ar (C. Filici), rsanchez@itba.edu.ar (R.S. Sánchez Peña).
[1] Full Member IAA.

complexity usually requires human analysis in the loop, and the fact that onboard real-time computation is significantly more limited than the one used at ground.

That is why other forms of solving these online optimization problems have been devised. As discussed before, the solution can be obtained by introducing simplifications on the dynamics so as to generate analytic expressions for an approximate solution of the problem [8].

Though successful in practice, this method can handle only small deviations from the nominal trajectory. This lack of robustness may have as a consequence delays in the launch date, for instance, due to adverse weather conditions that cannot be handled by the proposed guidance [9]. Another alternative, which is more robust in the sense discussed in the previous paragraph, is the introduction of a neural network approximator as an aid in the computation of the online guidance parameters. The neural approximator has the advantage that it can be trained off-line using a wide variety of flight scenarios, without any need of introducing simplifications on the dynamic model. Once carefully trained, the neural network parameters can be loaded in the vehicle's guidance computer and used in the computation of the guidance online, as network interrogation is a very fast process [7]. The latter is very useful due to the fact that it should be updated by real-time measurements.

The aim of this paper is to present a method that combines the characteristics of the classical guidance with a neural network approximator, trying to exploit the best of the two approaches. Though simple in its conception, the method involves several steps which will be described in this paper. The first sections are devoted to present a general description which can be adapted to several types of problems. Afterwards, an application to a specific example of guidance command computation for a 2-D trajectory for an elliptical transfer orbit applied to a launcher's last stage is presented.

Section 2 gives some background on the optimal control problem that has to be solved in order to compute the guidance commands and the neural approximator. Section 3 discusses the definition of the input and output spaces of the neural approximator. Section 4 describes the algorithm used to generate the patterns for the training of the neural network. Section 5 is devoted to describe the online guidance update process while Section 6 illustrates the method by means of a launcher example. Finally, summary and conclusions are presented in Section 7.

2. Background

2.1. Optimal control problem

We start by recalling the classical optimal control problem in the interval $[t_0, t_f]$, adopting the notation of Kraft [10]. Controls $\mathbf{g}(t) : \mathbb{R} \to \mathbb{R}^q$ and design parameters $\pi \in \mathbb{R}^{q_\pi}$ are sought in order to minimize the functional relation

$$\min_{u,\pi} \psi_0[t_0, \mathbf{z}(t_0), t_f, \mathbf{z}(t_f)] + \int_{t_0}^{t_f} \phi_0[t, \mathbf{z}(t), \mathbf{g}(t), \pi]\, \mathrm{d}t \tag{1}$$

subject to a set of dynamic, boundary and algebraic constraints given by

$$0 = \dot{\mathbf{z}} - \mathbf{f}[t, \mathbf{z}(t), \mathbf{g}(t), \pi] \tag{2}$$

$$0 = \Psi[t_0, \mathbf{z}(t_0), \tau, \mathbf{z}(\tau), t_f, \mathbf{z}(t_f)] \tag{3}$$

$$0 \leq \Xi[t, \mathbf{z}(t), \mathbf{g}(t), \pi] \tag{4}$$

where $\tau \in [t_0, t_f]$, and the dynamics is represented by function $\mathbf{f} : \mathbb{R}^{p+q+\pi_q+1} \to \mathbb{R}^p$. The boundary constraints are defined by $\Psi : \mathbb{R}^{3\times(p+1)} \to \mathbb{R}^r$, and the algebraic restrictions are given by $\Xi : \mathbb{R}^{p+q+\pi_q+1} \to \mathbb{R}^s$. This problem is solved assuming nominal dynamic conditions in order to obtain the nominal states $\mathbf{z}_N(t)$ and nominal guidance controls $\mathbf{g}_N(t)$.

2.2. Neural approximator

Several neural networks types and topologies could be used in this method. In this study, a multiple linear output feedforward network is proposed as the neural approximator, $\mathscr{N} : \mathbb{R}^{s+1} \to \mathbb{R}^q$, which can be written as

$$\mathscr{N}(t, \mathbf{D}) = \sum_{i=1}^{H} \mathbf{v}_i \sigma(\zeta_i) - \mathbf{s} \tag{5}$$

with $\zeta_i = \sum_{k=1}^{s} \omega_i^k \mathbf{D}_k + \omega_i t - h_i$ and activation function σ. The output weights are given by vectors $\mathbf{v}_i$, the output threshold by $\mathbf{s}$, and ω_i and ω_i^k represent the hidden weights while h_i is used for the hidden thresholds. Function $\mathbf{D}$ measures the difference between the actual and the nominal final conditions. The algorithm proposed for network training is the Levenberg Marquardt one [11].

3. Neural network *I*/*O* spaces

In this section, the definition of the input and output spaces of the neural network is presented. The main task that has to be performed in order to be able to use a neural network approximator to provide guidance parameter updates during the vehicle's flight is to define the input space of the approximator. This is the case because it is clear that the output of the neural network will be the guidance parameters or some function of them, for instance the difference with respect to the previously computed nominal ones.

The input parameters of the network should reflect in one form or another the reality that the vehicle is facing at the instant the online guidance computation is taking place. For instance, the current state vector and also possibly the current perturbations on the dynamic forces could be chosen as the input parameters of the network.

As the perturbations on the dynamic forces may not always be available during the flight, the choice of the input space is based on state vector information. One option is to choose the current state, or its difference with respect to the nominal one, as the network input parameters. This approach defines an input space which is based only on local real-time information.

Another alternative is to use some function of the actual flight conditions rather than directly adopting the current state. In this work we have chosen a function $\mathbf{D}$, which measures the deviations of the current trajectory final conditions

from the nominal ones. In this way, though sacrificing the direct use of current information, the input space of the neural approximator is more uniform than the one based on the direct use of the current state. This is due to the fact that it is not based only on local information, but it also reflects the ability of the vehicle to reach the desired final conditions under non-nominal flight. In other words, function **D** is obtained computing final conditions at t_f by means of using actual real-time information.

At a given flight time t_1, function **D** is computed by propagating the current state variables, sensed by the vehicle's navigation system, up to the nominal final time. The final conditions reached by this propagation will most probably be different from the desired final nominal conditions, reflecting the actual perturbations that have acted on the vehicle up to time t_1, thus the nominal final conditions can be subtracted from the propagated ones in order to complete the definition of function **D**.

The neural approximator $\mathscr{N} : \mathbb{R}^{s+1} \to \mathbb{R}^q$ shall be a function that maps the ability of the vehicle to reach final conditions **D**, onto a correction on the nominal guidance parameters Δ_G, as follows:

$$\Delta_G = \mathscr{N}(t, \mathbf{D}) \tag{6}$$

where the number of guidance parameters is q. In other words, for each time during the flight, and each deviation from the final conditions, we are seeking for a function that furnishes the related correction to the nominal guidance in order to achieve the desired final conditions. For each particular problem, the variables taking part in Δ_G and in **D** have to be selected appropriately.

4. Training set generation

In this section the generation of the patterns for the training of the neural approximator is discussed. Once the input/output spaces of the neural approximator have been defined, we shall describe in more detail the algorithm proposed for the generation of the patterns that will be used for neural network training.

We start by discussing the generation of a single pattern. Let us assume that a nominal trajectory, composed by nominal states $\mathbf{z}_N$ and nominal guidance $\mathbf{g}_N$, has been obtained by solving the optimal control problem of Section 2.1 under nominal conditions. Then, as before, take t_1 as a fixed time in the flight, and $\mathbf{z}_N(t_1)$ as the corresponding nominal state vector. Let us consider a perturbation $\delta\mathbf{z}(t_1)$ on the state vector at time t_1 and call the perturbed state $\tilde{\mathbf{z}}(t_1) = \mathbf{z}_N(t_1) + \delta\mathbf{z}(t_1)$.

In order to generate a single pattern, two procedures have to be considered. One of them will furnish the value of the input part of the pattern, while the other one will provide the corresponding output.

On the one hand regarding the input, the perturbed state $\tilde{\mathbf{z}}(t_1)$ is propagated to the final time and the achieved final conditions are compared against the nominal, which defines **D**.

Formalizing the previous paragraph, we define the final condition vector as $\mathbf{c}_f \in \mathbb{R}^{r_f}$. The nominal final condition, $\mathbf{c}_f^N$, is a fixed vector which is obtained from the nominal trajectory. The propagated final condition vector, $\mathbf{c}_f^P$, is dependent both on the start time of the propagation and on the perturbed state vector at that time, thus $\mathbf{c}_f^P[t, \tilde{\mathbf{z}}(t)] : \mathbb{R}^{p+1} \to \mathbb{R}^{r_f}$. Hence, **D** is computed by subtracting the two final condition vectors, $\mathbf{D}[t_1, \tilde{\mathbf{z}}(t_1)] = \mathbf{c}_f^P[t_1, \tilde{\mathbf{z}}(t_1)] - \mathbf{c}_f^N$.

Let us call $\mathbf{z}_f^P[t, \mathbf{z}(t)] : \mathbb{R}^{p+1} \to \mathbb{R}^p$ the final state resulting from the propagation of the state $(t, \mathbf{z}(t))$ under the nominal control $\mathbf{g}_N$, and $\mathbf{c} : \mathbb{R}^p \to \mathbb{R}^{r_f}$ the function that maps the final state vector into final conditions. With this notation, we can write function **D** as $\mathbf{D}[t_1, \tilde{\mathbf{z}}(t_1)] = \mathbf{c}(\mathbf{z}_f^P[t_1, \tilde{\mathbf{z}}(t_1)]) - \mathbf{c}(\mathbf{z}_f^P[t_1, \mathbf{z}_N(t_1)])$.

A bound on the magnitude of function D can be derived assuming that function $\mathbf{f}(t, \bullet, \mathbf{g}_N)$ is Lipschitz with constant L_f uniformly in t and $\mathbf{g}$ inside an open convex set containing the trajectories $\mathbf{z}_N$ and $\tilde{\mathbf{z}}$; and that a Lipschitz constant L_c exists on function **c** on a sufficiently large neighborhood of $\mathbf{c}_f^N$, the nominal final condition. Under the previous assumptions, and following a derivation similar to that of Khalil [12], p. 79, we have that

$$\tilde{\mathbf{z}}(t) = \tilde{\mathbf{z}}(t_1) + \int_{t_1}^{t} f(s, \tilde{\mathbf{z}}, \mathbf{g}_N)\, ds \tag{7}$$

$$\mathbf{z}_N(t) = \mathbf{z}_N(t_1) + \int_{t_1}^{t} f(s, \mathbf{z}_N, \mathbf{g}_N)\, ds \tag{8}$$

these identities, together with the Lipschitz condition, imply

$$\|\tilde{\mathbf{z}}(t) - \mathbf{z}_N(t)\| \leq \|\tilde{\mathbf{z}}(t_1) - \mathbf{z}_N(t_1)\| + \int_{t_1}^{t} L_f \|\tilde{\mathbf{z}} - \mathbf{z}_N\|\, ds \tag{9}$$

Applying the Gronwall lemma, integrating, and evaluating at $t = t_f$, we have that

$$\|\mathbf{z}_f^P[t_1, \tilde{\mathbf{z}}(t_1)] - \mathbf{z}_f^P[t_1, \mathbf{z}_N(t_1)]\| \leq \|\delta\mathbf{z}(t_1)\| \exp[L_f(t_f - t_1)] \tag{10}$$

Now, due to the Lipschitz assumption on **c** we can write

$$\|\mathbf{D}[t_1, \tilde{\mathbf{z}}(t_1)]\| \leq L_c \|\mathbf{z}_f^P[t_1, \tilde{\mathbf{z}}(t_1)] - \mathbf{z}_f^P[t_1, \mathbf{z}_N(t_1)]\| \tag{11}$$

$$\leq L_c \|\delta\mathbf{z}(t_1)\| \exp[L_f(t_f - t_1)] \tag{12}$$

which gives the bound on **D**.

Resuming the discussion on the generation of a single pattern, and regarding the output, a new optimal control problem is set up with initial time t_1, considering $\tilde{\mathbf{z}}(t_1)$ as initial condition, and with control function $\mathbf{g}_N(t) + \Delta_G(t)$ which allows the introduction of an optimal correction $\Delta_G(t)$, to the nominal guidance that takes into account the perturbed initial condition. Although this new optimal guidance will be a function of time inside the interval $[t_1, t_f]$, we shall focus only on time t_1, and consider $\Delta_G(t_1)$ for the computations that follow.

Thus, a single pattern is defined by considering the relationship

$$\{t_1, \mathbf{D}[t_1, \tilde{\mathbf{z}}(t_1)]\} \longleftrightarrow \Delta_G(t_1)$$

Keeping time t_1 unchanged, different perturbations can be introduced to the nominal state vector, so that a set of patterns is defined for time t_1.

Finally selecting times different from t_1 and repeating the above procedure, the construction of the training set is

completed. In order to summarize the above discussion, the complete algorithm is presented:

1. For a given time t_i $(i = 1, \ldots, N)$.
   (a) Perturb the nominal state vector by $\delta \mathbf{z}_j$ $(j = 1, \ldots, M)$.
      (i) Propagate the perturbed state to final time and compute $\mathbf{D}$.
      (ii) Considering the perturbed state, re-optimize guidance parameters to get Δ_G.
      (iii) Store pattern $\{t, \mathbf{D}[t, \tilde{\mathbf{z}}(t)]\} \longleftrightarrow \Delta_G(t)$.
   (b) $j = j + 1$ and GOTO (1a)

2. $i = i + 1$ and GOTO (1)

5. Online guidance update process

In this section we describe how the neural approximator is used in the computation of the online guidance updates. These will rely on the usage of the neural approximator and will enhance the nominal guidance that has been computed off-line.

Nominal guidance is computed off-line before the flight by means of an optimal control algorithm and usually requires engineering analysis in order to achieve appropriate convergence conditions. This nominal solution, after computing, is stored and used during the flight as a reference guidance which is enhanced by the introduction of the neural approximator that will take into account the actual perturbations on the trajectory in real-time.

In essence, during the flight, the guidance computer shall issue guidance commands to the vehicle's attitude control system. If no perturbations were encountered, these guidance commands shall be equal to the nominal ones. However, perturbations are expected and thus the nominal guidance will be augmented by the result of the evaluation of the neural approximator.

At a given time t_1 during the flight, the current state vector $\mathbf{z}(t_1)$ provided by the navigation system which includes the actual perturbations, is propagated until nominal final time in order to assess the difference between nominal and propagated final conditions. In other words, at time t_1 function $\mathbf{D}[t_1, \mathbf{z}(t_1)]$ is evaluated. Once $\mathbf{D}$ is obtained, it is possible to interrogate the previously trained neural approximator in order to compute the correction to the nominal guidance, ΔG.

Thus, at a given time t_1 during the flight, the guidance command $\mathbf{g}$ shall have the form

$$\mathbf{g}(t_1) = \mathbf{g}_N(t_1) + \mathscr{N}\{t_1, \mathbf{D}[t_1, \mathbf{z}(t_1)]\} \tag{13}$$

This can be repeated for any subsequent time, providing thus a means for reshaping the trajectory until final time, where the current final condition is expected to be close to the nominal one.

It is possible to outline a derivation of a bound for the difference between the current and the nominal final conditions. Besides the assumptions made to derive the bound on D in Section 4, let us assume that function $\mathbf{f}(t, \mathbf{z}, \bullet)$ is Lipschitz with constant L_g uniformly in t and $\mathbf{z}$, inside an open ball centered in the nominal control $\mathbf{g}_N$, then following [12]

$$\mathbf{z}(t) = \mathbf{z}(t_1) + \int_{t_1}^{t} f(s, \mathbf{z}, \mathbf{g}_N + \mathscr{N})\, ds \tag{14}$$

$$\mathbf{z}_N(t) = \mathbf{z}_N(t_1) + \int_{t_1}^{t} f(s, \mathbf{z}_N, \mathbf{g}_N)\, ds \tag{15}$$

which, by adding and subtracting $f(s, \mathbf{z}, \mathbf{g}_N)$, and noting that $\mathbf{z}(t_1) = \tilde{\mathbf{z}}(t_1)$, imply

$$\begin{aligned} \|\mathbf{z}(t) - \mathbf{z}_N(t)\| \leq \|\tilde{\mathbf{z}}(t_1) - \mathbf{z}_N(t_1)\| + \int_{t_1}^{t} L_g \|\mathscr{N}\|\, ds \\ + \int_{t_1}^{t} L_f \|\mathbf{z} - \mathbf{z}_N\|\, ds \end{aligned} \tag{16}$$

The approximator $\mathscr{N}(t, \mathbf{D})$ is continuous by definition, and $\mathbf{D}$ is bounded due to expression (12), so $\mathscr{N}$ is evaluated on a compact set, and consequently a constant Γ exists such that $\|\mathscr{N}\| \leq \Gamma$. Then, by means of the Gronwall lemma and noting that $t \leq t_f$, we have that

$$\|\mathbf{z}(t) - \mathbf{z}_N(t)\| \leq [\|\delta \mathbf{z}(t_1)\| + L_g(t_f - t_1)\Gamma] \exp[L_f(t - t_1)] \tag{17}$$

Now, evaluating the previous inequality at $t = t_f$ and due to the Lipschitz assumption on $\mathbf{c}$ we can write

$$\begin{aligned} &\|\mathbf{c}(\mathbf{z}(t_f)) - \mathbf{c}(\mathbf{z}_N(t_f))\| \\ &\quad \leq L_c\{[\|\delta \mathbf{z}_N(t_1)\| + L_g(t_f - t_1)\Gamma] \exp[L_f(t_f - t_1)]\} \end{aligned} \tag{18}$$

which gives the bound on the error in the final conditions with respect to the nominal ones.

It shall be noted that the proposed method allows flexibility in the final time, thus, the current trajectory may end at a time different from t_f, say $\hat{t}_f$. If we assume that function $f(t, \mathbf{z}(t), \mathbf{g}_N(t) + \mathscr{N}(t))$ is continuous in $[t_f, \hat{t}_f]$, the previous bound can be extended as

$$\|\mathbf{c}(\mathbf{z}(\hat{t}_f)) - \mathbf{c}(\mathbf{z}_N(t_f))\| \leq L_c(\|\mathbf{z}(\hat{t}_f)) - \mathbf{z}(t_f)\| + \|\mathbf{z}(t_f) - \mathbf{z}_N(t_f)\|) \tag{19}$$

From the continuity assumption on $\mathbf{f}$, there exists a constant ϕ such that, the first term of the right-hand side can be bounded by

$$\begin{aligned} \|(\mathbf{z}(\hat{t}_f)) - \mathbf{z}(t_f)\| &\leq \int_{[t_f, \hat{t}_f]} \|f(s, \mathbf{z}(s), \mathbf{g}_N(s) + \mathscr{N}(s))\|\, ds \\ &\leq \phi |\hat{t}_f - t_f| \end{aligned} \tag{20}$$

so that, from (19), the following bound is obtained:

$$\begin{aligned} \|\mathbf{c}(\mathbf{z}(\hat{t}_f)) - \mathbf{c}(\mathbf{z}_N(t_f))\| \leq L_c\{|\hat{t}_f - t_f|\phi + [\|\delta \mathbf{z}_N(t_1)\| + L_g(t_f - t_1)\Gamma] \\ \times \exp[L_f(t_f - t_1)]\} \end{aligned} \tag{21}$$

It is interesting to note that when t_1 approaches t_f, the previous bound depends mainly on the terms $|\hat{t}_f - t_f|\phi$ and $\|\delta \mathbf{z}_N(t_1)\|$. If the former could be disregarded (for instance, if $\hat{t}_f \approx t_f$), then the bound would depend on the constant L_c and on the difference between the current and the desired states.

6. Example

In this section, the proposed method is applied to a specific example. Practical issues that arise during the implementation of the method in order to be applied to this ascent trajectory problem are also discussed.

The example is a 2D ascent of a 3-staged launcher vehicle carrying a 2.4 T payload, to an elliptical transfer orbit with perigee and apogee heights of 170 km and 18 000 km respectively, adapted from [13]. The launch profile consists of four phases, being the first one a 28 s vertical ascent (Table 1). The durations of the first three phases are fixed, while the duration of the last phase is constrained to be less or equal than a predetermined value. The first stage of the vehicle is active during the first and second phases of the flight. The dynamic model studied considers the radius vector, velocity magnitude, flight path angle, longitude and mass as the states variables.

The whole set of controls is composed by the pitch, roll and yaw angles; however being the flight constrained to lay inside a vertical plane, the yaw angle can be assumed to be zero. Moreover, as vehicle longitudinal axis symmetry is assumed, also the roll angle can be disregarded. Thus the control angle present in this example is the pitch angle, which is modelled by the bilinear tangent guidance law.

The nominal trajectory together with the nominal guidance are computed by means of the direct shooting method TOMP [10], taking as objective function the payload mass. As a result of this optimization, the duration of the final stage is also obtained, with total flight duration of 847 s.

The computed nominal optimal trajectory is stored and comprises the final payload mass, the final stage duration and a function describing the parameters of the bilinear tangent guidance law of the pitch angle for the whole flight.

This example focuses on the last part of the flight. Perturbations on the states (radius and velocity magnitude) at given moments $t_i > 750$ s during the last stage are considered. These perturbations may be the result of unmodelled dynamics or vehicle non-nominal performance during the previous portions of the flight. Here, the neural approximator takes care of these discrepancies during the flight, as described next.

The generation of the training patterns is performed by randomly perturbing the state vector at times t_i with 5 s intervals. In this example we are taking the maximum perturbation on the radius as $\delta_r = 2$ km, and the one on the velocity magnitude as $\delta_v = 50$ m/s over nominal values of 6562 km and 8387 m/s, respectively. Note that if these perturbations are not corrected, they would produce a 1500 km error in apogee and a 2 km error in perigee, approximately. As the neural approximator is expected to reduce the effect of these perturbations when the vehicle approaches the end of its flight, it is convenient, in this region of the input space, to generate training patterns with smaller perturbations. In the example considered, a maximum $\delta_r = 1$ km has been chosen for times $t > 800$ s.

Table 1
Vehicle characteristics.

Stage number	First	Second	Third
Stage mass (T)	174.50	38.33	9.70
Stage fuel mass (T)	157.00	34.00	8.50
Drag effective area (m^2)	12.60	12.60	12.60
Vacuum thurst (kN)	2992.00	760.00	62.00
Nozzle area (m^2)	2.96	0.78	0.06
Isp (s)	268.36	296.52	434.67
Burn time (s) (nominal)	28 + 110.00	130.00	578.88
Active in phase	1–2	3	4

Then, as described in Section 4, to compute the input patterns to the guidance neural approximator, the perturbed state is propagated to final time in order to obtain **D**, by comparing final nominal vs. optimal parameters. These parameters have been chosen as the difference in apogee and perigee altitude at final time, thus, $\mathbf{D}(t_i) = (\delta_{apo}, \delta_{per})$.

Computation of the output patterns is performed by running new optimizations that take the perturbed state as initial condition so as to obtain optimal corrections Δ_G to the nominal guidance $\mathbf{g}_N$. Final condition dispersions are accounted for by means of imposing the following constraints on them: ± 10 km for the apogee and ± 3 km for the perigee. In this example, the control is the pitch angle, which is modelled by a three parameter bilinear tangent guidance law [1,14] mentioned in the introductory section. Also the time of engine cutoff, T_{CO} is introduced as a guidance parameter, which completes the definition of the neural approximator output patterns $\Delta_G = (c_1, c_2, c_3, \delta_{Tco})$. This provides a time dependent law with parameters that do not need to be updated each time the pitch angle is needed. It also has a very small number of parameters, so that the training process is not so time consuming. Finally it provides a quasi optimal model for pitch guidance in vacuum flight.

It is interesting to note that the objective function of these re-optimizations has been chosen to be different from the one used when computing the nominal trajectory (i.e. the payload mass). This is due to the fact that the re-optimization result is going to be used online during the flight (indirectly, through the neural approximator), and at that time the payload mass cannot be changed. That is why for the re-optimizations performed in order to obtain the corrections to the guidance parameters Δ_G, the payload mass is assumed to be fixed. Hence, a cost function different from the one used in the nominal trajectory computation shall be selected by taking into account that it is reasonable to ask for the new guidance parameters not to differ much from the nominal ones in order to avoid big loads on the attitude control system [7]. Thus, this cost function has been chosen as a function of the difference between the re-optimized and the nominal guidance, i.e. the norm of $\Delta_G(t)$.

Generation of the training patterns involves a great number of optimizations which in principle may take some time in order to be completed. However, as the guidance parameters are defined as the addition of a correction Δ_G to the nominal ones, a very good starting point for the iterative process of the optimization consists in taking this augmented guidance term Δ_G equal to zero. Thus, the generation of the patterns takes a fraction of the time that it would normally take under other conditions. A similar approach has been presented in [7].

After these processes are completed a total number of 6400 patterns, 400 for each t_i, is available for training. These patterns have three inputs and four outputs, and in order to ease the learning process, instead of training one neural

Table 2
Summary of the results.

Ex.	N	ΔT_N (s)	$SD(\delta_{apo})$ nominal (km)	$SD(\delta_{per})$ nominal (km)	$SD(\delta_{apo})$ neural (km)	$SD(\delta_{per})$ neural (km)
1a	200	10	6.7×10^2	1.2×10^0	1.2×10^0	9.0×10^{-2}
1b	200	5	6.9×10^2	1.1×10^0	1.6×10^0	8.3×10^{-2}
1c	200	2.5	6.8×10^2	1.2×10^0	2.3×10^0	9.6×10^{-2}
1d	200	1.25	6.5×10^2	1.2×10^0	2.1×10^0	9.0×10^{-2}
2a	200	5	1.4×10^3	2.5×10^0	6.1×10^0	3.4×10^{-1}
2b	200	5	6.5×10^2	1.3×10^0	1.8×10^0	2.2×10^{-2}
2c	200	5	6.8×10^2	7.5×10^0	1.3×10^2	4.8×10^0

approximator, the training set is partitioned in four, one set for each of the outputs. Thus, four neural approximators are trained to be used for guidance.

In this example, a simple network topology consisting of a feedforward network with one hidden layer is proposed. It is a well-known result from neural network theory that the approximation error depends both on the number of hidden neurons and on the number of patterns [15]. In this study the number of neurons in the hidden layer has been chosen to be 9. The training method used involves non-linear programming and the Levenberg Marquardt method was selected for this problem [16,17].

The trained networks are finally used to augment nominal guidance in ascent flight simulations. The initial state vector is propagated with a high fidelity numerical integration method [18] using nominal guidance until time $t_0 =$ 750 s. Here a random perturbation to the states is introduced and the neural guidance is activated.

The neural approximator is interrogated at discrete intervals ΔT_N during the flight. To do so, it is first necessary to propagate the dynamics until final time in order to obtain the deviations between desired and achieved final parameters, which are used as inputs to the neural approximator. With this input data, the approximator is used in order to obtain a correction to the nominal guidance that will aid in the achievement of the desired final parameters.

By means of this setting, several tests were performed which are summarized in Table 2. The first set of tests introduces perturbations on the radius and velocity magnitude at time $t_0 = 750$ s. These random perturbations are of the same maximum magnitude as those used to generate the training patterns. Four different tests are defined, each of them with a different value of the network interrogation step ΔT_N. All tests involve $N = 200$ flight ascent simulations. Relevant parameters of the tests are the standard deviation of the achieved final parameters, $SD(\delta_{apo}), SD(\delta_{per})$ for both the nominal and the neural augmented guidance schemes.

The second set of tests are aimed at verifying the neural approximator generalization property. Thus, ascent flight simulations are carried out introducing perturbations that were not accounted for during the training. Test 2a introduces perturbations on r and v at time t_0 but of twice the magnitude of those used to generate the training patterns. Test 2b introduces perturbations on r and v at time 700 s $< t_0$. Test 2c introduces perturbations on r, v and flight path angle γ (with maximum magnitude 1°) at time t_0.

Results of the set of tests 1 show that the neural approximator was successful in reducing orbit injection errors by one or two orders of magnitude. Typically it could drive an apogee height error greater than 600 km, resulting from the nominal guidance to values of the order of 1–4 km. The same can be stated on the perigee height error, which is driven from values greater than 1 km to magnitudes smaller than 100 m.

Also note that from the results of these tests, changes in the size of the network interrogation interval ΔT_N do not significantly impact the achieved neural guidance final parameters, as they remain inside the prescribed constraints for the final apogee and perigee values, i.e. ± 10 and ± 3 km respectively.

Results of tests 2a and 2b are very similar to those discussed previously. This shows that for these tests, the neural approximator generalization property provided a suitable guidance under non-trained conditions. Test 2c is somewhat different from the other tests, as it considers the introduction of a perturbation on a state variable (the flight path angle, γ) that was not used for training, which is why the performance of the approximator was lower than in the other tests; however the neural guidance was able to reduce the errors of the nominal guidance, driving the solution near the desired final parameter values.

Figs. 1–3 show the pitch resulting from the proposed guidance together with the nominal one for some of the test cases presented. Also, a plot of the resulting correction on the final time is presented for test 1a.

6.1. Practical computational issues

In this section some practical issues related to computation times for the procedures involved in the proposed method will be discussed.

The off-line computations are the most time consuming, as they require the generation of patterns and the network training. The neural approximator training requires the generation of a set of patterns, which are the result of several optimizations. The computational cost in terms of elapsed time of the application of this procedure to the example under study is of approximately 30 min. After the patterns are available, four neural approximators are trained to adjust these patterns, the average computing time for training being 75 min. This implies no practical problem, because it can be made off-line before the actual flight. The platform used here is a Pentium M, with a 1.7 GHz processor and Fortran 90 programming language.

As described before, the computation of the online guidance updates entails two procedures. One of them is the generation of the neural approximator input and the other is the evaluation of the approximator itself. The generation of the neural approximator input requires the forward propagation of the current state vector, which is assumed to be provided by the navigation system with negligible error. The numerical propagation is performed using the nominal dynamic model and with methods that ensure a solution of high accuracy [18], and though it is not a trivial computation, our simulations on the example studied show that the average computing time is less than 10 ms on the same

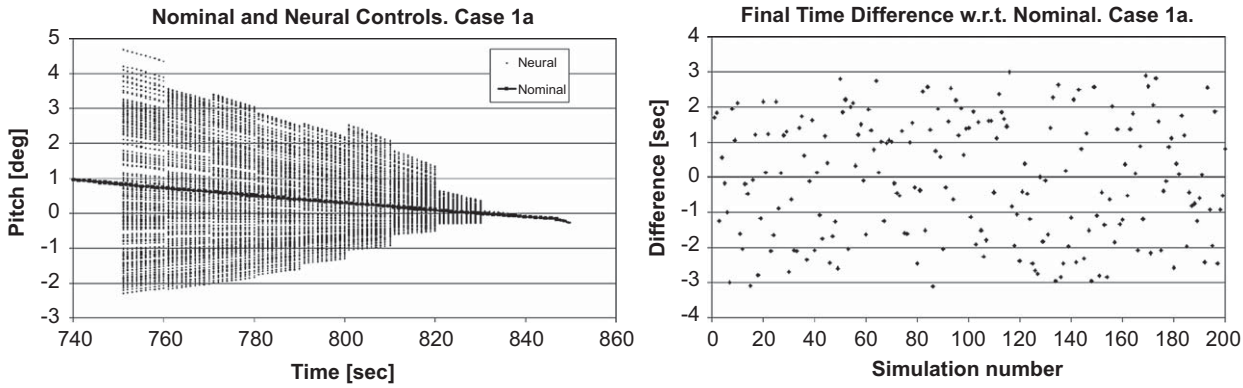

Fig. 1. Resulting vs. nominal pitch angle, and final time difference, case 1a.

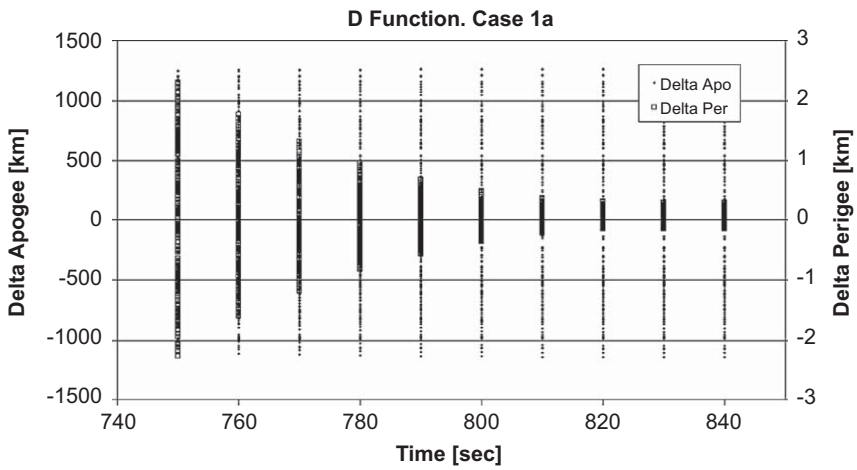

Fig. 2. Function D, case 1a.

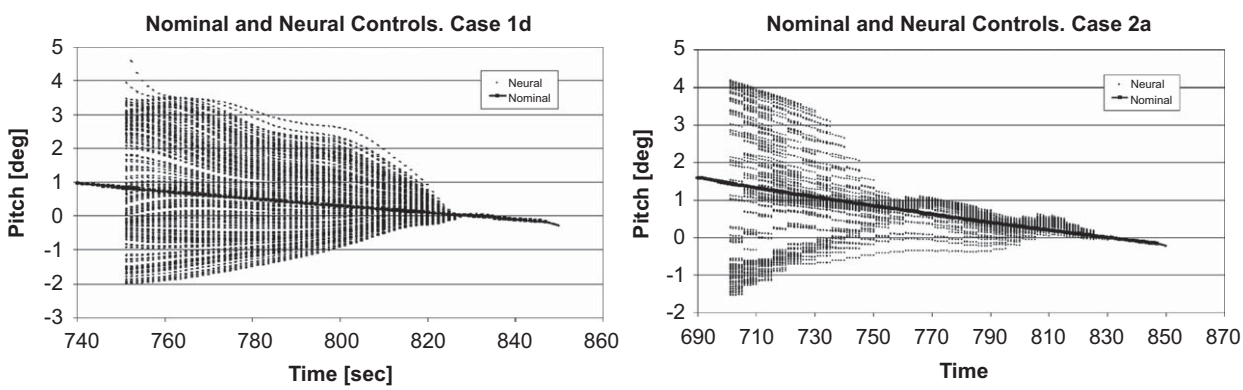

Fig. 3. Resulting vs. nominal pitch angle, cases 1d and 2b.

platform mentioned previously. This computation time is compatible with an online guidance scheme to be applied during the flight, which would probably be implemented on a faster PC board and with a real-time programming language.

On the other hand, it is widely known that the interrogation of the neural approximator is a very fast process [15]. This is confirmed by this example, where the average computing time for network evaluation was approximately 10 μs.

7. Conclusions

A method for online guidance updates using neural networks has been presented. The main idea is to obtain a guidance approximator to be used online during the ascent flight of a launcher vehicle, with a training process performed off-line, prior to the flight. In this way, all the time consuming training is compressed in the neural approximator, which is available during the flight and provides a very fast means for obtaining online guidance corrections.

Definition of the neural approximator input and output spaces is discussed. Neural approximator input patterns are taken as errors in the desired end of flight conditions, while the output patterns are defined as the corrections on nominal guidance arising from an optimization of the flight under non-nominal conditions.

These definitions allow for off-line generation of training patterns that are used in the learning process of the neural approximator. The construction of the training set is typically performed by numerical simulations, as it is not practical to obtain such large number of data from actual flights, due to the fact that an enormous amount of flights would be necessary.

The method is applied during the flight by propagating the current state vector to final time and obtaining the deviations from the desired end conditions. The latter are used in order to interrogate the neural approximator which provides as a result a correction to the nominal guidance.

An example is presented, based on an ascent launcher flight to an elliptical orbit. Details on the application of the method on this example are presented, together with the computing times involved.

The results of the example studied show that the neural approximator is successful in handling perturbations on the state vector during the flight so that the resulting trajectory final orbit conditions are more accurate than the ones that result from the application of the nominal guidance.

The method, as presented, provides a means to reshape the trajectory when a desired performance, e.g. final orbital conditions, is not achieved. This is done taking into account the nominal dynamics, thus a possible enhancement to the method would be to include the ability to reshape the trajectory under non-nominal dynamics.

The application of the method depends mainly on the fact that a suitable approximator can be defined and computed, that is why we consider that a parameter that has to be carefully chosen in order to apply the method is the dimension of the input space of the neural approximator. Choosing a very large input space dimension may make training not practical even though it is performed off-line.

Extension to other flight phases is possible in principle, due to the flexibility of the method which would allow its application with practically no changes. If for instance the whole flight sequence is considered, our approach would be to define a different neural approximator for each phase. In principle, all of the approximators would have the same I/O space structure, e.g. deviations from the final flight conditions as inputs, etc., and would be trained with optimal trajectories considering the same objective function as presented in this report. However, the method allows for flexibility on these aspects also.

Moreover, one possible way to treat the particular case of a flight phase where inflight constrains are relevant is to include some measure of the compliance with inflight constraints in the neural approximator input space. This may be suitable for inflight constraints whose assessment can be compressed in a few parameters. Otherwise, the input space dimension would be very large, and training would be hard to achieve. Another alternative is to note that the inflight constraints would be part of the full dynamic model that is used not only to compute the nominal trajectory, but also to build the training set of the neural approximator; thus the trained network will provide guidance updates that are at the same time feasible in terms of inflight constraints.

Acknowledgments

The first author appreciates the essential support of CONAE (Argentina), of DLR (Germany) where he performed some preliminary work in this area, and the helpful discussions with Prof. Klaus H. Well from the Institut für Flugmechanik und Flugregelung of the University of Stuttgart. The work of the second author has been financed by ICREA and CICYT Project no. DPI2005-04722 from the Ministry of Education and Science of Spain.

References

[1] A. Bryson, Y. Ho, Applied Optimal Control, Optimization, Estimation and Control, Hemisphere, USA, 1975.
[2] A. Calise, N. Melamed, S. Lee, Design and evaluation of a three-dimensional optimal ascent guidance algorithm, Journal of Guidance, Control and Dynamics 21 (6) (1998) 867–875.
[3] M.S.K. Leung, A.J. Calise, A hybrid approach to near-optimal launch vehicle guidance, AIAA Paper 1992-4304.
[4] P.F. Gath, A.J. Calise, Optimization of launch vehicle ascent trajectories with path constraints and coast arcs, AIAA paper 1999-4308.
[5] K. Well, Real-time on-board trajectory optimization for upcoming missions, in: 3rd International Workshop on Astrodynamics Tools and Techniques, October 2006.
[6] M.H. Graesslin, J. Telaar, U.M. Schoettle, Ascent and Reentry Guidance concepts based on NLP methods, 54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law, Bremen, Germany, Sep. 29-3, 2003.
[7] J.D. Schierman, J.R. Hull, D.G. Ward, Adaptive guidance with trajectory reshaping for reusable launch vehicles, AIAA Paper 2002-4458.
[8] C.D. Baker, Concepts of the iterative guidance law for Saturn launch vehicles, Technical memorandum, NASA, October 1965.
[9] R.G. Brusch, T.E. Reed, Real-time launch vehicle steering programme selection, Journal of the British Interplanetary Society 26 (1973) 279–290.
[10] D. Kraft, A software package for sequential quadratic programming, Technical Report 88-28, DFVLR-FB, 1988.
[11] B. Garbow, K. Hillstrom, J. More, Documentation for MINPACK subroutine LMDER, Technical Report, Argonne National Laboratory, March 1980.
[12] H. Khalil, Nonlinear Systems, second ed., Prentice-Hall, Upper Saddle River, NJ, USA, 1996.
[13] M. Noton, Spacecraft Navigation and Guidance, Advances in Industrial Control, Springer, London, 1998.
[14] A. Markl, An initial guess generator for launch and reentry vehicle trajectory optimization, Dr.-ing. Thesis, University of Stuttgart, Institute of Flight Mechanics and Control, June 2001.
[15] B. Krose, P. van der Smagt, An Introduction to Neural Networks, The University of Amsterdam, Amsterdam, 1996.

[16] T. Chen, D. Han, F. Au, L. Tham, Acceleration of Levenberg–Marquardt training of neural networks with variable decay rate, in: Proceedings of the International Joint Conference on Neural Networks, 2003, pp. 1873–1878.

[17] C. Filici, On a neural approximator to ODEs, IEEE Transactions on Neural Networks 19 (3) (2008) 539–543.

[18] E. Fehlberg, Classical fifth-, sixth-, seventh- and eighth-order Runge–Kutta, NASA Technical Report 287.