AAS 15-356

PERFORMANCE EVALUATION OF ARTIFICIAL NEURAL NETWORK-BASED SHAPING ALGORITHM FOR PLANETARY PINPOINT GUIDANCE

Jules Simo* **and Roberto Furfaro †** **and Joel Mueting ‡**

Computational intelligence techniques have been used in a wide range of application areas. This paper proposes a new learning algorithm that dynamically shapes the landing trajectories, based on potential function methods, in order to provide computationally efficient on-board guidance and control. Extreme Learning Machine (ELM) devises a Single Layer Forward Network (SLFN) to learn the relationship between the current spacecraft position and the optimal velocity field. 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. Furthermore, the proposed efficient algorithm is tested in typical simulation scenarios which include a set of Monte Carlo simulation to evaluate the guidance performances.

INTRODUCTION

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.[1,2] Thus, 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.[3,4] 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 planets surface.[1] Generating guidance algorithms for Mars landing has been the focus of many scientists and engineers.[5,6,7,8,9,10] 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). Although the method is originally devised to compute the reference trajectory used by the Lunar Exploration Module,[11,12] it is currently employed to generate a feasible reference trajectory comprising the three segments of the MSL powered descent phase. A fifth-order 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 time-to-go. In recent years several authors have tried determine reference trajectories (and guidance commands) that are fuel-optimal, i.e. minimum-fuel trajectories that satisfy the

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

†Assistant Professor, 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.

desired boundary conditions and possibly additional constraints. For such cases, analytical solutions are possible only for the energy-optimal landing problem with unconstrained thrust.[9] 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. Thus, such trajectories can be found only numerically using either direct or indirect methods.[13,14,15,16,17] However, 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.[18,19,20,21,22,23,24] More recently, Acikmese et al.[25] devised a convex optimization approach where the minimum-fuel soft landing problem is cast as a Second Order Cone Programming (SOCP).[26,27] The authors showed that the appropriate choice of a slack variable can convexify the problem.[28] Therefore, the resulting optimal problem can be solved in polynomial time using interior-point method algorithms.[29,30] In the following and for a prescribed accuracy, convergence is guaranteed to the global minimum within a finite number of iterations. In addition, the method is 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.[31] This paper develops 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 planets surface with zero velocity (soft landing). In a previous study, McInnes[32] developed a trajectory shaping approach for terminal lunar descent. The basic idea was to investigate potential function methods,[33] to generate families of trajectories that are globally convergent to the target landing location. The method relies on Lyapunovs 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 work, we selected a class of networks called Extreme Learning Machines (ELM).[34,35,36,37] Advantages of using such techniques will be illustrated in the upcoming sections.

In this paper, the neural-based trajectory shaping guidance performance will be implemented and evaluated in a full $3-D$ environment. Thus, the extension is fairly straightforward. For example one will use the $2-D$ SLFN[38] as reference and generate an axial-symmetric conical optimal vector field, by rotating the current $2-D$ field around the vertical $(z-axis)$. The proposed efficient algorithm is also tested in typical simulation scenarios which include a set of Monte Carlo simulation to evaluate the guidance performances.

PROBLEM STATEMENT

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.

EQUATIONS OF MOTION

The equations of motion of a spacecraft moving in the gravitational field of a planetary body can be described using Newtons law. Assuming a mass variant system and a flat planetary surface, the equations of motion is given by

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

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

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

where $\boldsymbol{r} = [x, y, z]^T$ and $\boldsymbol{v} = [v_x, v_y, v_z]^T$ are the position and velocity of the lander with respect to a coordinate system with origin on the planets 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 landers propulsion system, g_0 is the reference gravity, and $\boldsymbol{a_P}$ is a vector that accounts for unmodeled forces (e.g. thrust misalignment, effect of higher order gravitational harmonics, atmospheric drag, etc.).

The equations of motion can be explicitly written in their scalar form as

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

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

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

$$\dot{v}_x = -g_x(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_x + a_{px}, \tag{7}$$

$$\dot{v}_y = -g_y(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_y + a_{py}, \tag{8}$$

$$\dot{v}_z = -g_z(r) + \left(\frac{\boldsymbol{T}}{m_L}\right)_z + a_{pz}. \tag{9}$$

The mathematical model described in Eqs. (1-9) is employed to simulate the spacecraft descent dynamics by the proposed guidance law. In order to ensure a successful soft landing, the lander will stay above the planetary surface all the time during the powered descent phase.

GUIDANCE ALGORITHM DEVELOPMENT

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).[32,33] The guidance reference frame is shown in Figure 1. 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,[32,33] 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.

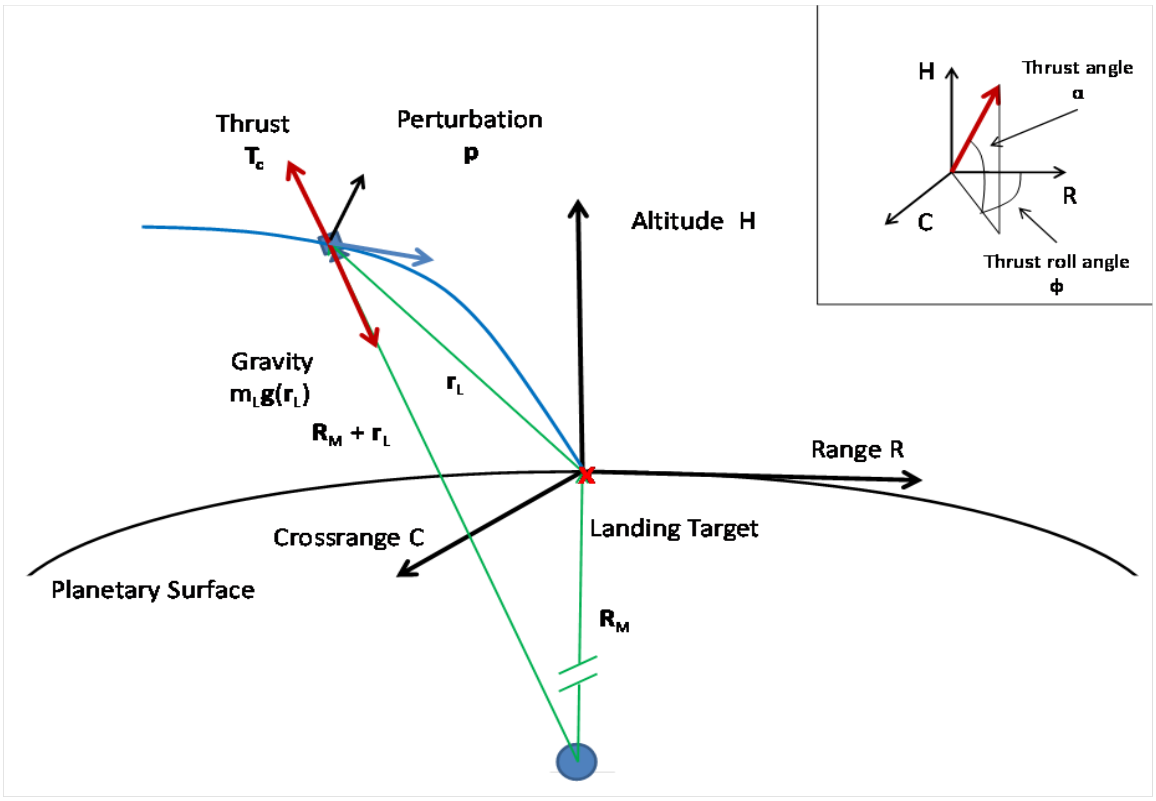


Figure 1 Guidance reference frame and free-body force diagram for a planetary lander during the powered descent to the designated target.

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 r as the current spacecraft position and r_d the desired position on the planets surface, we have

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

According to the second Lyapunovs 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[32] used the potential function method to generate families of descent path toward the lunar surface as follows:

- The magnitude of the spacecraft velocity is shaped by specifying a pre-defined velocity profile as function of the altitude

- The direction of the velocity vector, which in turn define 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.

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 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

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

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

where $\boldsymbol{\Lambda}$ is the 3×3 matrix of partial derivatives of the velocity components and $\boldsymbol{g}$ is the gravity vector. 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.

EXTREME LEARNING MACHINES

Computational intelligence 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. Neural Networks (NN), 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 the 'generalized' Single Layer Forward Networks (see 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[39] 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.

Consider a SLFN with L hidden nodes, as shown in Figure 2. The output function can be represented as follows

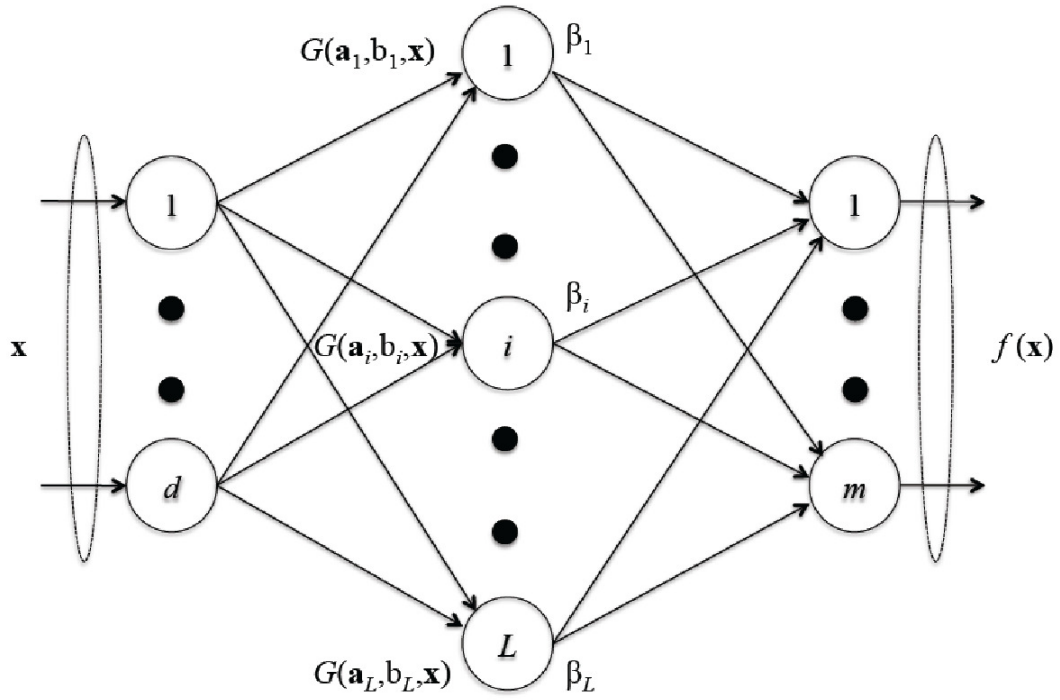


Figure 2 Typical architecture of a Single Layer Forward Network (SLFN) which is the most fundamental EML.

$$f_L(x) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) \tag{14}$$

with $\boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{\beta}_i \in \mathbb{R}^m$.

For additive nodes with activation function g

$$G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(\boldsymbol{a}_i \boldsymbol{x} + b_i) \tag{15}$$

with $\boldsymbol{a}_i \in \mathbb{R}^m, b_i \in \mathbb{R}$.

For RBF nodes with activation function g

$$G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(b_i||\boldsymbol{x} - \boldsymbol{a}_i|| \quad with \quad \boldsymbol{a}_i \in \mathbb{R}^m, b_i \in \mathbb{R}^+. \tag{16}$$

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

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = \boldsymbol{t}_j, \quad for \quad j \in 1, \cdots, N. \tag{17}$$

Compactly, we have

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}, \tag{18}$$

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

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

with

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times m},$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1^T \\ \vdots \\ \boldsymbol{T}_N^T \end{bmatrix}_{N \times m}.$$

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

Theorem 1 *Given any non-constant piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$, if $span\{G(\boldsymbol{a}, b, \boldsymbol{x}) : (\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}\}$ is dense in $\mathbb{L}^2$, 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 $||f(\boldsymbol{x}) - \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\boldsymbol{x})||$.*

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 $\hat{\boldsymbol{\beta}}$ such that

$$||\boldsymbol{H}\hat{\boldsymbol{\beta}} - \boldsymbol{T}|| = min_{\boldsymbol{\beta}}||\boldsymbol{H\beta} - \boldsymbol{T}||. \tag{19}$$

FUEL-EFFICIENT VELOCITY FIELD COMPUTATION USING 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 $v(\boldsymbol{r}, \boldsymbol{r}_d)$ 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 $v(\boldsymbol{r}, \boldsymbol{r}_d)$ 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) 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)[35,36] to approximate the fuel-efficient velocity field.

NETWORK DESIGN AND TRAINING SET GENERATION USING 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 $v(\boldsymbol{r}, \boldsymbol{r}_d)$. The development goes through the following stages: 1) training set generation, 2) SLFN architecture design, 3) Training phase and 4) testing and validation phase.

The minimum-fuel problem for planetary landing can be defined as follows: 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)

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

subject to

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

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

and the following boundary conditions

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

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

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

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

The thrust is limited to operate between a minimum value (T_{min}) and a maximum value (T_{max}). An open-loop, fuel-optimal thrust program can be obtained by using optimal control software packages such as the General Pseudospectral Optimal Control Software (GPOPS).[17] 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.

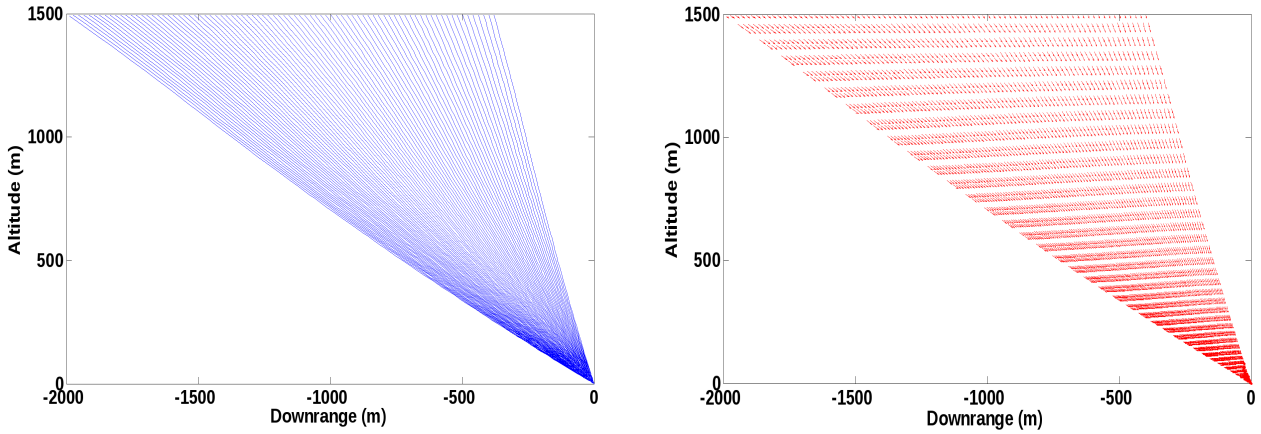


Figure 3. (a) Samples of fuel-efficient trajectoties; (b) Samples of fuel-efficient velocity vectors.

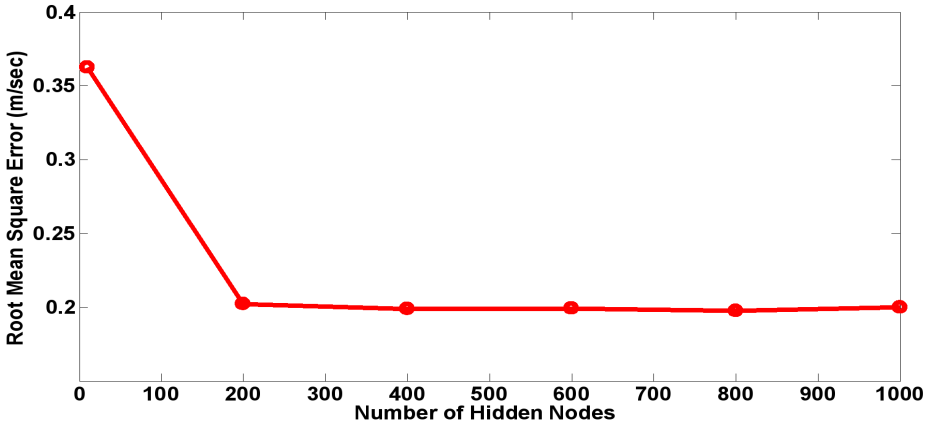


Figure 4. Root Mean Square Error (RMSE) as function of SLFN number of hidden nodes.

PARAMETER DESIGN

Samples from an optimal (continuous) vector field are generated by numerically computing a set of fuel-efficient trajectories via proper application of the optimal control theory. A set of $2-D$ (vertical plane)[38] 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 ($I_{sp} = 292\ sec$). For these simulations, the only dynamical force included is the gravitational force of the moon.The lander has a weight of 1900 kg (wet mass) and is capable of a maximum (allowable) thrust of $13\ kN$ as well as a minimum (allowable) thrust of $5\ kN$. 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

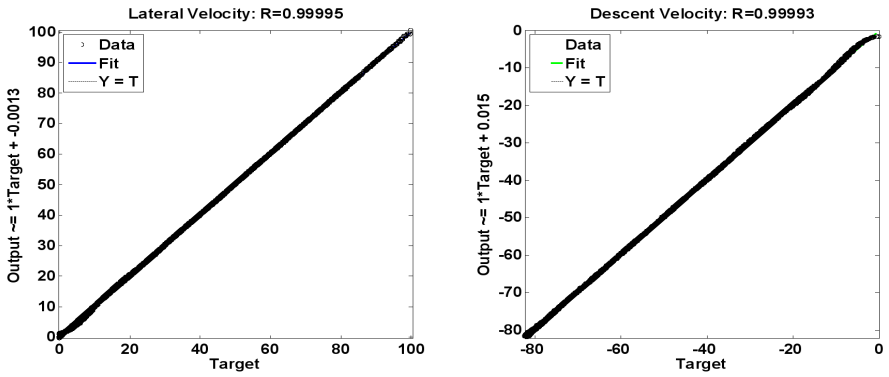


Figure 5 Test and validation regression plots: (a) Regression on lateral velocity; (b) Regression on descent velocity.

traning set is comprised of $N = 201 \times 161 = 32361$ training samples. Figure 3 shows trajectories and velocity sampled from the training set.

ELM DESIGN: TRAINING AND TEST

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 $\boldsymbol{v}_{OPT} = \boldsymbol{v}(\boldsymbol{r}, \boldsymbol{r}_d)$

- randomly generate the parameters describing the hidden nodes $(\boldsymbol{a}_i, b_i) \quad for \quad i = 1, \cdots, L$
- Compute the hidden layer output matrix $\boldsymbol{H}$
- Compute the output weight vector $\boldsymbol{\beta}$ by solving $\boldsymbol{H}^{+}\boldsymbol{\beta} = \boldsymbol{T}$

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

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

$$\boldsymbol{v}_{OPT}(\boldsymbol{r}, \boldsymbol{r}_d) = \boldsymbol{h}(\boldsymbol{r})\boldsymbol{\beta} = \boldsymbol{h}(\boldsymbol{r})\boldsymbol{H}^{T}\left(\frac{1}{\lambda} + \boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}}\right)^{-1}\boldsymbol{T}. \tag{28}$$

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 has dimension $N \times L$ where $N = 16180$ (size of the training set) and therefore $\boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}}$ has dimension L. The training occurs by computing $\boldsymbol{\beta}$ (Eq. (27)). 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 shows 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. A complete regression analysis is then illustrated in Figure 5. 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.

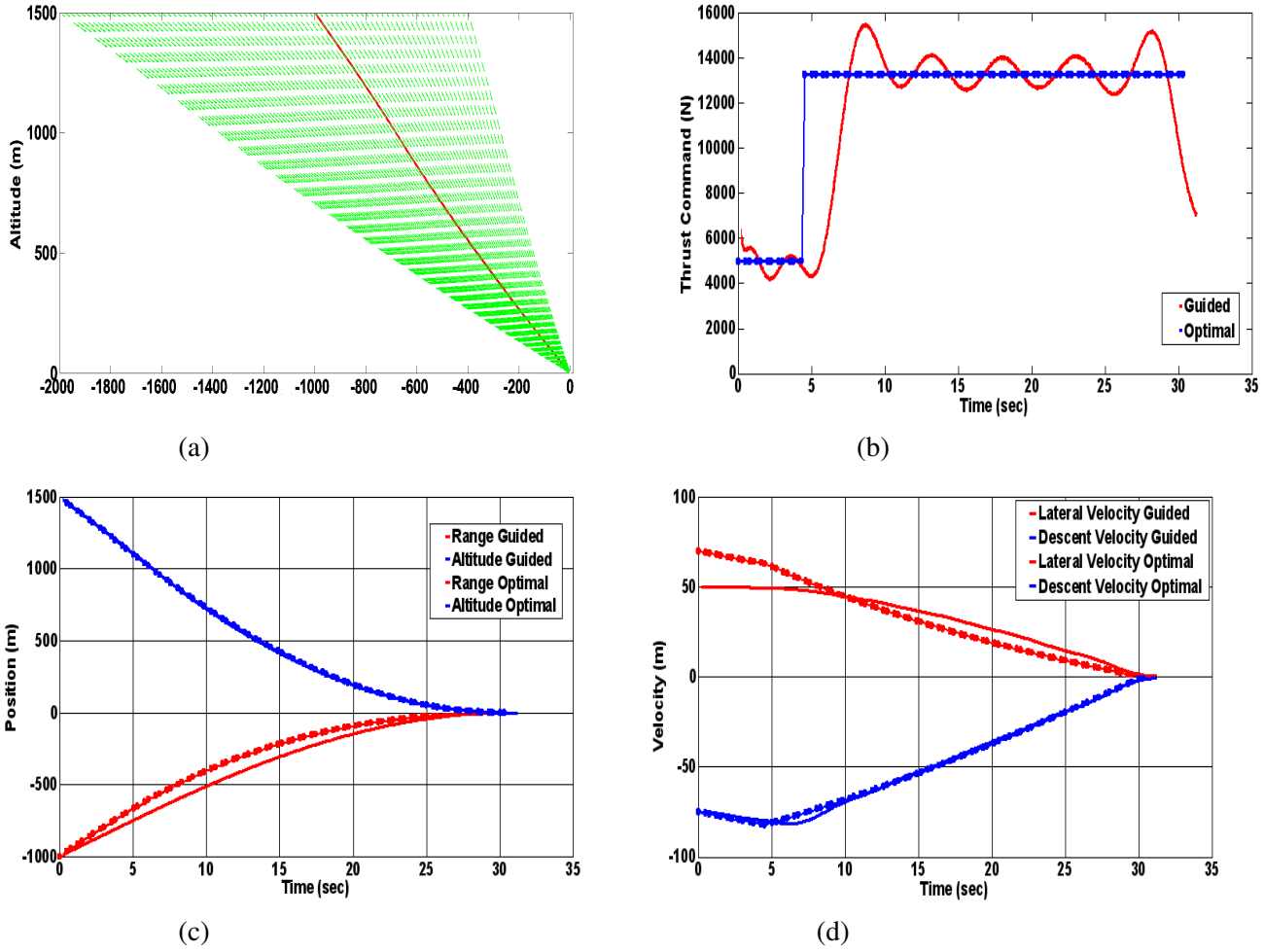


Figure 6 Single feedback trajectory simulation employing the neural-based trajectory shaping scheme and comparison with a GPOPS open-loop optimal solution: (a) Descent trajectory tracking the optimal velocity field; (b) History of the optimal and commanded thrust; (c) History of the Altitude and downrange; (d) History of the lateral and dexent velocity.

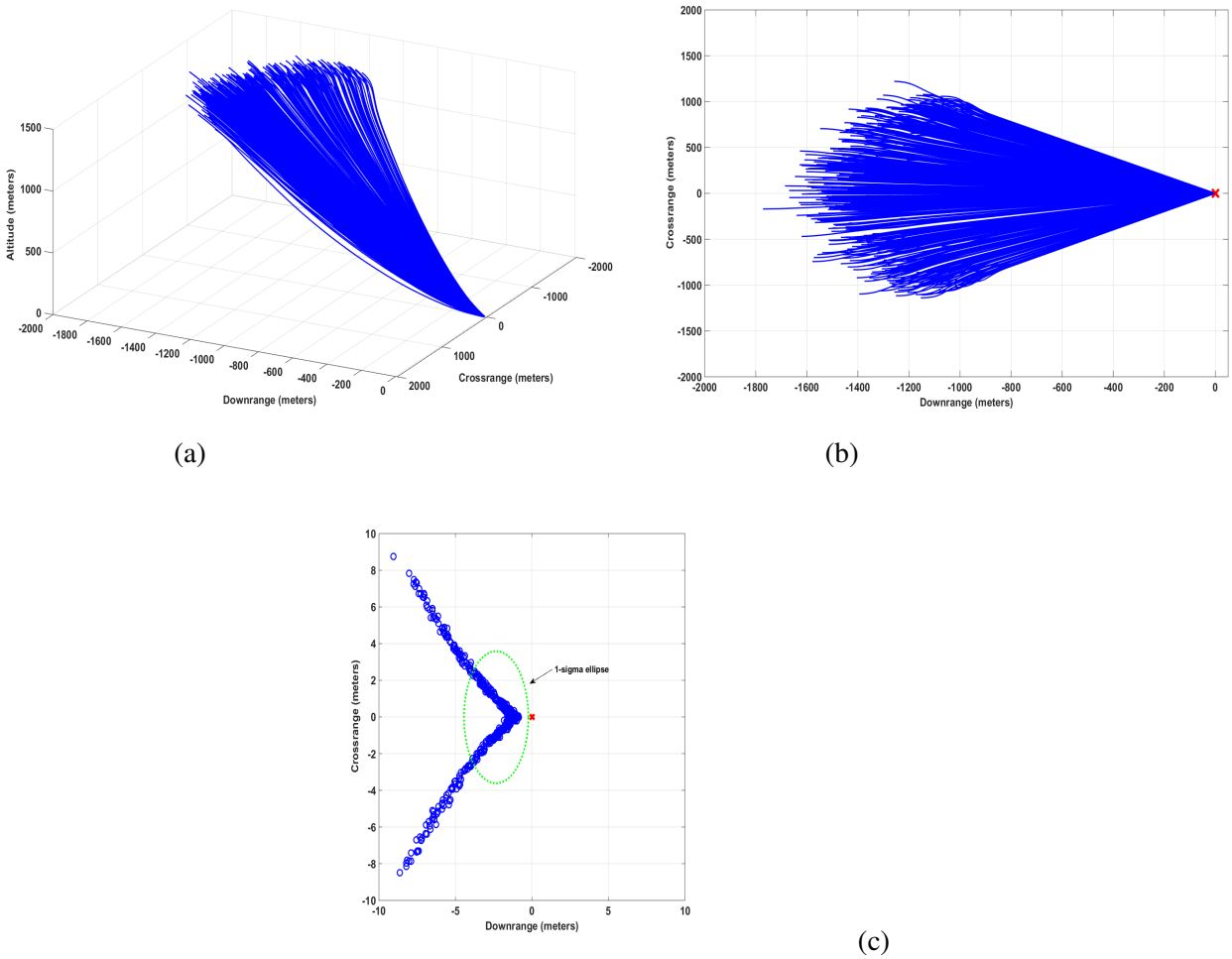


Figure 7. 3-D Monte Carlo trajectories and landing ellipses.

EVALUATION OF GUIDANCE ALGORITHM PERFORMANCE

Two-Dimensional Results

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 Eqs. (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 $\boldsymbol{r}(0) = [1000, 1500]\ m$ with initial velocity $\boldsymbol{v}(0) = [50, -74]^T\ m/s$. 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 10 Hz frequency. Figure 6 shows the simulation of a single feedback trajectory using the neural-based trajectory shaping approach and the comparison with a GPOPS open-loop optimal solution. Furthermore, the time 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 shown in Figure 6. 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.

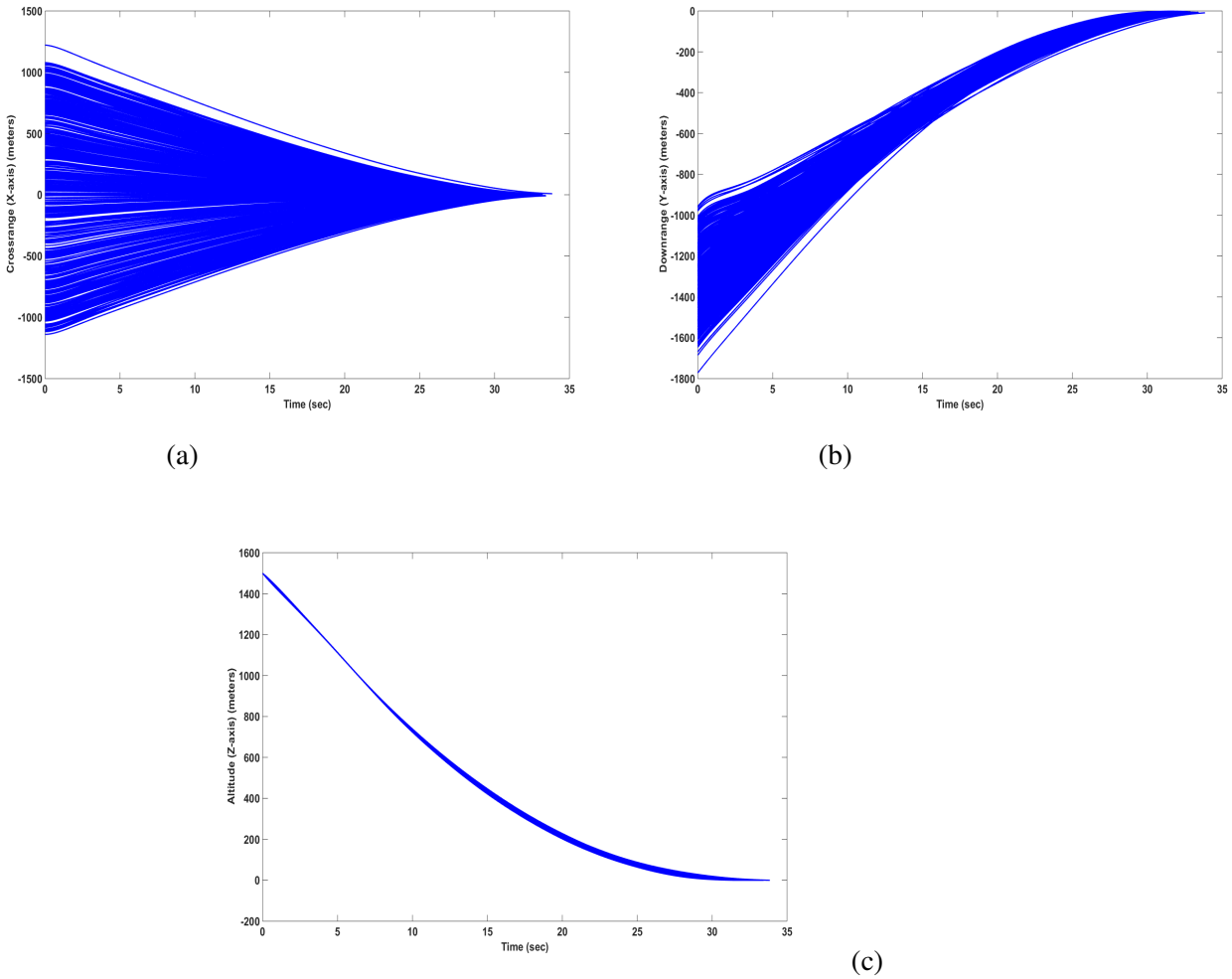
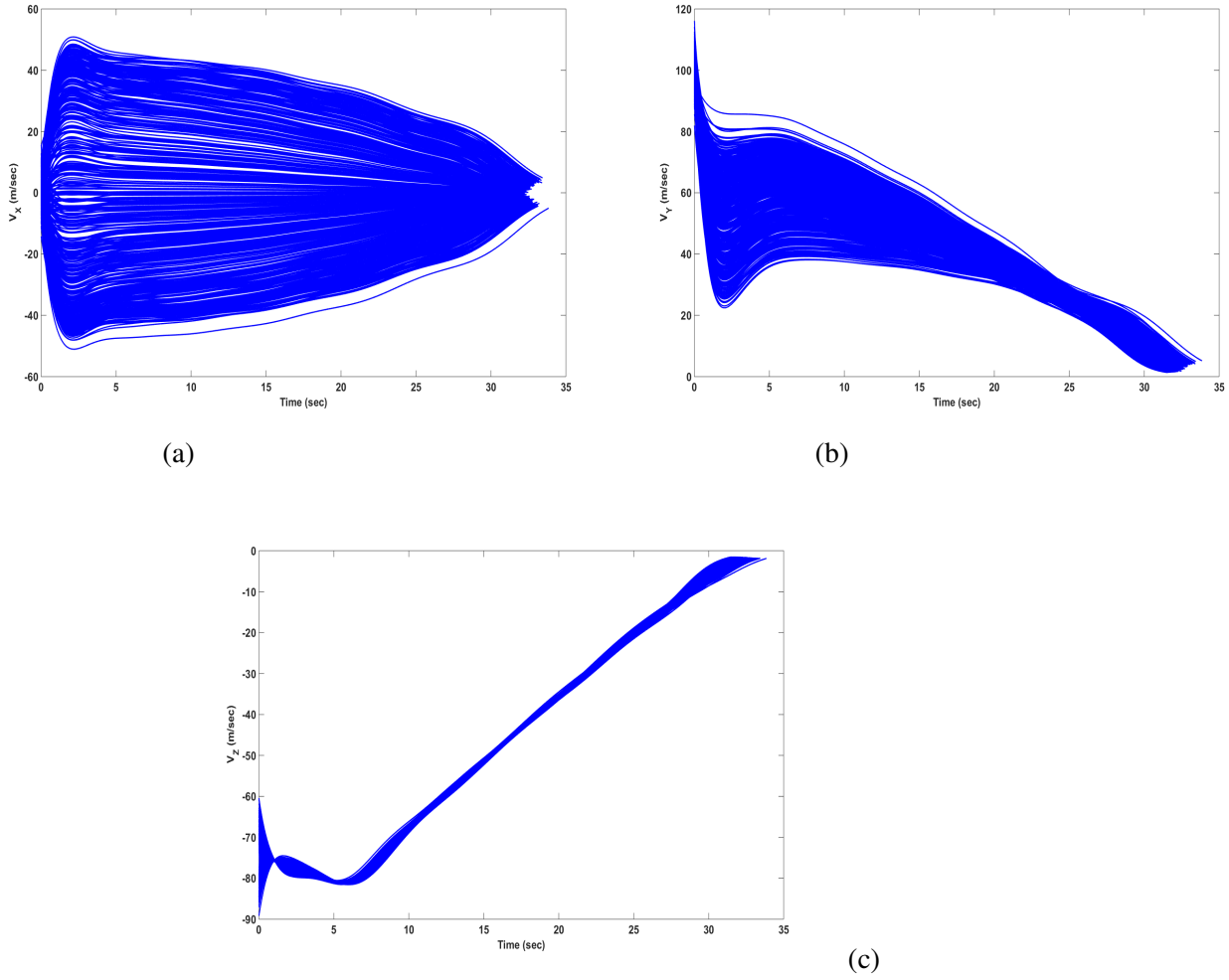


Figure 8. Crossrange, Downrange and Altitude vs time.

Three-Dimensional Results

In this section, the neural-based trajectory shaping guidance performance is implemented and evaluated in a full $3-D$ environment. is fairly straightforward. The $2-D$ SLFN[38] has been used as reference and generate an axial-symmetric conical optimal vector field, by rotating the current $2-D$ field around the vertical $(z-axis)$. To evaluate further the guidance algorithm performance, a set of 1000 Monte Carlo simulations have been implemented as shown in Figure 7. 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) (see Figure 8). Initial lateral and descent velocities are perturbed by a gaussian noise with zero mean and 2 m/sec standard deviation (1σ) as shown in Figure 9. Table 1 reports the landing statistics for the Monte Carlo Simulations. Figures 10, 11 show the landing histograms for the 1000 Monte Carlo simulations. The guidance algorithm is shown to perform well.

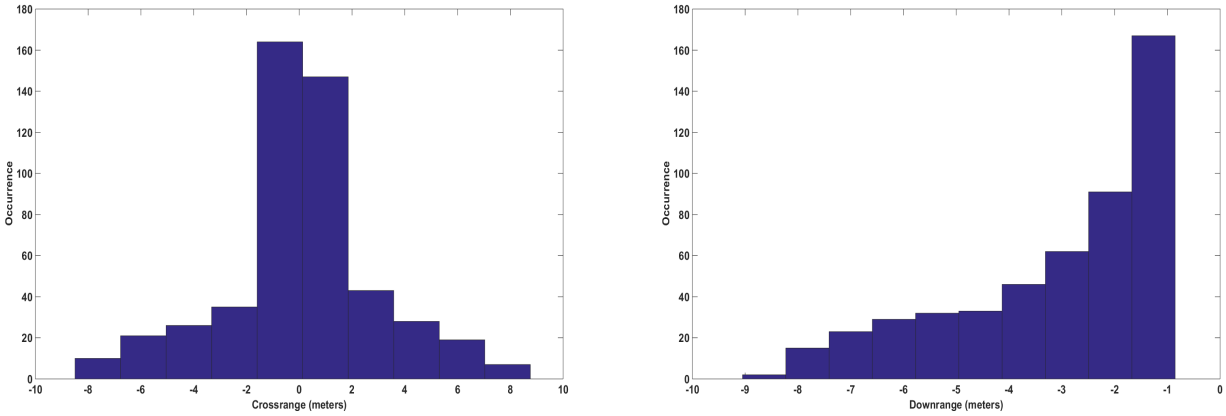


(a) (b) (c)

Figure 9. Velocity variation components.

Table 1. Landing Statistics for the Monte Carlo Simulations.

	Mean	Standard Deviation
Crossrange (m)	0.1125	2.8673
Downrange (m)	-3.1229	1.9642
Crossrange Velocity (m/s)	-0.0964	1.8498
Downrange Velocity (m/s)	2.4401	0.9434
Descent Velocity (m/s)	-1.6466	0.0874



(a) (b)

Figure 10. Landing histograms: Positions.

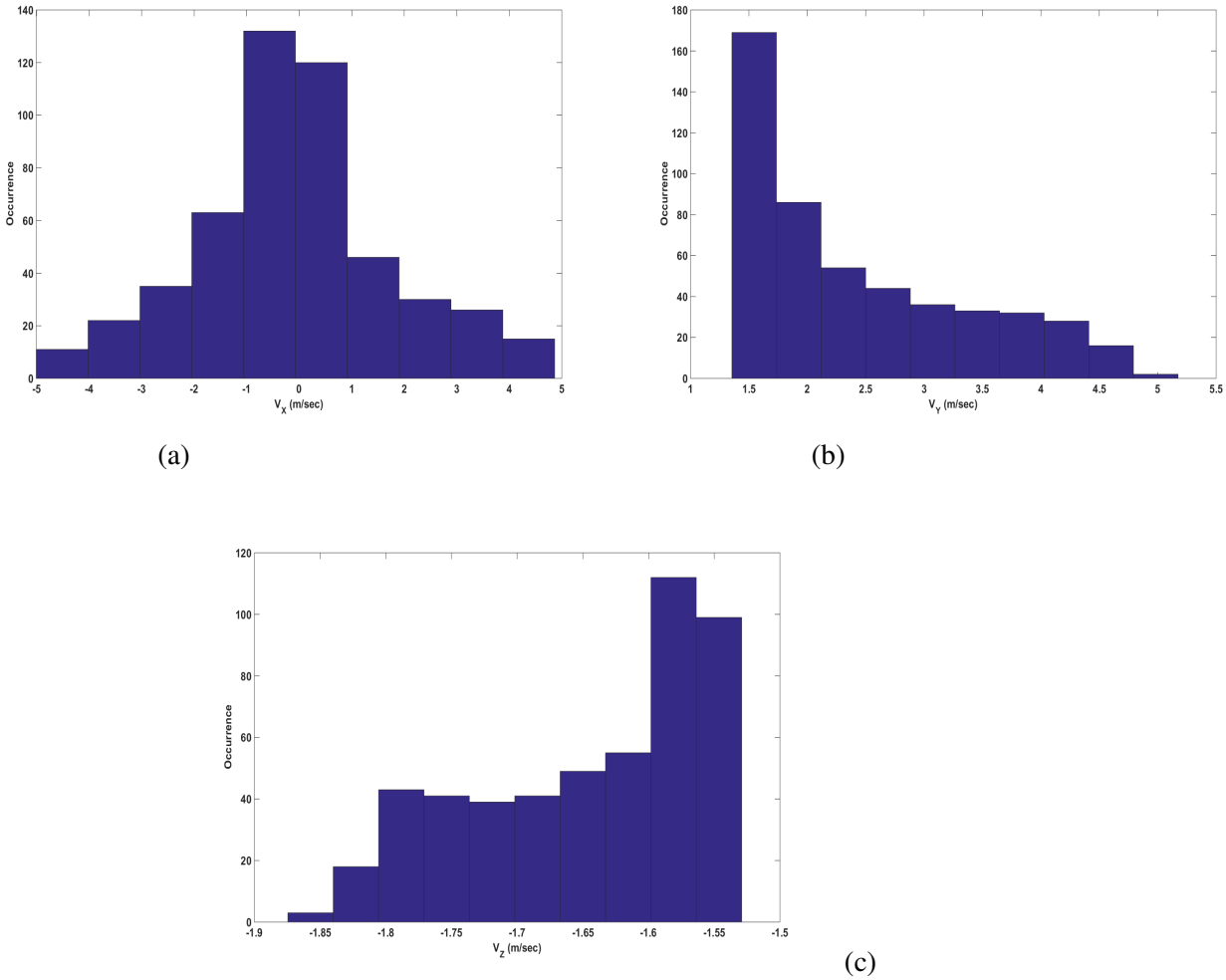


Figure 11 Landing histograms for the 1000 Monte Carlo simulations: Velocity variation components.

CONCLUSIONS

In this paper, the neural-based trajectory shaping guidance performance has been implemented and evaluated in a full $3-D$ environment. The proposed efficient algorithm is also tested in typical simulation scenarios which include a set of Monte Carlo simulation to evaluate the guidance performances. An approach to feedback guidance for planetary landing that may represent the next evolution of path shaping algorithms has also been illustrated. 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.

REFERENCES

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

[2] B. Steinfeldt, M. Grant, D. Matz, and R. Braun, "Guidance, Navigation, and Control Technology System Trades for Mars Pinpoint Landing," *Atmospheric Flight Mechanics Conference and Exhibit*, Honolulu, HI, August 18-21 2008. Paper AIAA 2008-6216.

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

[4] A. A. Wolf, Sklyanskly, J. Tooley, and B. Rush, "Mars Pinpoint Landing Systems Trades," *In AIAA/AAS Astrodynamics Specialist Conference*, Mackinac Island, Michigan, August 19-23 2007. Paper AAS 07-310.

[5] 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, July-August 2000, pp. 620–628.

[6] U. Topcu, J. Casoliva, and K. D. Mease, "Fuel Efficient Powered Descent Guidance for Mars Landing," *In AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, August 15 - 18, 2005. AIAA 2005-6286.

[7] R. R. Sostaric and J. R. Rea, "Powered Descent Guidance Methods for the Moon and Mars," *In AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, August 15 - 18, 2005. AIAA 2005-6287.

[8] F. Najson and K. D. Mease, "Computationally Inexpensive Guidance Algorithm for Fuel-Efficient Terminal Descent," *Journal of Guidance, Control and Dynamics*, Vol. 29, No. 4, July-August 2006, pp. 955–964.

[9] C. N. D'Souza, "An Optimal Guidance Law for Planetary Landing," *In AIAA Guidance, Navigation, and Control Conference*, New Orleans, Louisiana, 1997. AIAA 1997-3709.

[10] G. Singh, A. M. SanMartin, and E. C. Wong, "Guidance and Control Design for Powered Descent and Landing on Mars," *In Aerospace Conference IEEE*, Big Sky, Montana, March 3 - 10, 2007.

[11] A. R. Klumpp, "A Manually Retargeted Automatic Landing System for the Lunar Module (LM)," *Journal of Spacecraft and Rockets*, Vol. 5, No. 2, February 1968, pp. 129–138.

[12] A. R. Klumpp, "Apollo Lunar Descent Guidance," *Automatica*, Vol. 10, No. 2, March 1974, pp. 133–146.

[13] J. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control and Dynamics*, Vol. 21, No. 2, March-April 1998, pp. 193–207.

[14] F. Fahroo and M. Ross, "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control and Dynamics*, Vol. 25, No. 1, January-February 2002, pp. 160–166.

[15] O. von Stryk and R. Bulirsch, "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, Vol. 37, 1992, pp. 357–373.

[16] 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*, Vol. 29, No. 6, November-December 2006, pp. 1435–1440.

[17] A. V. Rao, D. A. Benson, C. Darby, M. A. Patterson, C. Francolin, and e. a. I. Sanders, "Algorithm 902: GPOPS, a MATLAB Software for solving Multiple Phase Optimal Control Problems using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, 2010, pp. 22:1–22:39.

[18] D. G. Hull, "Conversion of Optimal Control Problems into Parameter Optimization Problems," *Journal of Guidance, Control and Dynamics*, Vol. 20, No. 1, January-February 1997, pp. 57–60.

[19] D. Kraft, "On Converting Optimal Control Problems into Nonlinear Programming Problems," *In Computational Mathematical Programming* F15 *of NATO ASI Series,* K. Schittkowsky Ed., Springer, 1985, pp. 261–280.

[20] J. Vlassenbroeck and R. V. Dooren, "A Chebyshev Technique for Solving Nonlinear Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. 33, No. 4, April 1988, pp. 333–340.

[21] O. von Stryk, "Numerical Solution of Optimal Control Problems by Direct Collocation," *In Optimal Control - Calculus of Variation - Optimal Control Theory and Numerical Methods,* R. Bulirsch, A. Miele, J. Stoer and K. H. Well Eds., *International Series of Numerical Mathematics*, Vol. 111, Birkhäuser, 1993, pp. 129–143.

[22] E. D. Dickmanns and K. H. Well, "Approximate Solution of Optimal Control Problems using Third Order Hermite Polynomial Functions," *Lecture Notes in Computer Science*, Vol. 27, 1975, pp. 158–166.

[23] P. J. Enright and B. A. Conway, "Discrete Approximations to Optimal Trajectories using Direct Transcription and Nonlinear Programming," *Journal of Guidance, Control and Dynamics*, Vol. 15, No. 4, July-August 1992, pp. 994–1002.

[24] C. R. Hargraves and S. W. Paris, "Direct Trajectory Optimization using Nonlinear Programming and Collocation," *Journal of Guidance, Control and Dynamics*, Vol. 10, No. 4, 1987, pp. 338–342.

[25] B. Açikmeşe and S. R. Ploen, "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control and Dynamics*, Vol. 30, No. 5, September-October 2007, pp. 1353–1366.

[26] 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.

[27] J. F. Sturm, "Implementation of Interior Point Methods for Mixed Semidefinite and Second Order Cone Optimization Problems," *Optimization Methods and Software*, Vol. 17, No. 6, 2002, pp. 1105–1154.

[28] B. Açikmeşe and L. Blackmore, "Lossless Convexification of a Class of Optimal Control Problems with Non-Convex Control Constraints," *Automatica*, Vol. 47, No. 2, February 2011, pp. 341–347.

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

[30] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large Scale Constrained Optimization," *Technical Report NA 96-2, University of California, San Diego, CA, USA*, 1996.

[31] L. Blackmore, B. Açikmeşe, and D. P. Scharf, "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing using Convex Optimization," *Journal of Guidance, Control and Dynamics*, Vol. 33, No. 4, July-August 2010, pp. 1161–1171.

[32] C. R. McInnes, "Path Shaping Guidance for Terminal Lunar Descent," *Acta Astronautica*, Vol. 36, No. 7, 1995, pp. 366–377.

[33] C. R. McInnes, "Potential Function Methods for Autonomous Spacecraft Guidance and Control," *In AIAA/AAS Astrodynamics Specialist Conference*, Halifax, Nova Scotia, Canada, August 1995. Paper AAS 95-447.

[34] H. Guang-Bin, D. H. Wang, and Y. Lan, "Extreme Learning Machines: A Survey," *International Journal of Machine Learning and Cybernetics*, Vol. 2, 2011, pp. 107–122.

[35] H. Guang-Bin, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machines: Theory and Applications," *Neurocomputing*, Vol. 70, 2006, pp. 489–501.

[36] H. Guang-Bin, L. Chen, and C.-K. Siew, "Universal Approximation using Incremental Constructive Feedforward Networks with Random Hidden Nodes," *IEEE Transactions on Neural Networks*, Vol. 17, No. 4, July 2006, pp. 879–892.

[37] H. Guang-Bin and Z. Qin-Yu, "Real-Time Learning Capability of Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 17, No. 4, July 2006, pp. 863–878.

[38] R. Furfaro, J. Simo, B. Gaudet, and D. R. Wibben, "Neural-based Trajectory Shaping Approach for Terminal Planetary Pinpoint Guidance," *In AAS/AIAA Astrodynamics Specialist Conference*, Hilton Head, South Carolina, August 11 - 15, 2013. AAS 13-875.

[39] P. L. Bartlett, "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is more Important than the Size of the Network," *IEEE Transactions on Information Theory*, Vol. 44, No. 2, 1998, pp. 525–536.