AAS 13-875

NEURAL-BASED TRAJECTORY SHAPING APPROACH FOR TERMINAL PLANETARY PINPOINT GUIDANCE

Roberto Furfaro,[*] Jules Simo[†], Brian Gaudet[‡], Daniel R. Wibben[§]

In this paper, we present an approach to pinpoint landing based on what we consider to be the next evolution of path shaping methodologies based on potential functions. Here, we employ Extreme Learning Machine (ELM) theories to devise a Single layer Forward Network (SLFN) that learns the relationship between current spacecraft position and the optimal velocity field required to shape the path to the surface in a fuel efficient fashion. ELM techniques enable fast and accurate training as well as better generalization. The network is trained using open-loop, fuel-efficient trajectories that are numerically generated using pseudo-spectral methods. After test and validation, the SLFN becomes a critical element in the linear guidance algorithm loop. More specifically, a Linear Quadratic Regulator (LQR) is employed to track the optimal velocity field which is naturally defined to be attractive to the landing target. The guidance approach is tested on a simulation environment to evaluate the performance of proposed algorithm. Monte Carlo simulations show that the algorithm achieve a low guidance residual error which is less than one meter in position and less than -0.9 m/sec in impact velocity.

INTRODUCTION

Future unconstrained, science-driven, robotic and human missions to Mars and other planetary bodies will require a high degree of landing accuracy. Over the past decade, the Mars pinpoint landing problem, i.e. the ability to guide one or more landers to a specified point on the Martian surface with accuracy less than 100 m, has been steadily gaining importance[1,2]. Indeed, the sustained robotic exploration experienced by the red planet, as well as the continued interest in conceiving and studying missions that may one day deliver humans to Mars contributed to generate the need for more precise delivery of cargo on to the planet's surface[3]. The science return of past robotic missions (e.g. Viking 1 and 2, Mars Pathfinder, Mars Exploration Rovers, Phoenix Lander) have been severely limited by the landing accuracy provided by the Entry, Descent, and Landing (EDL) system[3,4]. At the beginning of the Mars exploration era, safety was

[*] Assistant Professor, Department of Systems and Industrial Engineering, Department of Aerospace and Mechanical Engineering, University of Arizona, 1127 E. James E. Roger Way, Tucson, Arizona, 85721,USA

[†] Academic Visitor, Department of Mechanical and Aerospace Engineering, University of Strathclyde, Glasgow, G1 1XJ, United Kingdom.

[‡] Research Engineer, Department of Systems and Industrial Engineering, University of Arizona, 1127 E. James E. Roger Way, Tucson, Arizona, 85721, USA

[§] Graduate Student, Department of Systems and Industrial Engineering, University of Arizona, 1127 E. James E. Roger Way, Tucson, Arizona, 85721, USA probably the major concern that lander designers had to face. Because knowledge of Mars' surface characteristics (e.g. geomorphology, mineralogy, stratigraphy) was very limited, exploring any portion of the planet would have provided an initial understanding of Mars natural processes. Consequently, a standard practice was established where the spacecraft would be delivered within a relatively large area that 1) guaranteed safety against possible terrain hazards (e.g. large rocks, high surface slope) and 2) met a set of minimum science requirements. In such a case, mission success is ensured without the need of precisely specifying the exact landing location within the targeted area. The landing accuracy, usually characterized by the 3-sigma landing ellipse, was established to be 135 km for Mars Pathfinder[5], 120 km for Mars Phoenix Mission[6]and 35 km for both Mars Exploration Rovers[7]. In all of the above mentioned cases, the landing selection process was mostly constrained by finding relatively flat surfaces with the smallest rock-size and crater-size distribution[8]. While safe, such areas may not necessarily yield the highest science return. The on-going Mars Science Laboratory (MSL[9], launched November 2011) has taken important steps toward enabling less constrained science. The final landing site, which was selected out of a short list of four (4) possible locations[10], was constrained to be no larger than 10 km, i.e. the landing accuracy guaranteed by the newly designed system, the "Sky Crane"[11]. Such improvement in accuracy was achieved by actively controlling the capsule's bank angle during the hypersonic entry phase[11]. Despite the abovementioned improvements, future robotic missions will require additional pinpoint precision. For example, identification of targets and/or locales that have the potential to yield the highest geological and exobiological information[12] may require systems capable of landing in their close proximity in spite of rough terrain and other landing hazards. Human missions to Mars may require the delivery of cargo packages on specific locations followed by crewed spacecraft landing next to the assets already robotically delivered[13].

Powered descent algorithms[15,16] generally comprise of two major components, i.e. a) a targeting (guidance) algorithm and b) a trajectory-following, real-time guidance algorithm. The targeting algorithm, often referred to as "guidance" in the Mars landing community, is responsible for generating a reference trajectory (position, velocity and thrust profile) that explicitly defines the path driving the lander from the initial position (i.e., beginning of the powered descent phase) to the desired landing location (target point). Conversely, the trajectory-following algorithm is designed to close the loop on the desired trajectory ensuring that the spacecraft follows the planned path. Generating guidance algorithms for Mars landing has been the focus of many scientists and engineers[15,16,17,18]. Current practice for Mars and Lunar landing employs a guidance approach where the reference trajectory is generated on-board. The trajectory is computed as a time-dependent polynomial whose coefficients are determined by solving a Two-Point Boundary Value Problem (TPBVP). Originally devised to compute the reference trajectory used by the Lunar Exploration Module[25-27], the method is currently employed to generate a feasible reference trajectory comprising the three segments of the MSL powered descent phase[28]. A fifth-order (minimal) polynomial in time satisfies the boundary conditions for each of the three position components (downrange, crossrange and altitude). The required coefficients can be determined analytically as a function of the pre-determined (fixed) time-to-go. Recently, more research efforts have been devoted toward determining reference trajectories (and guidance commands) that are fuel-optimal, i.e. minimum-fuel trajectories that satisfy the desired boundary conditions and possibly additional constraints[15,17,19]. For such cases, analytical solutions are possible only for the energy-optimal landing problem with unconstrained thrust[18]. To the best of our knowledge, closed-form, analytical solutions for the full three-dimensional, minimum-fuel, soft landing problem with state and thrust constraints are not available. Indeed, such trajectories can be found only numerically using either direct or indirect methods. Solutions based on direct methods are generally obtained by converting the infinite-dimensional optimal control problem into a finite constrained Non-Linear Programming (NLP) problem[20]. Recently, Acikmeseet al.[19] devised a convex optimization approach where the minimum-fuel soft landing problem is cast as a Second Order Cone Programming (SOCP[21]). The authors showed that the appropriate choice of a slack variable can convexify the problem[22]. Consequently, the resulting optimal problem can be solved in polynomial time using interior-point method algorithms[23]. In such a case and for a prescribed accuracy, convergence is guaranteed to the global minimum within a finite number of iterations. The latter makes the method attractive for possible future on-board implementation. Moreover, the method has been extended to find solutions where optimal trajectories to the target do not exist, i.e. the guidance algorithm finds trajectories that are safe and closest to the desired target[24].

Despite the abovementioned advancements in trajectory-generating (guidance) algorithms for on-board determination of minimum-fuel flyable trajectories, such algorithms require a significant amount of real-time computation. Moreover, once the trajectory is generated, the guidance system has the additional burden of determining a closed-loop command that allows the vehicle to follow the desired path. In this paper, we develop a novel guidance approach based on a combination of trajectory shaping and neural network methodologies to devise an algorithm capable of generating an acceleration command that guides the spacecraft to the desired location on the planet's surface with zero velocity (soft landing). During the mid-90s, McInnes[29] developed a trajectory shaping approach for terminal lunar descent. The basic idea was to investigate potential function methods[30], to generate families of trajectories that are globally convergent to the target landing location. The method, which relies on Lyapunov's theorem for determining the stability of non-linear systems, hinges on the definition of a scalar function which satisfies the properties typically exhibited by a Lyapunov function. Under such conditions the landing target is attractive and a descent path following the potential function gradient ensure convergence and safe landing. Both velocity field and desired acceleration can be computed analytically. However, the feedback trajectories derived via the potential methods are generally not fuel-efficient. The idea behind the proposed guidance scheme is to approximate the potential field and more importantly its gradient by using machine learning algorithm to learn the relationship between position and desired velocity. A fuel-efficient vector field that is attractive to the target point, can be numerically computed using optimal control theory. The numerical data points can be employed to train a neural network which is therefore employed to within a linear guidance scheme to determine the desired velocity as function of the position. Among the possible plethora neural networks candidate for the job, we selected a class of networks called Extreme Learning Machines (ELM[31,33]). Advantages of using such techniques will be illustrated in the upcoming sections.

This paper is organized as follows. First the landing guidance problem is formulated. Subsequently, a discussion of the guidance development is presented including trajectory shaping and ELM theories, ELM design and training and a description of the EML-based linear guidance algorithm. Finally the proposed approach is tested in typical simulation scenarios which include a set of Monte Carlo simulation to evaluate the guidance residual error.

GUIDANCE PROBLEM FORMULATION

The planetary landing guidance problem that can be formulated as follows: given the current state of the spacecraft, determine a real-time acceleration and attitude command program that reaches the target point on the surface with zero velocity.

Guidance Model: 3-D Equations of Motion

The fundamental equations of motion of a spacecraft moving in the gravitational field of a planetary body can be described using Newton's law. Assuming a mass variant system and a flat planetary surface, the equations of motion can be written as:

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{1}$$

$$\dot{\boldsymbol{v}} = -\boldsymbol{g}(\boldsymbol{r}) + \frac{\boldsymbol{T}}{m_L} + \boldsymbol{p} \tag{2}$$

$$\dot{m}_L = -\frac{||\boldsymbol{T}||}{I_{sp} g_0} \tag{3}$$

Here, $\boldsymbol{r}$ and $\boldsymbol{v}$ are the position and velocity of the lander with respect to a coordinate system with origin on the planet's surface, $\boldsymbol{g}(\boldsymbol{r})$ is the gravity vector, $\boldsymbol{T}$ is the thrust vector,m_L is the mass of the spacecraft, I_{sp} is the specific impulse of the lander's propulsion system, g_0 is the reference gravity, and $\boldsymbol{p}$ is a vector that accounts for unmodeled forces (e.g. thrust misalignment, effect of higher order gravitational harmonics, atmospheric drag, etc.). If $\boldsymbol{r} = [x, y, z]^T$ and $\boldsymbol{v} = [v_x, v_y, v_z]^T$ the equations of motion can be written by components as:

$$\dot{x} = v_x \tag{4}$$

$$\dot{y} = v_y \tag{5}$$

$$\dot{z} = v_z \tag{6}$$

$$v_x = -g_x(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_x + p_x \tag{7}$$

$$v_y = -g_y(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_y + p_y \tag{8}$$

$$v_z = -g_z(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_z + p_z \tag{9}$$

The considered mathematical model is a 3-DOF model with variable mass. This model is employed to simulate spacecraft descent dynamics by the proposed guidance law.

LANDING GUIDANCE ALGORITHM DEVELOPMENT

Trajectory Shaping Guidance scheme

The Trajectory Shaping Guidance (TSG) algorithm is based on the idea that the spacecraft path defining the closed-loop descent trajectory toward a planetary surface can be shaped to follow a path that is attractive (i.e. ensure convergence toward the target) and safe (i.e. avoid obstacles)[29,30]. Although vehicle path and velocity-altitude profile can be defined a-priori, this will result in lack of flexibility and potential degradation in robustness. TSG relies on methodologies based on potential functions to establish a class of trajectories that are globally stable to the selected target location. As shown by McInnes[29,30], potential functions can be selected to enable landing the spacecraft in regions of complex terrain, i.e. shape the landing trajectories to reduce the probability of failure and/or avoid hazards.

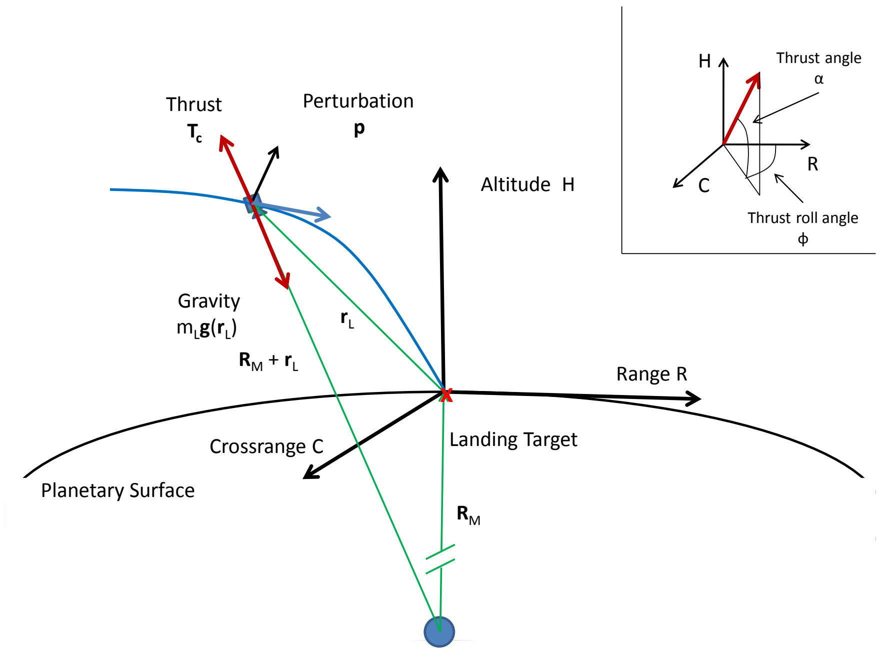

Figure 1.Guidance Reference Frame and Free-Body Force Diagram for a Planetary Lander during the Powered Descent to the Designated Target

Potential functions methods are rooted in the Lyapunov stability theory for non-linear systems. Such methods are applied by defining a scalar potential function that may represent the topography of the landing location. To ensure the definition of a class of descent paths that globally converges to the selected target location, one must develop a class of guidance algorithms that ensure that the derivative of the potential function (gradient) along the trajectory is always negative. Following the standard Lyapunov definition of a potential function, and defining $\boldsymbol{r}$ as the current spacecraft position and $\boldsymbol{r}_d$ the desired position on the planet's surface, we have:

$$\begin{gathered} \phi(\boldsymbol{r}_d) = 0 \\ \phi(\boldsymbol{r}) > 0, \quad \forall \boldsymbol{r} \neq \boldsymbol{r}_d \\ \dot{\phi}(\boldsymbol{r}) < 0, \ \forall \boldsymbol{r} \neq \boldsymbol{r}_d \\ \phi(\boldsymbol{r}) \to 0, \quad as \ \|\boldsymbol{r}\| \to \infty \end{gathered} \tag{10}$$

According to the second Lyapunov's theorem, it can be shown that the desired point $\boldsymbol{r}_d$ is globally attractive and all trajectories converge toward it. On this basis, McInnes[29] used the potential function method to generate families of descent path toward the lunar surface as follows:

1. The magnitude of the spacecraft velocity is shaped by specifying a pre-defined velocity profile as function of the altitude
2. The direction of the velocity vector, which in turn defines the trajectory path, is shaped using a potential function containing geometric information about the terrain surrounding the landing location. As a result, one can find an acceleration command that forces the vehicle to follow the negative of the gradient potential, which ensure global convergence to the desired location.

Importantly, for a given potential function $\phi(\boldsymbol{r})$ that satisfies Eq.(10), one can define a velocity field as follows:

$$\boldsymbol{v} = -v(h)\frac{\nabla\phi(\boldsymbol{r})}{\|\nabla\phi(\boldsymbol{r})\|} \tag{11}$$

Where $v(h)$ is a pre-defined velocity-altitude profile (velocity magnitude shape). If the potential function $\phi(\boldsymbol{r})$ has an analytical expression, one can easily find the acceleration command required to follow the prescribed path as defined by the potential function and the initial conditions. Indeed, the acceleration command is found as follows:

$$\boldsymbol{a}_c = \boldsymbol{\Lambda v} - \boldsymbol{g}(\boldsymbol{r}) \tag{12}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{\Lambda v},\ \boldsymbol{\Lambda} = \left\{\frac{\partial v}{\partial r}\right\}_{3x3} \tag{13}$$

Whereas this method has been previously studied, here we propose and evolution of the potential function-based methodology by a) numerically approximating the gradient of the potential function that yield a set of shaped path that are fuel-efficient and enforce specific trajectory constraints and b) define a guidance algorithm that tracks the optimal velocity field as function of the position.

Fuel-Efficient Velocity Field Computation via Extreme Learning Machines

The fundamental idea behind the guidance algorithm development is the ability to numerically approximate the gradient of a fuel-optimal potential function. Such function $\phi(\boldsymbol{r}, \boldsymbol{r}_d)$ depends on the actual position and desired (target) final position. Moreover, the gradient of the optimal potential function generates a velocity field $\boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$ that generates families of trajectories that are a) fuel-efficient and b) satisfy specific constraints (e.g. thrust direction and flight path angle) and c) drive the spacecraft to the desired point with a zero terminal velocity. Optimal control theory can be employed to appropriately define the optimal control problem and numerically determine fuel-efficient trajectories that satisfy specified boundary conditions as well as path and thrust constraints. However, an explicit, closed-form representation of either $\phi(\boldsymbol{r}, \boldsymbol{r}_d)$ and/or $\boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$ is not generally available. However, if sufficient samples representing the functional relationship between potential function and position, as well as velocity and position, are available, one can employ machine learning techniques to numerically approximate the desired function. Over the past two decades, Neural Networks (NN[32]) have emerged as powerful computational devise capable of approximate any piecewise continuous function to the desired degrees. Biologically inspired, NNs are comprised of computational units called neurons that have the ability to learn the relationship between any desired function (assuming that certain conditions are satisfied) from inputs-output example. Here the goal is employ a class of neural networks called Extreme Learning Machines (ELM[31,33]) to approximate the fuel-efficient velocity field.

Extreme Learning Machines: Theory

Computational intelligent techniques have been successfully used in learning functional relationships that are only described by a limited about of data points. Most of such techniques (e.g. NNs, Support Vector Machines (SVM)) are faced with many challenges including, slow learning speed, poor computational scalability as well as requirement of ad-hoc human intervention. Extreme Learning Machines have been recently established as an emergent technology that may overcome some of the abovementioned challenges providing better generalization, faster learning speed and minimum human intervention. ELMs work with "generalized" Single Layer Forward Networks (SLFN, Figure 2). SLFN are computationally designed to have a single hidden layer (which can be either Radial Basis Function (RBF) or other activation functions) couple to a linear output layer. The key point is that the hidden neurons need not to be tuned and their weights (training parameters) can be sampled from a random distribution. Theoretical studies[34] show that feed-forward networks with minimum output weights tend to achieve better generalization. EML tend to reach a) the minimum training error and b) the smallest norm of output weights with consequent improved generalization. Importantly, since the hidden nodes can be selected and fixed, the output weights can be determined via least-square methodologies.

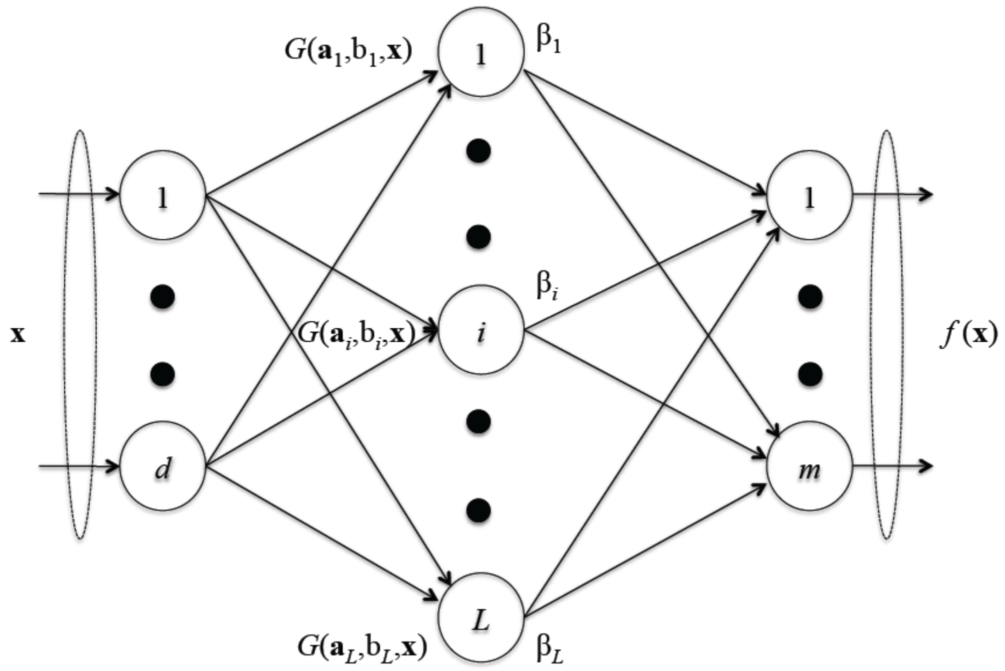
$G(\mathbf{a}_1, b_1, \mathbf{x})$
β_1
1
1
1
$\mathbf{x}$
$G(\mathbf{a}_i, b_i, \mathbf{x})$
i
β_i
$f(\mathbf{x})$
d
m
L
$G(\mathbf{a}_L, b_L, \mathbf{x})$
β_L

Figure 2. Typical architecture of a Single Layer Forward Network (SLFN) which is the most fundamental EML

Consider a SLFN with L hidden nodes (Figure 2). The output function can be represented as follows:

$$f_L(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g_{\boldsymbol{i}}(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) \quad with \quad \boldsymbol{x} \in \boldsymbol{R}^d, \boldsymbol{\beta}_i \in \boldsymbol{R}^m \tag{12}$$

For additive nodes with activation function g

$$G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(\boldsymbol{a}_i\boldsymbol{x} + b_i)\ with\ \ \boldsymbol{a}_i \in \boldsymbol{R}^m, b_{\boldsymbol{i}}\ \in R \tag{13}$$

For RBF nodes with activation function g

$$G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(b_i\|\boldsymbol{x} - \boldsymbol{a}_i\|) with\ \ \boldsymbol{a}_i \in \boldsymbol{R}^m, b_{\boldsymbol{i}}\ \in R^+ \tag{14}$$

Consider a training set comprising N distinct samples, i.e. $[\boldsymbol{x}_i, \boldsymbol{t}_i] \in \boldsymbol{R}^d \times \boldsymbol{R}^m$. The mathematical model describing SLFNs can be cast as follows:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = \ \boldsymbol{o}_j, \quad for\ j = 1, \dots\dots, N \tag{15}$$

Stating that SLFNs can approximate N samples with zero error is equivalent to state that there exist pairs $(\boldsymbol{a}_i, b_i)$ and $\boldsymbol{\beta}_i$ such that:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = \ \boldsymbol{t}_j, \quad for\ j = 1, \dots\dots, N \tag{16}$$

Compactly:

$$\boldsymbol{H\beta} = \boldsymbol{T} \tag{17}$$

Where the hidden layer output matrix $\boldsymbol{H}$ is formally written as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \dots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \dots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N \times L} \tag{18a}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L\times m} \quad \text{and} \quad \boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N\times m} \tag{18b}$$

Huang et al.[35] theoretically showed that SLFNs with randomly generated additive or RFB nodes can universally approximate any desired (target) function over a compact subset of $X \in \boldsymbol{R}^d$**.** Such results can be generalized to any piecewise continuous activation function in the hidden node[36]. The following theorem holds true:

Theorem [36]: Given any non-constant piecewise continuous function $g: R \to R$, if $span\{G(\boldsymbol{a}, b, \boldsymbol{x}): (\boldsymbol{a}, b) \in \boldsymbol{R}^d \times R\}$ is dense in L^2, any continuous target function f and any function sequence $\{g_L(\boldsymbol{x}) = G(\boldsymbol{a}_L, b_L, \boldsymbol{x})\}$ randomly generated based on any continuous sampling distribution, $\lim_{L\to\infty} \|f - f_L\|$ holds with probability one if the output weights $\boldsymbol{\beta}_i$are determined by ordinary least square to minimize $\left\|f(\boldsymbol{x}) - \sum_{i=1}^{L} \boldsymbol{\beta}_i g_{\boldsymbol{i}}(\boldsymbol{x})\right\|$.

The basic ELM can be constructed as follows. After selecting a sufficiently high number of hidden nodes (ELM architecture), the parameters $(\boldsymbol{a}_i, b_i)$ are randomly generated and remain fixed. Training occurs by simply finding a least-square solution $\boldsymbol{\beta}$ of the system $\boldsymbol{H\beta} = \boldsymbol{T}$**,** i.e. find $\widehat{\boldsymbol{\beta}}$ such that

$$\left\|\boldsymbol{H}\widehat{\boldsymbol{\beta}} - \boldsymbol{T}\right\| = \min_{\boldsymbol{\beta}} \|\boldsymbol{H\beta} - \boldsymbol{T}\| \tag{19}$$

ELM for Fuel-Efficient Velocity Field Approximation: Network Design and Training Set Generation via Pseudo-Spectral Methods

At the core of the proposed methodology is the ability to approximate a fuel-optimal velocity field that is representative of the gradient of a potential function. We employ ELM theory and we design and train a SLFN capable of approximating $\boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$. The development goes through the following stages: 1) training set generation, 2) SLFN architecture design, 3) Training phase and 4) testing and validation phase.

Training set generation: Samples from an optimal (continuous) vector field may be generated by numerically computing a set of fuel-efficient trajectories via proper application of the optimal control theory. The basic minimum-fuel problem for planetary landing can be defined as follows:

<u>*Minimum-Fuel Problem:*</u> Find the thrust program that minimizes the following cost function (negative of the lander final mass; equivalent to minimizing the amount of propellant during descent):

$$\max_{t_F, \mathbf{T}(\cdot)} m_L(t_F) = \min_{t_F, \mathbf{T}(\cdot)} \int_0^{t_F} \|\mathbf{T}\|\, dt \tag{20}$$

Subject to the following constraints (equations of motion):

$$\ddot{\boldsymbol{r}}_L = -\boldsymbol{g}_{\boldsymbol{L}} + \frac{\boldsymbol{T}}{m_L} \tag{21}$$

$$\frac{d}{dt} m_L = -\frac{\|\boldsymbol{T}\|}{I_{sp} g_0} \tag{22}$$

and the following boundary conditions and additional constraints:

$$0 < T_{min} < \|\boldsymbol{T}\| < T_{max} \tag{23}$$

$$\boldsymbol{r}_L(0) = \boldsymbol{r}_{L0}, \boldsymbol{v}_L(0) = \dot{\boldsymbol{r}}_L(0) = \boldsymbol{v}_{L0} \tag{24}$$

$$\boldsymbol{r}_L(t_F) = \boldsymbol{r}_{LF}, \boldsymbol{v}_L(t_F) = \dot{\boldsymbol{r}}_L(t_F) = \boldsymbol{v}_{LF} \tag{25}$$

$$m_L(0) = m_{Lwet} \tag{26}$$

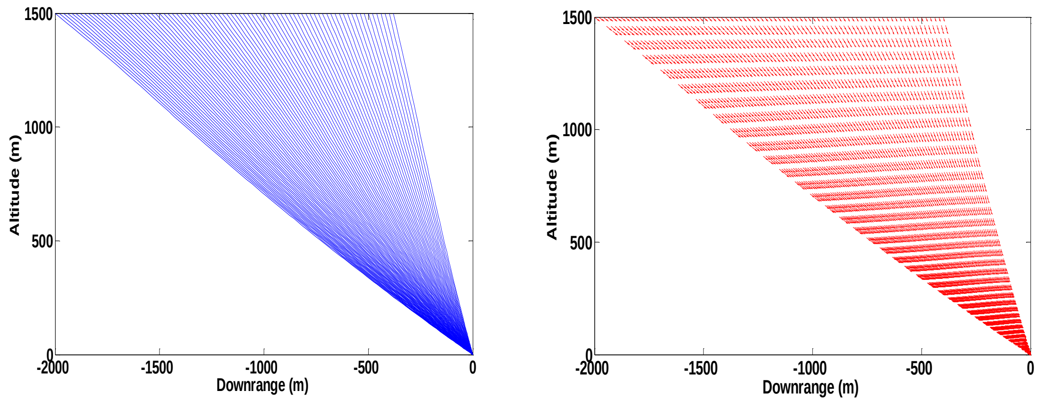

Figure 3. Samples of fuel-efficient trajectories and optimal velocity field. Right: Trajectories. Left: Velocity Vectors

Here, the thrust is limited to operate between a minimum value (T_{min}) and a maximum value (T_{max}). The problem formulated in Eq. (20-26) does not have an analytical solution and must be solved numerically. Open-loop, fuel-optimal thrust program can be obtained using optimal control software packages such as the General Pseudospectral Optimal Control Software (GPOPS[37]). Importantly, a set of open-loop trajectories may be employed to describe the relationship between the current position and the velocity vector, i.e. $\boldsymbol{v}_{OPT} = \boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$ that ensures a fuel-optimal path potentially subjected to flight-path constraints. GPOPS has been employed to numerically simulate the powered descent phase over Mars. A set of 201 2-D (vertical plane) fuel-optimal trajectories have been computed that are initiated at an altitude of *1500 meters*, with a downrange comprised between *-2000 meters* and *0 meters* (vertical descent).

For each case, the initial descent velocity has been kept constant (*-75 m/sec*) whereas the initial lateral velocity has been varied linearly between 100 m/sec (at *-2000 meters* downrange) to *0 m/sec* (at *0 meters* downrange). The lander is assumed to be a small robotic vehicle, with six throttlable engines, one pointing in each direction ($I_{sp} = 292\,\mathrm{sec}$). For these simulations, the only dynamical force included is the gravitational force of the moon, as seen in Eq. (21). The lander is assumed to weigh 1900 kg (wet mass) and is capable of a maximum (allowable) thrust of 13 kN as well as a minimum (allowable) thrust of 5kN. For each of the optimal trajectories, the position (ELM input) and velocity (ELM output) has been recorded 161 along the trajectories. Note that the sampling has been naturally established as an input to GPOPS to compute an accurate optimal solution. The total traning set is comprised of N = 201*161 = 32361 training samples. Figure 3, shows trajectories and velocity sampled from the training set.

ELM design, training and test: A typical arichitecture of a SLFN has been described in Figure 2. The network is comprised of an input layer (three nodes describing the current position) and hidden layer (number of nodes defined by the designer) and an output layer (three nodes that output the component of the optimal position). For a set of L hidden nodes and the above specified training set, the following steps are taken to design the SLFN capable of approximating the $\boldsymbol{v}_{OPT} = \boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$**:**

Step1: randomly generate the parameters describing the hidden nodes $(\boldsymbol{a}_i, b_i),\ i = 1, \dots, L$

Step 2: Compute the hidden layer output matrix $\boldsymbol{H}$

Step 3: Compute the output weight vector $\boldsymbol{\beta}$ by solving $\boldsymbol{H}^{\dagger}\boldsymbol{\beta} = \boldsymbol{T}$

The matrix $\boldsymbol{H}^{\dagger} = \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}^{\dagger}$or $\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T$if $\boldsymbol{H}\boldsymbol{H}^T$ is singular) is called the Moore-Penrose generalized inverse matrix. In computing the weights $\boldsymbol{\beta}$, a positive value $1/\lambda$ was added to the diagonal of $\boldsymbol{H}^T\boldsymbol{H}$ to improve the stability. Importantly, the linear weights and the ELM output function are computed as follows

$$\boldsymbol{\beta} = \boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T} \tag{27}$$

$$\boldsymbol{v_{OPT}}(\boldsymbol{r}, \boldsymbol{r_d}) = \boldsymbol{h}(\boldsymbol{r})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{r})\boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{\lambda} + \boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T} \tag{28}$$

One critical aspect of the ELM implementation is the selection of the the hidden number of nodes which is a typical network design parameter. The matrix $\boldsymbol{H}$ has dimension NxL where N = 16180 (size of the training set) and therefore $\boldsymbol{H}\boldsymbol{H}^T$ has dimension LxL. The training occurs by computing $\boldsymbol{\beta}$ (see to Eq. (27)). Training is extremely efficient and fast (tens of seconds using a MATLAB script in a standard quad-core laptop). To select the optimal number of hidden nodes, the training phase has been repeated by systematically increasing the number of neurons and recording the training performances. Figure 4 reports the root square mean error as function of the number of hidden layers. As the number of hidden nodes is increased, the root mean square error stabilizes. A value of L = 600 nodes has been selected as optimal architecture.

The SLFN design is tested and validated on a set of data comprising data points belonging to the training set on which the network has not been trained. Fifty percent of the overall training set is devoted to test and validation. Indeed, after proper training, the SLFN is run on test and validation input points and the output response predicted. Results are reported in figure 5 where a complete regression analysis is reported. The later shows that the designed network has generalized well, i.e. has learned the $\boldsymbol{v}_{OPT} = \boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$ functional relationship on points not included in the training set.

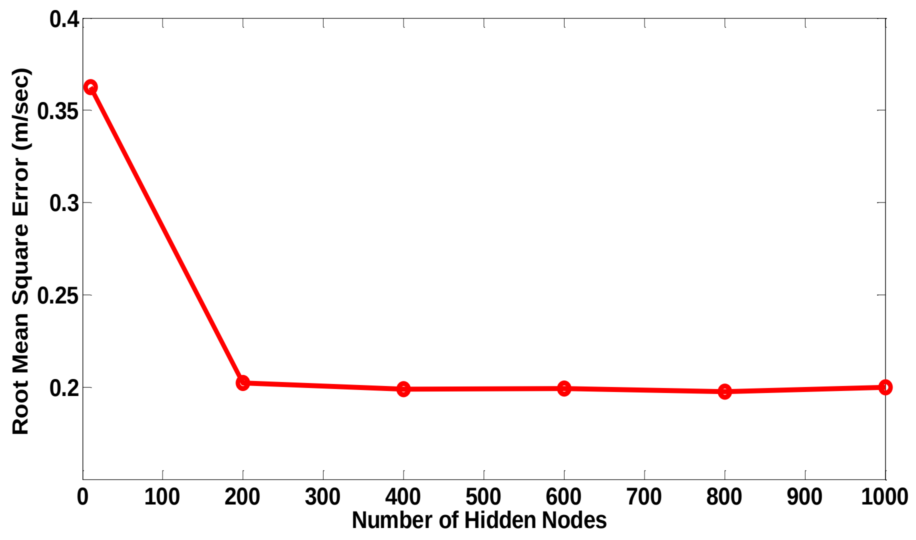

Figure 4: Root Mean Square Error (RMSE) as function of SLFN number of hidden nodes

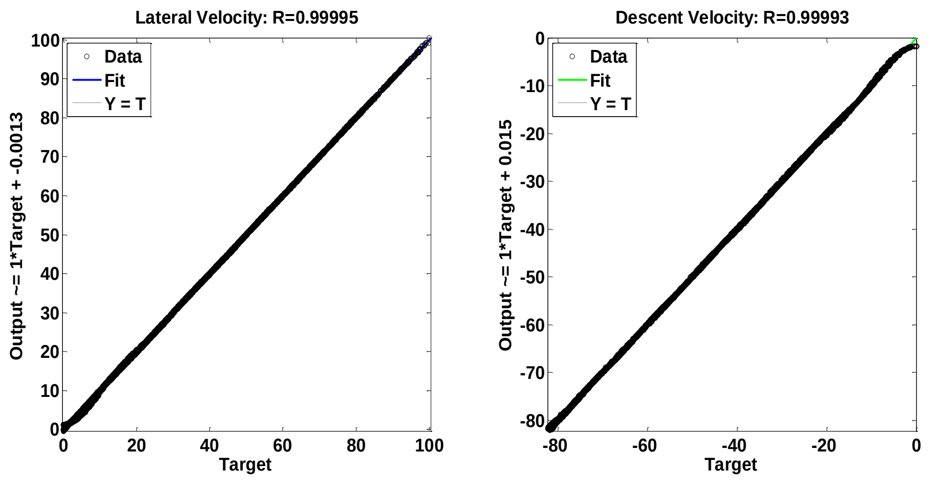

Figure 5: Test and validation regression plots. Left: Regression on lateral velocity. Right: regression on descent velocity.

Closed-Loop Guidance Design

The algorithm for neural-based trajectory shaping guidance heavily relies on the ability of the system to compute the optimal vector field as function of the position. Once position is available, the designed ELM is employed to compute the optimal vector field which becomes the desired velocity. At each time step the optimal velocity field is propagated on-board using an Euler propagator to compute the desired position. Both desired position and velocity are then fed to a standard Linear Quadratic Regulator (LQR) which tracks the state by trading between accuracy and effort. Figure 6 show a flow diagram of the proposed guidance scheme.

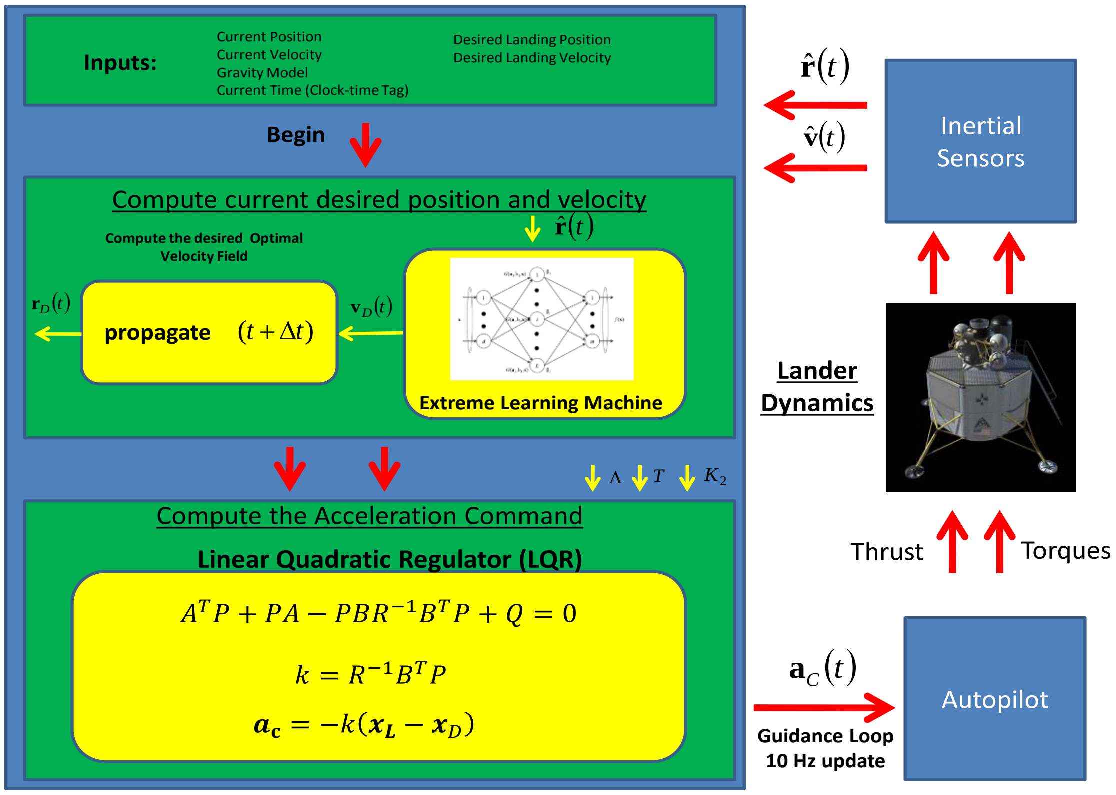

Figure 6: Guidance algorithm flow diagram

GUIDANCE SIMULATIONS AND PERFORMANCE ANALYSIS

The neural-based, trajectory shaping guidance algorithm devised using a combination of ELM theory for optimal velocity field approximation and closed-loop linear theory (tracking the velocity field) has been tested in a simulation scenario implementing the equations of motion Eq.(1-9). A powered descent phase describing the terminal guidance for Mars landing is considered. The spacecraft lander properties are the same as reported in the previous section. At this stage, the motion is constrained to occur in a vertical plane.The first set of simulations consider one single guided trajectory starting at $\boldsymbol{r}(0) = [1000{,}1500]^T meters$ with initial velocity $\boldsymbol{v}(0) = [50, -74]^T m/sec$. The guidance algorithms takes over immediately to guide the lander toward the desired point on the Martian surface (set to be the origin of the reference system) with zero velocity. Note that such conditions are not part of the training set employed to train and test the proposed ELM. The guidance algorithm is activated with 100 Hz frequency. Figure 7 shows the results of the simulation reporting A) the guided trajectory while moving through the velocity field generated by the designed SLFN, B) the altitude and downrange as function of time; C) the lateral and descent velocity as fucntion of time; and D) the magnitude of the thrust command.

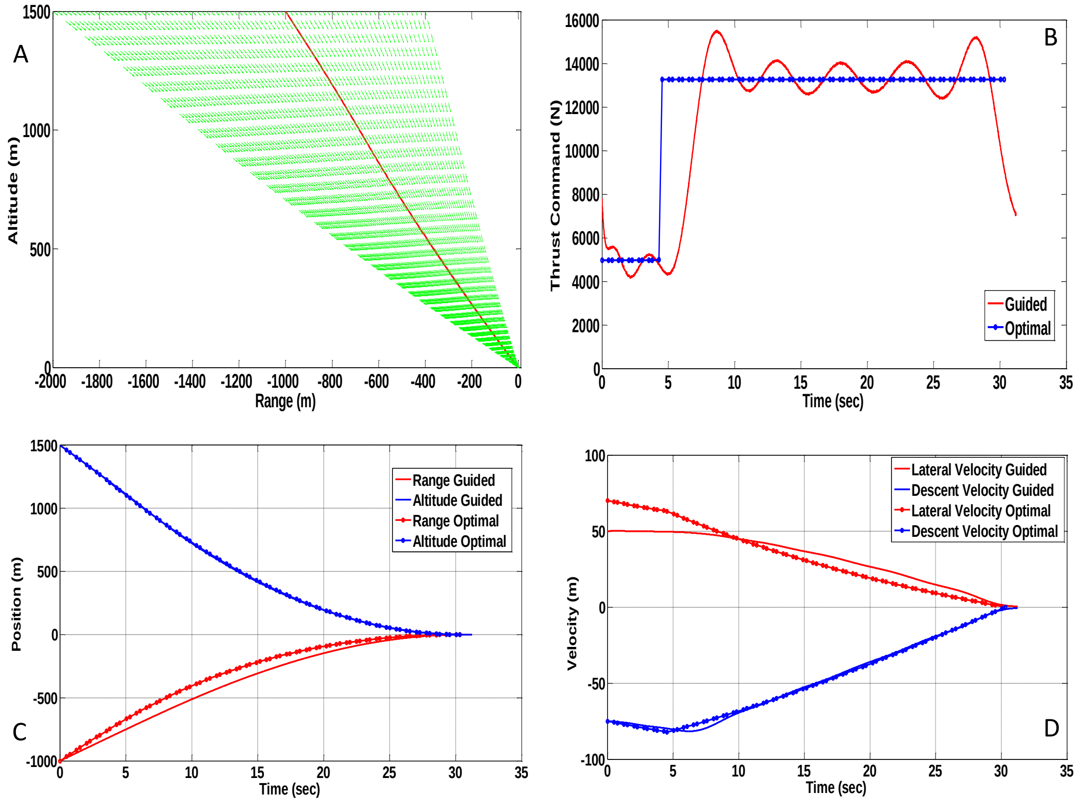

Figure 7. Single feedback trajectory simulation employing the neural-based trajectory shaping scheme and comparison with a GPOPS open-loop optimal solution. A) Descent trajectory tracking the optimal velocity field. B) history of the optimal and commanded thrust. C) History of altitude and downrange. D) History of lateral and descent velocity.

Importantly, the history of position, velocity and thrust have been compared with a newly generated numerical optimal solution generated via GPOPS that starts with the same initial conditions as the guided trajectory. Performance are comparable although it is worth noting that the neural-based trajectory shaping algorithm has not been designed to track the GPOPS-generated optimal fueld solution, but tracks the velocity field as approximated by the ELM. The open-loop, fuel efficient numerical solution achieve the desired target position with exactly zero velocity. The guided trajectory will generate guidance residual errors and it is not expect to have the same accuray of the open-loop ideal case. The magnitude of the thrust command oscillate as function of time as reported in Figure 7. This is due to the LQR design that has been chosen for the specific implementation. Importantly, in the last 10 seconds of the powered descent, the thrust command is reduced, which is probably the major cause of residual guidance errors. As noted it is not expected the guidance algorithm track exactly open-loop optimal trajectories. Indeed, as the lander moves across the vector field, the position associated to the guided trajectory trigger ELM outputs that approximate optimal velocity vectors associated to different fueld-optimal trajectories. Figure 8 shows the history of the mass for the guided lander versus the history of the optimal mass as computed by GPOPS. Interesting, the optimal case has a lower mass which implies larger amount of propellant. This is due to the fact that mass of propellant has been traded for accuracy in position and errors in terminal velocity (order of *1 m/sec*). The latter implies that not all lander's kinetic energy has been taken out by the guidance system. Moreover, during the guided phase, the lander does not track necessary that specific optimal solution with the same initial conditions, but other optimal trajectories or a combination of those that exhibit lowe mass of propellant.

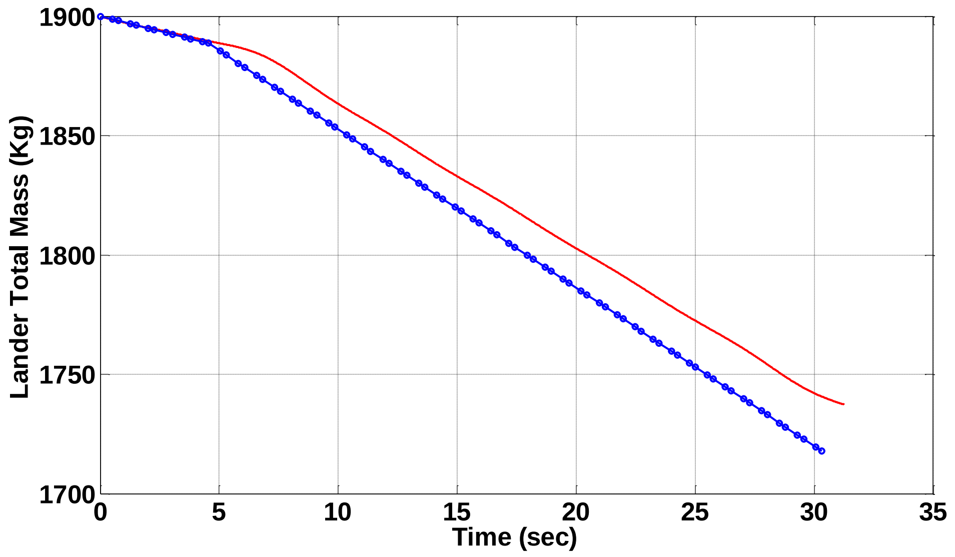

Figure 8: History of the lander's total mass. Red: neural-based trejatory shaped guidance. Blue: open-loop optimal

To evaluate further the guidance algorithm performance, a set of 1000 Monte Carlo simulations have been implemented and terminal guidance error recorded. A set of randomly generated initial conditions have been generated by randomly perturbing the samples of initial conditions taken from the training set. Initial altitude and downrange are perturbed by a gaussian noise with zero mean and *10 meters* standard deviation (1σ). Initial lateral and descent velocities are perturbed by a gaiussian noise with zero mean and *2 m/sec* standard deviation (1σ). Table 1 reports the statistic of the residual guidance error. Figure 9 shows the histogram of the terminal downrange error as well as terminal lateral and impact velocity. The guidance algorithm is shown to perform well with 1) a maximum absolute downrange error of *1.27 meters*; 2) a maximum lateral velocity error of *0.9 m/sec*; and 3) a maximum impact (descent) velocity of *-0.91 m/sec*.

Table 1: Landing statistics for the Monte Carlo Simulations

	Mean	Standard Deviation
Downrange Error (m)	-0.0680	0.7366
Lat Velocity Error (m/sec)	0.4762	0.2226
Imp Velocity Error (m/sec)	-0.7319	0.0652

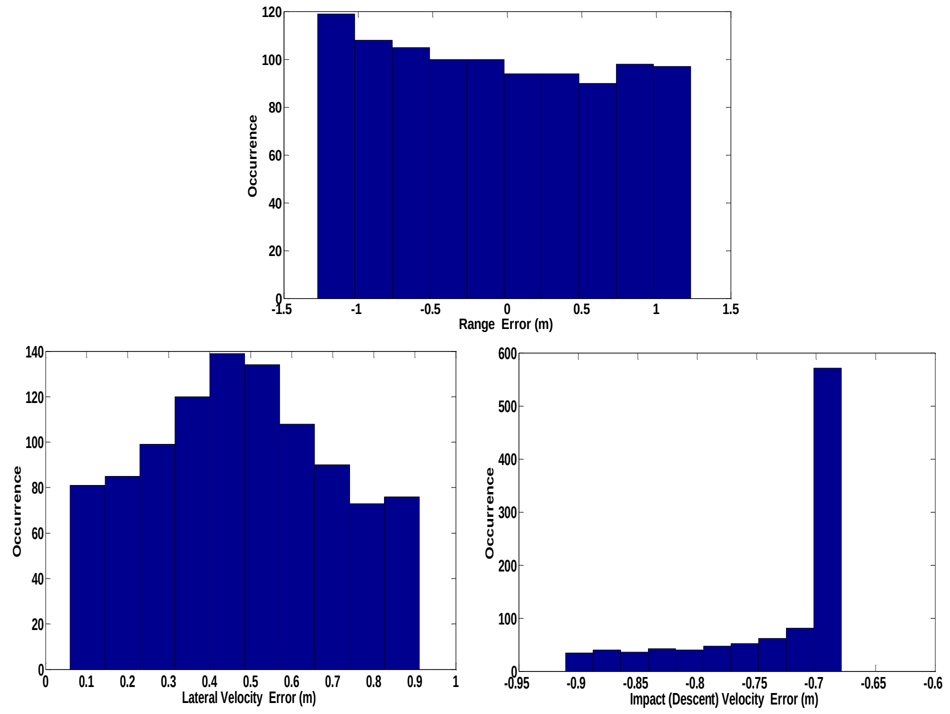

Figure 9: Landing histograms for the 1000 Monte Carlo simulations. Top: Error range. Bottom Left: Lateral velocity error. Bottom Right: Impact (descent) velocity error.

CONCLUSIONS AND FUTURE EFFORTS

This paper presented an approach to feedback guidance for planetary landing that may represent the next evolution of path shaping algorithms. Conventional path shaping algorithms, as investigated for lunar descent and landing, have been based on the idea of defining a potential function that exhibits the typical properties of a Lyapunov function. In previous studies, potential functions have been selected to be analytical (e.g. quadratic function or any of its linear combination). Despite the convenience (e.g. vector field and acceleration command can be computed analytically), the families of trajectories generated in this fashion in generally non-optimal. The next evolution of trajectory shaping employs modern machine learning techniques to approximate a potential function as function of position that generates trajectories that are attractive to the target and fuel efficient. Apparently, what is really needed is an algorithm that computes the optimal velocity field as function of the spacecraft actual position. Fuel efficient trajectories and velocities cannot be computed analytically, but numerical computation is required by using methods borrowed from optimal control theory (e.g. pseudo-spectral methods). ELM can be designed and trained on the output of open-loop, fuel-efficient trajectories generated via GPOPS. Such trajectories are shown to be convergent to the target and can be easily generated off-line. EML have shown to be fast and accurate in learning the desired functional relationship between the spacecraft position and the optimal velocity field. For real-time implementation, the guidance algorithm evaluate current position and velocity, compute via ELM the actual desired velocity and employs a LQR to track the velocity field. Guidance simulations have demonstrated the ability of the guidance algorithm to drive the lander toward the desired position exhibiting low residual errors in both terminal downrange, lateral velocity and impact velocity.

Future work needs to be done to implement and evaluate the neural-based trajectory shaping guidance performance in a full 3-D environment. Importantly, the extension is fairly straightforward. For example one can use the 2-D SLFN as reference and generate an axial-symmetric conical optimal vector field, by rotating the current 2-D field around the vertical (z-axis).

REFERENCES

[1]A. A. Wolf, J. Tooley, S. Ploen, M. Ivanov, B. Acikmese, K. Gromov, Performance Trades for Mars Pinpoint Landing, IEEE Aerospace Conference Proceedings, March 2006, IEEE-1661, March 2006.

[2] A. Wolf, E. Sklyanskly, J. Tooley, B. Rush, Mars Pinpoint Landing Systems Trades, AAS/AIAA Astrodynamics Specialist Conference Proceedings, AAS 07-310, August 19-23, 2007

[3] R.D. Braun and R.M. Manning, “Mars Exploration Entry, Descent, and Landing Challenges,” Journal of Spacecraft and Rockets, Vol. 44, No. 2, 310-323, March – April 2007.

[4] B. Steinfeldt , M. Grant, D. Matz, and R. Braun, Guidance, Navigation, and Control Technology System Trades for Mars Pinpoint Landing, AIAA Guidance, Navigation, and Control Conference, AIAA Paper 2008-6216, 2008.

[5] D. R., Williams, “Mars Pathfinder Landing Site,” Dec. 2004, http:// nssdc.gsfc.nasa.gov/planetary/marsland.html [retrieved 26 March 2012].

[6] R. Shotwell, Phoenix-the first Mars Scout mission, *Acta Astronautica*, Volume 57, Issue 2-8, (2005), p. 121-134.

[7] P. C., Knocke, G. G., Wawrzyniak, B. M., Kennedy, P. N., Desai, T. J., Parker, M. P., Golombek, T. C., Duxbury, and Kass, “Mars Exploration Rovers Landing Dispersion Analysis,” Proceedings of the AIAA/AAS Astrodynamics Specialist Conference, AIAA Paper 2004- 5093, 2004.

[8] D., A., Spencer, D., S., Adams, E., Bonfiglio, M., Golombek, R., Arvidson, K., Seelos, Phoenix Landing Site Hazard Assessment and Selection, Journal of Spacecraft and Rockets, vol. 46, issue 6, (2009), pp. 1196-1201

[9] A. Steltzner, A, Mars Science Laboratory Entry, Descent and Landing System, 2006 IEEE Aerospace Conference, Paper 1497, Big Sky, Montana, March 2006.

[10]http://marsoweb.nas.nasa.gov/landingsites/msl/4th_workshop/program.html (Retrieved March 26, 2012)

[11] A. D., Steltzner, D. M., Kipp, A., Chen, P. D.,Burkhart, C. S., Guernsey, G. F., Mendeck, R. A., Mitcheltree, R. W., Powell, T. P., Rivellini, A. M., San Martin, D. W., Way, Mars Science Laboratory Entry, Descent, and Landing System, *IEEE Aerospace Conference* Paper No. 2006-1497, Big Sky, MT,(2006)

[12] R. Furfaro, J., M. Dohm, W. Fink, J. S. Kargel, D. Schulze-MakuchA. G. Fairén, P. T. Ferré, A. Palmero-Rodriguez, V. R. Baker, T., M., Hare, M. A. Tarbell, H. H. Miyamoto, G. Komatsu, The Search for Life Beyond Earth Through Fuzzy Expert Systems; *Planetary and Space Science*, (2008) Volume 56, Issues 3-4, 448-472.

[13] G. Wells, et al., Entry, Descent, and Landing Challenges of Human Mars Exploration, *29th AAS Guidance and Control Conference*, AAS 06-072, Breckenridge, Colorado, February 2006.

[14] K.-Y., Tu, M. S., Munir, K. D., Mease, and D. S., Bayard, Drag-Based Predictive Tracking Guidance for Mars Precision Landing, *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, 2000, pp. 620–628.

[15] U., Topcu, J., Casoliva, and K., Mease, Fuel Efficient Powered Descent Guidance for Mars Landing, AIAA Paper 2005-6286, 2005.

[16] R., Sostaric, and J., Rea, Powered Descent Guidance Methods for the Moon and Mars, AIAA Paper 2005-6287, 2005.

[17] F., Najson, and K., Mease, A Computationally Non-Expensive Guidance Algorithm for Fuel Efficient Soft Landing, AIAA Guidance, Navigation, and Control Conference, San Francisco, AIAAPaper 2005-6289, 2005.

[18] C. D’Souza, An Optimal Guidance Law for Planetary Landing, AIAA Guidance, Navigation, and Control Conference, AIAA Paper 1997-3709, 1997.

[19] B., Acikmese, and S. R., Ploen, Convex Programming Approach to Powered Descent Guidance for Mars Landing, Journal of Guidance, Control, and Dynamics, Vol. 30, No. 5, 2007, pp. 1353–1366.

[20] D. A., Benson, G. T., Huntington, T. P., Thorvaldsen, and A. V., Rao, Direct trajectory optimization and costate estimation via an orthogonal collocation method. Journal of Guidance, Control, and Dynamics, 29(6), 2006, 1435-1440.

[21] J. F., Sturm, Using SeDuMi 1.02, a MATLAB Toolbox for Optimization Over Symmetric Cones, Optimization Methods and Software, Vol. 11, No. 1, 1999, pp. 625–653.

[22]B. Acıkmese and L. Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2), 2011.

[23] Y., Nesterov, and A., Nemirovsky, Interior-Point Polynomial Methods in Convex Programming, SIAM, Philadelphia, PA, 1994.

[24] L. Blackmore, B. Acıkmese, and D. P Scharf. Minimum landing error powered descent guidance for Mars landing using convex optimization. *AIAA Journal of Guidance, Control and Dynamics*, 33(4):1161–1171, 2010.

[25] A. R., Klumpp, A Manually Retargeted Automatic Landing System for the Lunar Module (LM), *Journal of Spacecraft and Rockets*, Volume 5, Issue 2, 1968, pp 129-138.

[26] A. R., Klumpp, Apollo Guidance, Navigation, and Control: Apollo Lunar-Descent Guidance, Massachusetts Inst. of Technology, Charles Stark Draper Lab., TR R-695, Cambridge, MA, June 1971.

[27]A. R., Klumpp, Apollo Lunar Descent Guidance, *Automatica*, Volume 10, Issue 2, 1974, pp. 133-146.

[28] G. Singh, A. SanMartin, and E. Wong. Guidance and control design for powered descent and landing on Mars. *Aereospace Conference IEEE*, 2007.

[29]McInnes, Colin R. "Path shaping guidance for terminal lunar descent." *Acta Astronautica* 36.7 (1995): 367-377.

[30]Mcinnes, Colin R. "Potential function methods for autonomous spacecraft guidance and control." *Astrodynamics 1995: Proceedings of the AAS/AIAA Astrodynamics Conference, Halifax, Canada*. 1996.

[31] Huang, Guang-Bin, Dian Hui Wang, and Yuan Lan. "Extreme learning machines: a survey." *International Journal of Machine Learning and Cybernetics* 2.2 (2011): 107-122.

[32]Hagan, Martin T., Howard B. Demuth, and Mark H. Beale. *Neural network design*. Boston: Pws Pub., 1996.

[33]Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications." *Neurocomputing* 70.1 (2006): 489-501.

[34]Bartlett, Peter L. "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network." *Information Theory, IEEE Transactions on* 44.2 (1998): 525-536.

[35]Huang, Guang-Bin, Lei Chen, and Chee-Kheong Siew. "Universal approximation using incremental constructive feedforward networks with random hidden nodes." *Neural Networks, IEEE Transactions on* 17.4 (2006): 879-892.

[36]Huang, Guang-Bin, Lei Chen, and Chee-Kheong Siew. "Universal approximation using incremental constructive feedforward networks with random hidden nodes." *Neural Networks, IEEE Transactions on* 17.4 (2006): 879-892.

[37]A. V., Rao, D. A., Benson, C., Darby, M. A., Patterson, C., Francolin, I., Sanders, et al. Algorithm 902: GPOPS, a MATLAB software for solving multiple phase optimal control problems using the Gauss pseudospectral method. ACM Transactions on Mathematical Software, 37(2), 2010, 22:1-22:39.