AAS 13-328

MULTIPLE SLIDING SURFACE GUIDANCE FOR PLANETARY LANDING: TUNING AND OPTIMIZATION VIA REINFORCEMENT LEARNING

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

The problem of achieving pinpoint landing accuracy in future space missions to extra-terrestrial bodies such as the Moon or Mars presents many challenges, including the requirements of higher accuracy and more flexibility. These new challenges may require the development of novel and more advanced guidance algorithms. Conventional guidance schemes, which generally require a combination of off-line trajectory generation and real-time, trajectory tracking algorithms, have worked well in the past but may not satisfy the more stringent and difficult landing requirements imposed by future mission architectures to bring landers very near to specified locations. In this paper, a novel non-linear guidance algorithm for planetary landing is proposed and analyzed. Based on Higher-Order Sliding Control (HOSC) theory, the Multiple Sliding Surface Guidance (MSSG) algorithms has been specifically designed to take advantage of the ability of the system to reach the sliding surface in a finite time. The high control activity seen in typical sliding controllers is avoided in this formulation, resulting in a guidance law that is both globally stable and robust against unknown, but bounded perturbations. The proposed MSSG does not require any off-line trajectory generation and therefore it is flexible enough to target a large variety of point on the planet's surface without the need for calculation of multiple reference trajectories. However, after initial analysis, it has been seen that the performance of MSSG is very sensitive to the choice in guidance gains. MSSG generated trajectories have been compared to an optimal solution to begin an investigation of the relationship between the optimality and performance of MSSG and the selection of the guidance parameters. A full study has been performed to investigate and tune the parameters of MSSG utilizing reinforcement learning in order to truly optimize the performance of the MSSG algorithm. Results show that the MSSG algorithm can indeed generate trajectories that come very close to the optimal solution in terms of fuel usage. A full comparison of the trajectories is included, as well as a further study examining the capability of the MSSG algorithm under perturbed conditions using the optimized set of parameters.

[*] Graduate Student, 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
[‡] Assistant Professor, Department of Systems and Industrial 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.

INTRODUCTION

Future planetary missions, both robotic and human, will require unprecedented levels of landing accuracy and system flexibility. Over the past decade, pinpoint landing on Mars has been gaining great importance in the community, and will continue to gain importance in the future due to the continued interest of the scientific community.[1,2] In addition, there has been recent renewed interest in the Moon and its potential economic returns by mining for the various resources that it contains.[3] In both cases, the continued interest in missions to specific locations on the planetary surface contributes to the need for more precise delivery of the vehicle. In past missions to both the Moon and Mars, mission success was ensured with a safe landing without the need to land the spacecraft precisely at the targeted location with little residual error. The landing accuracy, usually described by a 3-sigma landing ellipse, has been established to be very large, on the order of 100 km, for previous Mars missions. The recent landing of the Mars Science Laboratory (MSL) has taken important steps towards increasing the precision of landing on Mars by utilizing bank angle control during the initial atmospheric entry and by using the newly designed “Sky Crane” system.[4] Despite these improvements, future missions may require an even higher level of landing precision. In addition, the accuracy for lunar landing can also be improved as few vehicles have successfully landed on the moon since the days of the Apollo missions, which were human-guided.

Powered descent algorithms generally comprise of two major components: a) a targeting (guidance) algorithm and b) a trajectory-following, real-time guidance algorithm. The targeting algorithms is responsible for generating a reference trajectory (position, velocity, and thrust profile) that explicitly defines the path for driving the vehicle to the desired landing location. Subsequently, the trajectory-tracking algorithm is designed to close the loops on the desired trajectory ensuring that the spacecraft follows the planned path. 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[5-7], the method is currently employed to generate a feasible reference trajectory comprising of the three segments of the MSL powered descent phase.[8] A fifth-order (minimal) polynomial in time satisfies the boundary condition for each of the three position components. The required coefficients can be determined analytically as a function of the pre-determined time-to-go.

Recently, more research efforts have been devoted towards determining reference trajectories (and guidance commands) that are fuel-optimal, i.e. trajectories that satisfy both the desired boundary conditions and any additional constraints while minimizing the fuel usage.[9-11] For such cases, analytical solutions are possible only for the energy-optimal landing problem with unconstrained thrust.[12] To the best of our knowledge, closed-form 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.[13] Recently, Acikmese et al.[11] devised a convex optimization approach where the minimum-fuel soft landing problem is cast as a Second Order Cone Programming problem (SOCP[14]). The authors showed that the appropriate choice of a slack variable can convexify the problem.[15] Consequently, the resulting optimal problem can be solved in polynomial time using interior-point method algorithms.[16] 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.[17]

Despite these advancements in trajectory-generating algorithms for on-board determination of minimum-fuel flyable trajectories, such algorithms require a significant amount of real-time computation and are very dependent on the designed reference trajectory. In this paper, we present a method for generating real-time, closed loop, planetary powered descent trajectories that take advantage of the finite-time reaching phase of the sliding mode control.[18,19] The proposed algorithm has its theoretical foundation on the well-known sliding control theory as well as on the more recently developed Higher Order Sliding Control (HOSC) approach.[20-22] Sliding mode control has been recently employed to develop innovative and more robust algorithms for endo-atmospheric flight system guidance (e.g. missiles[23]). In particular, sliding control methods have emerged as attractive techniques that can be applied to develop robust missile autopilots[24,25] and guidance algorithms.[26,27] However, such non-linear guidance design methods have rarely been used to design guidance algorithms for planetary precision landing. Recently, Furfaro et al. have explored sliding control theory as a mean to develop two classes of robust guidance algorithms for both precision lunar and asteroid landing.[28-30] Here, we propose a Multiple Sliding Surface Guidance (MSSG) approach for power descent guidance that is robust against perturbations and unmodeled dynamics. MSSG, which is designed on the principles of 2-sliding mode control, employs multiple sliding surfaces to generate on-line targeting trajectories that are guaranteed to be globally stable under bounded perturbations (with known upper bound).[18,21] Two sliding surface vectors are concatenated in such a way that an acceleration command program that drives the second surface to zero automatically drives the dynamical system on the first surface in a finite time. The on-line trajectory generation and the determination of the guidance command require only knowledge of the system state (position and velocity) and the desired landing position. Importantly, one of the key principles behind the proposed methodology is that the landing problem is considered complete once the sliding surface is reached, i.e. the dynamical system reaches the surface for the first time at the landing location (with the desired velocity). Clearly, chattering on the first surface is avoided because the system does not need to slide along it. Such approach has been first proposed and discussed by Harl and Balakrishnan who applied HOSC to design a class of sliding-based guidance algorithms for the terminal guidance of an unpowered lifting vehicle during the approach and landing phase.[31]

However, while the MSSG has been shown to be robust and globally stable in the previous work done by Furfaro et al.[28-30], preliminary results have demonstrated that it is sensitive to the guidance parameters and is generally sub-optimal for any given set of parameters. This paper aims to investigate the behavior of the guidance gains and to utilize reinforcement learning to select proper parameter values in order to optimize MSSG in terms of both residual guidance error and fuel usage when applied to the lunar landing problem.

GUIDANCE PROBLEM FORMULATION

We consider the planetary descent and 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:

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

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

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

Here, $\boldsymbol{r}_L$ and $\boldsymbol{v}_L$ are the position and velocity of the lander with respect to a coordinate system with origin on the planet's surface, $\boldsymbol{g}(\boldsymbol{r_L})$ 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 $\boldsymbol{r}_L = [x, y, z]^T$ and $\boldsymbol{v_L} = \left[v_x, v_y, v_z\right]^T$ the equations of motion can be written by components as:

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

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

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

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

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

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

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.

NON-LINEAR LANDING GUIDANCE LAW DEVELOPMENT

Sliding Control Theory

The sliding control methodology is an elementary approach to robust control.[32] Intuitively, it is based on the observation that it is much easier to control non-linear and uncertain 1st order systems (i.e. described by 1st order differential equations) than nth-order systems (i.e. described by nth-order differential equations). Generally, if a transformation is found such that an nth-order problem can be replaced by a 1st order problem, it can be shown that, for the transformed problem, perfect performance can be in principle achieved in presence of parameter inaccuracy. As a drawback, such performance is generally obtained at the price of higher control activity.

Consider the following single-input nth-order dynamical system:

$$\frac{d^n}{dt^n} x = f(\boldsymbol{x}) + b(\boldsymbol{x}) u \tag{10}$$

Here, x is the scalar output, u is the control variable and $\boldsymbol{x} = \left[x, \dot{x}, \ldots\ldots, x^{(n)}\right]^{\boldsymbol{T}}$ is the state vector. Both $f(\boldsymbol{x})$, which describes the non-linear system dynamics, and the control gain $b(\boldsymbol{x})$ are not exactly known. Assuming that both $f(\boldsymbol{x})$ and $b(\boldsymbol{x})$ have a known upper bound, the sliding control goal is to get the state $\boldsymbol{x}$ to track the desired state $\boldsymbol{x_d} = \left[x_d, \dot{x}_d, \ldots\ldots, x_d^{(n)}\right]^{\boldsymbol{T}}$ in presence of model uncertainties. The time varying sliding surface is introduced as a function of the tracking error $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x_d}$ by the following scalar equation:

$$s(\boldsymbol{x}, t) = (\frac{d}{dt} + \lambda)^{n-1}\tilde{\boldsymbol{x}} = 0 \tag{11}$$

For example, if $n = 2$ we obtain:

$$s(\boldsymbol{x}, t) = \dot{\tilde{\boldsymbol{x}}} + \lambda\tilde{\boldsymbol{x}} = 0 \tag{12}$$

With the definitions in Eq. (11) and Eq. (12), the tracking problem is reduced to the problem of forcing the dynamical system in Eq. (10) to remain on the time-varying sliding surface. Clearly, tracking an n-dimensional vector $\boldsymbol{x_d}$ has been reduced to the problem of keeping the scalar sliding surface to zero, i.e. the problem has been reduced to a 1st order stabilization problem in s. The simplified 1st order stabilization problem can be now achieved by selecting a control law such that outside the sliding surface the following is satisfied:

$$\frac{1}{2}\frac{d}{dt}s^2 \le -\eta|s| \tag{13}$$

Here, η is a strictly positive constant. Eq. (13), also called the "sliding condition", explicitly states that the distance from the sliding surface decreases along all system trajectories. Generally, constructing a control law that satisfies the sliding condition is fairly straightforward. For example, using the Lyapunov direct method one can select a candidate Lyapunov function as follows:

$$V(s) = \frac{1}{2}s^T s \tag{14}$$

with $V(0) = 0$ and $V(s) > 0$ for $s > 0$. By taking the derivative of Eq. (14), it is easily concluded that the sliding condition (Eq. (13)) is satisfied. The control law is generally obtained by substituting the sliding control definition, Eq. (12), and the system dynamical equations, Eq. (10), into Eq. (13).

Multiple sliding surface control approach to the Design of Planetary Landing Guidance

The overall approach to the development of multiple sliding surface guidance and control algorithms is to apply the notion that the motion of the guided spacecraft during descent and landing exists in a second order sliding mode. The following definitions clarify this concept:

Definition: Consider a smooth dynamical system with a smooth output $s(\boldsymbol{x})$, called a sliding function. Then, provided that $s, \dot{s}, \ddot{s}, \ldots, s^{r-1}$ are continuous and that $s = \dot{s} = \ddot{s} = \cdots = s^{r-1} = 0$, then the motion on the set $\{s, \dot{s}, \ddot{s}, \ldots, s^{r-1}\} = \{0,0,0, \ldots, 0\}$ is said to exist on an r-order sliding mode.

For a class of sliding surfaces that are of interest to the landing guidance problem, the dynamics of the system are such that the sliding surfaces are of order two. The first surface is defined as the vector difference between the current position and the desired position. The second surface is defined as the vector difference between the current and desired velocity. Now let us define the first sliding vector surface in the following way:

$$\boldsymbol{s}_1 = \boldsymbol{r_L} - \boldsymbol{r_d} \tag{15}$$

Here, $\boldsymbol{r_d}$ is the desired target landing point on the lunar surface. Taking the derivative of $\boldsymbol{s}_1$ we obtain:

$$\dot{\boldsymbol{s}}_1 = \boldsymbol{v}_L - \boldsymbol{v}_d \tag{16}$$

where $\boldsymbol{v}_d$ is the desired landing velocity (set to zero for soft landing). The guidance problem can now be set as a control problem where the acceleration command is found such that $\boldsymbol{s_1} \to \boldsymbol{0}$ and $\dot{\boldsymbol{s}}_\mathbf{1} \to \boldsymbol{0}$ in a finite time t_f. It is easy to verify that the sliding surface is of relative degree two, as the acceleration command appears in the second derivative of the sliding surface. Indeed, the second derivative of $\boldsymbol{s_1}$ is:

$$\ddot{\boldsymbol{s}}_1 = \dot{\boldsymbol{v}}_L = -\boldsymbol{g}(\boldsymbol{r}) + \frac{\boldsymbol{T}}{m_L} \tag{17}$$

Using a back-stepping approach, $\dot{\boldsymbol{s}}_1$ may be employed as a virtual controller to ensure that the system is driven to zero in a finite time.

$$\dot{\boldsymbol{s}}_1 = -\frac{\boldsymbol{\Lambda}}{(t_f - t)}\boldsymbol{s}_1 \tag{18}$$

where $\boldsymbol{\Lambda} = diag(\Lambda_1, \Lambda_2, \Lambda_3)$ is a diagonal matrix of guidance gains (all set to be positive and greater than one). The system is shown to be globally stable and reaches the vector surface $\boldsymbol{s}_1$ in a finite time under the specified conditions on $\boldsymbol{\Lambda}$ via a Lyapunov approach. A Lyapunov candidate function can be defined as follows:

$$V_1 = \frac{1}{2}\boldsymbol{s}_1^T\boldsymbol{s}_1 \to \dot{V}_1 = \boldsymbol{s}_1^T\dot{\boldsymbol{s}}_1 = -\frac{1}{(t_f - t)}\boldsymbol{s}_1^T\boldsymbol{\Lambda}\boldsymbol{s}_1 = -\frac{1}{\left(t_f - t\right)}\left(\Lambda_1 s_{11}^2 + \Lambda_2 s_{12}^2 + \Lambda_3 s_{13}^2\right) < 0 \tag{19}$$

Defined as a quadratic function, the Lyapunov candidate meets the first three criteria required to be a true Lyapunov function, i.e. V_1 has the following properties:

$$V_1(\boldsymbol{0}) = 0 \quad if\ \boldsymbol{s}_1 = \boldsymbol{0} \tag{20}$$

$$V_1(\boldsymbol{s}_1) > 0 \quad if\ \boldsymbol{s}_1 \neq \boldsymbol{0} \tag{21}$$

$$V_1(\boldsymbol{s}_1) \to \infty \quad if\ \boldsymbol{s}_1 \to \infty \tag{22}$$

In addition to these criteria, the time-derivative of V_1 must be negative along system trajectories. Eq. (19) shows that by imposing positive gains $\{\Lambda_1, \Lambda_2, \Lambda_3\} > 0$, the time derivative of V_1 can be made negative definite everywhere. However, it is generally desirable for the matrix gains to all be greater than one to ensure the finite time convergence of both the sliding surface and its derivative to zero. Indeed, the time variation of the surface $\boldsymbol{s}_1$ can be explicitly derived as a function of the gains. Applying separation of variables to Eq. (18), one obtains:

$$\frac{ds_{1_i}}{s_{1_i}} = -\frac{\Lambda_i dt}{t_f - t} \tag{23}$$

where $i = 1,2,3$ are the components of the sliding surface vector. Eq. (23) can be integrated to obtain:

$$ln(s_{1i}) = \Lambda_i\, ln\left(t_f - t\right) + C_i \tag{24}$$

By imposing the initial conditions $\boldsymbol{s}_1(0) = \boldsymbol{s}_{10}$ and taking the exponential of both sides, the solution becomes:

$$s_{1i}(t) = s_{10i}\left(t_f - t\right)^{\Lambda_i} \tag{25}$$

or in vector form:

$$\boldsymbol{s}_1(t) = \boldsymbol{s}_{10}\left(t_f - t\right)^{\Lambda} \tag{26}$$

The derivative of $\boldsymbol{s}_1$ can also be expressed explicitly as:

$$\dot{s}_{1i}(t) = \Lambda_i s_{10i}\left(t_f - t\right)^{\Lambda_i - 1} \tag{27}$$

or in vector form:

$$\dot{\boldsymbol{s}}_1(t) = \Lambda \boldsymbol{s}_{10}\left(t_f - t\right)^{\Lambda - I} \tag{28}$$

As can be seen in Eq. 25, if $\Lambda_i > 0, (i = 1,2,3)$, the sliding surface vector will reach 0 in a finite time. However, if $\Lambda_i < 1, (i = 1,2,3)$, the sliding surface derivative blows up as $t \to t_f$ as seen in Eq. (27). Therefore, if the matrix gains are selected such that $\Lambda_i > 1, (i = 1,2,3)$, both sliding surface and its derivative will approach zero as $t \to t_f$.

Clearly, when the descent toward the surface begins, the initial conditions are not such that Eq. (18) is generally satisfied. A meaningful guidance law requires that $\dot{\boldsymbol{s}}_1$ to be explicitly linked to the acceleration command that is executed to drive both $\boldsymbol{s}_1$ and $\dot{\boldsymbol{s}}_1$ to zero. To provide this critical link, a second sliding surface is defined as:

$$\boldsymbol{s}_2 = \dot{\boldsymbol{s}}_1 + \frac{\Lambda}{(t_f - t)}\boldsymbol{s}_1 = \boldsymbol{0} \tag{29}$$

The new sliding surface vector $\boldsymbol{s}_2$ has relative degree of one with respect to the control input. Indeed, it can be easily verified that the thrust command appears explicitly in the first derivative of $\boldsymbol{s}_2$:

$$\dot{\boldsymbol{s}}_2 = \ddot{\boldsymbol{s}}_1 + \frac{\boldsymbol{\Lambda}}{(t_f - t)}\dot{\boldsymbol{s}}_1 + \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)^2}\boldsymbol{s}_1 \tag{30}$$

Using Eq. (17), it is found that:

$$\dot{\boldsymbol{s}}_2 = -\boldsymbol{g}(\boldsymbol{r}) + \frac{\boldsymbol{T}}{m_L} + \boldsymbol{p} + \frac{\boldsymbol{\Lambda}}{(t_f - t)}\dot{\boldsymbol{s}}_1 + \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)^2}\boldsymbol{s}_1 \tag{31}$$

The desired thrust command is determined through the use of Lyapunov's second method. A quadratic form of the second sliding surface is chosen as the candidate Lyapunov function:

$$V_2 = \frac{1}{2}\boldsymbol{s}_2^T\boldsymbol{s}_2 \tag{32}$$

The acceleration command is obtained by differentiating V_2 with respect to time:

$$\dot{V}_2 = \boldsymbol{s}_2^T\dot{\boldsymbol{s}}_2 = \boldsymbol{s}_2^T\left(-\boldsymbol{g}(\boldsymbol{r}) + \frac{\boldsymbol{T}}{m_L} + \boldsymbol{p} + \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)}\dot{\boldsymbol{s}}_1 + \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)^2}\boldsymbol{s}_1\right) \tag{33}$$

If the commanded acceleration vector is chosen as follows:

$$\boldsymbol{a}_c = \frac{\boldsymbol{T}}{m_L} = \boldsymbol{g}(\boldsymbol{r}) - \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)}\dot{\boldsymbol{s}}_1 - \frac{\boldsymbol{\Lambda}}{\left(t_f - t\right)^2}\boldsymbol{s}_1 - \Phi\mathrm{sign}(\boldsymbol{s}_2) \tag{34}$$

The time derivative of the Lyapunov candidate becomes:

$$\dot{V}_2 = \boldsymbol{s}_2^T\left(\boldsymbol{p} - \Phi sign(\boldsymbol{s}_2)\right) \tag{35}$$

If an estimation of the upper bound on the perturbing forces is available, the coefficients of the matrix $\Phi = diag(\Phi_1, \Phi_2, \Phi_3)$ (gains) can be selected to guarantee the derivative of V_2 is always less than zero along all system trajectories experienced during the powered descent. Consequently, the control law is shown to be globally stable. If the gains are selected as:

$$\Phi_{\mathrm{i}} = \frac{\left|s_{2_i}(0)\right|}{t_r^*} \tag{36}$$

the surface $\boldsymbol{s}_2$ is guaranteed to converge to zero in a finite time $t_r^* < t_f$. Eq. (34) is the guidance law that has been named the Multiple Sliding Surface Guidance (MSSG) law.

Markov Decision Process and Reinforcement Learning

Sometimes called approximate dynamic programming, Reinforcement Learning (RL) is a machine learning technique concerned with how an agent (e.g. the lander) must take actions in uncertain environments to maximize a cumulative reward (e.g. minimize the landing errors, minimizing fuel consumption). The stochastic environment is conventionally formulated as a Markov Decision Process (MDP) where the transition between states for a given action is modeled using appropriate transition probability. A MDP consists of:

- A set of states S (where s denotes a state $s \in S$)
- A set of actions A (where a denotes an action $a \in A$)
- A reward function $R(s) \rightarrow \Re$ that maps a state (or possibly a state-action pair) to the set of real numbers
- State transition probabilities, which defines the probability distribution over the new state $s' \in S$ that will be transitioned to given action a is taken while in state s
- An optional discount rate that is typically used for infinite horizon problems

RL algorithms can learn a policy $\pi(s): S \mapsto A$ that maps each state to an optimal action. For a given state, these actions are considered optimal if they maximize the policy's utility over the resulting system's trajectory. The utility is defined as the expected value of the sum of discounted rewards computed starting from state s_0 and following the policy π. For a single trajectory i, one can write the following:

$$U^{(i)}(\pi) = E\left[R\left(s_0^{(i)}\right) + \gamma R\left(s_1^{(i)}\right) + \gamma^2 R\left(s_2^{(i)}\right) + \cdots\right] \tag{37}$$

Here, the expectation accounts for the stochastic nature of the environment, whereas the sequence $s_0^{(i)}, s_1^{(i)}, s_2^{(i)}, \ldots$ defines the trajectory associated with starting from initial condition i.

To conceptually illustrate how the RL theory can be applied to design adaptive controllers, consider the problem of stabilizing an inverted pendulum attached to a cart on a bounded track. From a control design point of view, for any possible starting location on the track, it is desired to generate an acceleration command that keeps the angle of the pendulum with the vertical axis less than some specified target value. Any possible starting location will define a unique trajectory - possibly over an infinite horizon, unless the controller finds a steady state control that keeps the pendulum balanced. For the cart control problem and given the current system state, reinforcement learning means that a policy that will generate an optimal action (i.e. an acceleration along the track) is learned through real or simulated experience. The action will be determined to be optimal with respect to the problem's definition of utility. For this type of problem, one can estimate a policy's utility using a sampling approach. In such a case, the estimate would be calculated as either the average or minimum utility over the sample trajectories:

$$U(\pi) = \frac{1}{N}\sum_{i=1}^{N} U^{(i)}(\pi) \qquad \text{or} \qquad U(\pi) = \min_i U^{(i)}(\pi) \tag{38}$$

Note that the concepts used to define the reinforcement learning problem have a clear counterpart to those used to define the optimal control problem: the policy becomes the controller, the action becomes the control, the state transition probabilities becomes the plant, the utility becomes the negative cost, while the state remains the same.

Many RL-based algorithms approach the problem of determining the optimal policy indirectly, i.e. through the value function. For a given policy π, the value function is defined as the expected sum of rewards given that the system is in state s and executes policy π. The value function can be expressed in terms of the reward associated with being in the current state plus the discounted expectation over the distribution of the next states given that the controller is following policy π:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \tag{39}$$

The optimal value function, $V^{*}(s)$ is generally found using dynamic programming algorithms, e.g. value iteration. Once the optimal value function is determined, the optimal policy is determined by taking the action that maximizes the expected sum of future rewards for the given current state; using Equation (39), the optimally policy can be formally expressed as follows:

$$\pi^{*}(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^{*}(s') \tag{40}$$

The principal limitation of this approach is that both the state and action space must be discretized. As a result, the number of discrete states and actions grow exponentially with the dimensionality of the state and action spaces. Often the state space has higher dimensionality than the action space; consequently it is often easier to discretize the action space than the state space. When this is the case, fitted value iteration, which learns a model that represents the value function as a continuous function of the state, may be effectively employed to determine the optimal policy. Suitable value approximation functions for fitted value iteration include least squares, weighted least squares, and neural networks using on-line learning. When it is desired to work with continuous state and action spaces, one approach (policy optimization) is to dispense with the value function, and directly search for the optimal policy.

RESULTS AND ANALYSIS

Initial Comparison to Optimal Solution

As shown in Eq. (34), the guidance algorithms depends on five guidance parameters: t_f, t_f^{*}, and $\mathbf{\Lambda} = diag(\Lambda_1, \Lambda_2, \Lambda_3)$. By examining these parameters, the behavior of the MSSG-generated algorithms can be assessed.

In order to investigate the effect of these parameters, the MSSG guidance algorithm is compared from a fuel-usage standpoint to an open-loop, fuel-optimal landing guidance solution found numerically via pseudo-spectral methods for the problem of lunar landing. The minimum-fuel optimal guidance problem can be formulated as follows:[11]

Minimum-Fuel Problem: 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):

$$\max_{t_F,\mathbf{T}(\cdot)} m_L(t_F) = \min_{t_F,\mathbf{T}(\cdot)} \int_0^{t_F} \|\mathbf{T}\|\, dt \tag{41}$$

Subject to the following constraints (equations of motion):

$$\ddot{\boldsymbol{r}}_L = -\boldsymbol{g_L} + \frac{\boldsymbol{T}}{m_L} \tag{42}$$

$$\frac{d}{dt} m_L = -\frac{\|\boldsymbol{T}\|}{I_{sp} g_0} \tag{43}$$

and the following boundary conditions and additional constraints:

$$0 < T_{min} < \|\boldsymbol{T}\| < T_{max} \tag{44}$$

$$\boldsymbol{r}_L(0) = \boldsymbol{r}_{L0}, \quad \boldsymbol{v}_L(0) = \dot{\boldsymbol{r}}_L(0) = \boldsymbol{v}_{L0} \tag{45}$$

$$\boldsymbol{r}_L(t_F) = \boldsymbol{r}_{LF}, \quad \boldsymbol{v}_L(t_F) = \dot{\boldsymbol{r}}_L(t_F) = \boldsymbol{v}_{LF} \tag{46}$$

$$m_L(0) = m_{Lwet} \tag{47}$$

Here, the thrust is limited to operate between a minimum value (T_{min}) and a maximum value (T_{max}). In this formulation, the lunar gravity, $\boldsymbol{g_L}$ is considered to be constant over the range of altitudes used for the terminal descent problem. The problem formulated in Eq. (41-47) does not have an analytical solution and must be solved numerically. To obtain the open-loop, fuel-optimal thrust program, the General Pseudospectral Optimal Control Software (GPOPS[33]) has been employed. GPOPS is an open-source optimal control software that implements Gauss and Radau hp-adaptive pseudospectral methods. After formulating the landing problem as described above, the software allows the direct transcription of the continuous-time, fuel-optimal control problem to a finite-dimensional Nonlinear Programming Problem (NLP). In GPOPS, the resulting NLP is solved using the SNOPT solver.[34] The pseudospectral approach is very powerful as it allows one to approximate both state and control using a basis of lagrange polynomials. Moreover, the dynamics is collocated at the Legendre-Gauss-Radau points. The use of global polynomials coupled with Gauss quadrature collocation points is known to provide accurate approximations that converge exponentially to continuous problems with smooth solutions.

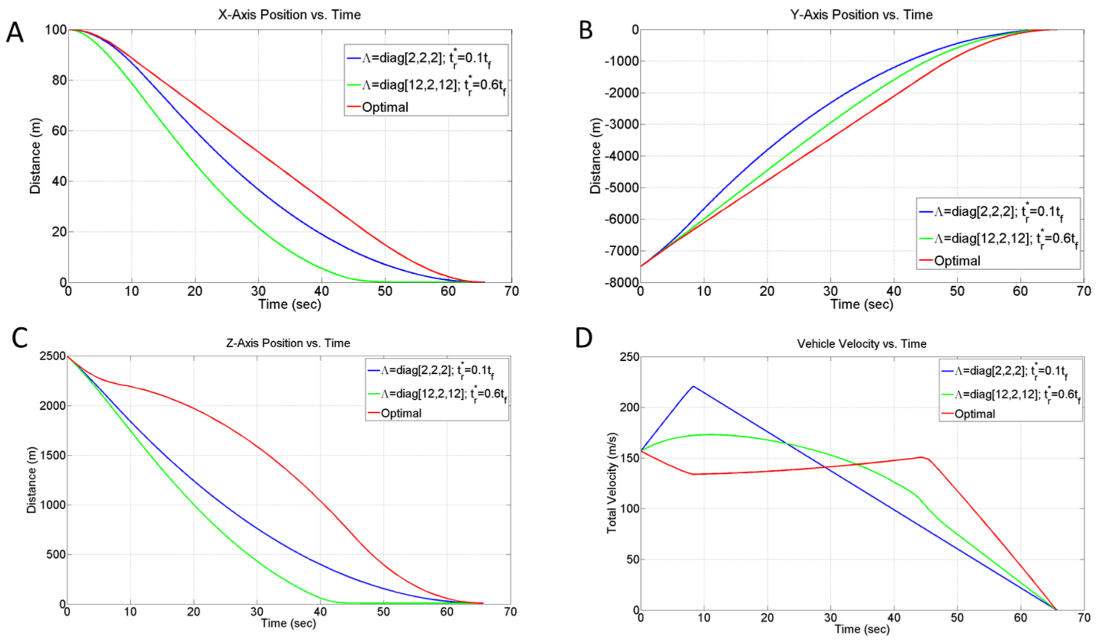

Figure 1. Comparison of MSSG System Trajectories to Optimal Solution

The open-loop, fuel-optimal landing problem is solved assuming that the lander's initial position and velocity are $\boldsymbol{r}_L(0) = [100, -7500,\ 2500]\,m$ and $\boldsymbol{v}_L(0) = [0, 145, -60\,]\,m/sec$, respectively. The guidance reference frame is fixed to the lunar surface with the origin located at the targeted landing point. The desired final state of the vehicle is set to be $\boldsymbol{r}_L(t_f) = [0,0,10]m$ and $\boldsymbol{v}_L(t_f) = [0,0,0]m/sec$. 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, as seen in Eq. (42). The lander is assumed to weigh 1900 kg (wet mass) and is capable of a maximum (allowable) thrust of 15 kN as well as a minimum (allowable) thrust of 1.5 N. While this lower thrust limit may not be truly realistic, it provides an extremely idealistic optimal case with which we can compare to the MSSG trajectories. The terminal position is set to be at the origin of the guidance reference frame to be achieved with zero velocity (soft landing). For comparison, the MSSG algorithm is initiated at the same initial conditions and defined to target the same terminal state. Figure 1 shows the trajectories and total velocity histories for the open-loop, fuel-optimal landing guidance found via GPOPS and two MSSG-guided cases. GPOPS found a total optimal flight time equal to $65.6233\ sec$ which is also assumed to be the t_F employed by the MSSG guidance scheme. The MSSG guidance gains are set to be $\boldsymbol{\Lambda} = diag(2,2,2)$ and $t_r^* = .1t_f$ for the first MSSG simulation and $\boldsymbol{\Lambda} = diag(12,2,12)$ and $t_r^* = 0.6t_f$ for the second MSSG simulation. These values were chosen based on previous simulations and previous papers[28-30], the knowledge that convergence is guaranteed for values of $\Lambda_i > 1$ (Eq. 28), as well as to show the range of the parameter space and how their selection alters the behavior of the guidance law. In both cases presented, the guidance updates at a frequency of 10 Hz. Figure 2 shows the behavior of the thrust magnitude and the total lander mass as a function of time for the three cases presented. A comparison between the mass of propellant employed by the three algorithms is reported in Table 1. Notably, it can be seen that the MSSG trajectory thrust profiles do not produce values near the lower thrust limit imposed on the optimal solution, but instead feature a minimum thrust limit of approximately 3000 N. For the same descent time, the MSSG algorithm tends to require more fuel mass than the optimal case. The fuel-efficient thrust profile is extremal, i.e. the optimal algorithm thrusts at a maximum value until it switches to the minimum allowable thrust, and finally returns to the maximum value (bang-bang type). The MSSG algorithms reduce the thrust command until the second surface is reached. At $t_r^* = nt_F$, the thrust command experiences a large shift and then decreases monotonically until the first surface is achieved. The thrust spike can be in principle reduced by increasing t_F.

Table 1. Comparison Between the Open-Loop Fuel-Optimal Guidance and the MSSG Guidance (Propellant Mass)

	Optimal (GPOPS)	**MSSG** $\boldsymbol{\Lambda = diag\{2,2,2\}}$ $\boldsymbol{t_r^* = 0.1t_f}$	**MSSG** $\boldsymbol{\Lambda = diag\{12,2,12\}}$ $\boldsymbol{t_r^* = 0.6t_f}$
Mass of Propellant/Optimal Value	1	1.36	1.16

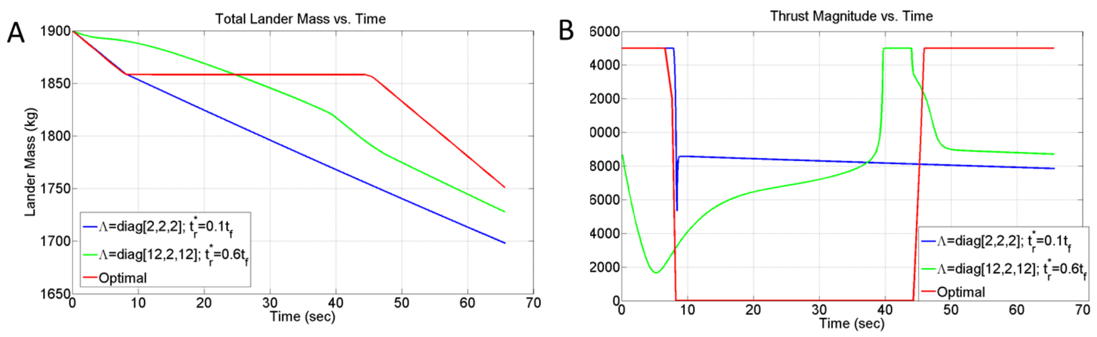

Figure 2. A) Lander Mass and B) Thrust Magnitude Histories Comparing Optimal Solution to MSSG

Clearly, MSSG is sub-optimal. Indeed, in order to better characterize the performance and to find the optimal guidance gains for MSSG, reinforcement learning is implemented. By using reinforcement learning policies to determine a set of optimal guidance gains, the performance of MSSG can truly be optimized in terms of both fuel optimality and landing accuracy.

Parameter Optimization via Reinforcement Learning

In order to explore the potential optimality of the MSSG algorithm, reinforcement learning has been employed to determine an optimal set of guidance gains. The vehicle state $[\boldsymbol{r}, \boldsymbol{v}]$ at the start of the powered descent phase (i.e. $t = 0$) is used for the policy optimization for this particular application and the parameters that were to be learned by this technique are the guidance parameters $\boldsymbol{\Lambda} = diag(\Lambda_1, \Lambda_2, \Lambda_3)$, the final time of the descent t_f, and the time of convergence of the second sliding surface, t_r^*. Indeed, these parameters make up the five-dimensional action space for this problem. By utilizing the initial vehicle state, a set of these five parameters was chosen and used for the entirety of that scenario, i.e. only one set of parameters are used and the parameters do not update adaptively during the course of the simulation.

Importantly, by utilizing reinforcement learning, the guidance algorithm is optimized in a stochastic environment. That is, the guidance parameters are being learned to be optimal over a range of starting condition and also account for sensor or navigation errors. By including these in the learning process, the resulting parameters will not only provide optimality from a fuel consumption standpoint, they will also include robustness to initial state perturbations and sensor errors, which is not accounted for in the given optimal trajectory seen in Fig. 1 and 2. The reinforcement learning simulation used to calculate the utility of a certain combination of parameters uses conditions which are very similar to those provided in the previous section. The initial wet mass of the spacecraft remains 1900 kg. The initial conditions of the simulation used to learn the parameters are distributed deterministically according to the values in Table 2. Each of these values are uniformly distributed between the respective minimum and maximum values, providing a total of 144 samples that are used to learn an optimal set of parameters. Further, these simulations included uncertainties of $1\ m$ and $0.5\ m/sec$ in position and velocity, respectively, which are provided to the guidance algorithm to simulate sensor noise when estimating the current state of the spacecraft. Finally, the utility function, which is determined at the end of the simulation of each sample, which is used in order to determine which set of parameters is optimal, was chosen to be:

$$U^{(i)} = a\left\|\boldsymbol{r}_f - \boldsymbol{r}_d\right\|^2 + b\left\|\boldsymbol{v}_f - \boldsymbol{v}_d\right\|^2 + c\left(m_f - m_d\right)^2 \tag{48}$$

Here, the terms $\left\|\boldsymbol{r}_f - \boldsymbol{r}_d\right\|$ and $\left\|\boldsymbol{v}_f - \boldsymbol{v}_d\right\|$ represent the final position and velocity errors that result from the chosen guidance parameters, $m_f - m_d$ represents the difference between the final lander mass resulting from the chosen parameters and the final mass observed from the optimal solution, and $a, b,$ and c represent weights for each of their respective terms. Essentially, this utility function is attempting to minimize the residual error of the guidance law and minimizing the fuel usage over the given set of initial conditions.

Table 2. Deterministic Initial Conditions for Reinforcement Learning Parameter Optimization

Initial Condition	**Minimum Value**	**Maximum Value**
X-Axis Position	$-100\ m$	$100\ m$
Y-Axis Position	$-8000\ m$	$-7000\ m$
Z-Axis Position	$2250\ m$	$2750\ m$
X-Axis Velocity	$-10\ m/sec$	$10\ m/sec$
Y-Axis Velocity	$140\ m/sec$	$160\ m/sec$
Z-Axis Velocity	$-70\ m/sec$	$-50\ m/sec$

The results of this optimization are captured in Table 3. Importantly, the final time of the simulation t_f determined by reinforcement learning is very similar to the value determined by the optimal solution, $t_{f_{optimal}} = 65.6233\ sec$.

Table 3. Optimal Parameter Results of Reinforcement Learning

$\boldsymbol{\Lambda_1}$	$\boldsymbol{\Lambda_2}$	$\boldsymbol{\Lambda_3}$	$\boldsymbol{t_f}$	$\boldsymbol{t_r^*}$
$\mathbf{2.8518}$	1.9021	1.8215	$65.2324\ sec$	$34.0957\ sec$

These results have been further validated by inputting the resulting parameters into the MSSG simulation environment used to produce Fig. 1 and 2 for direct comparison to the optimal solution, which can be found in Fig. 3 and 4.

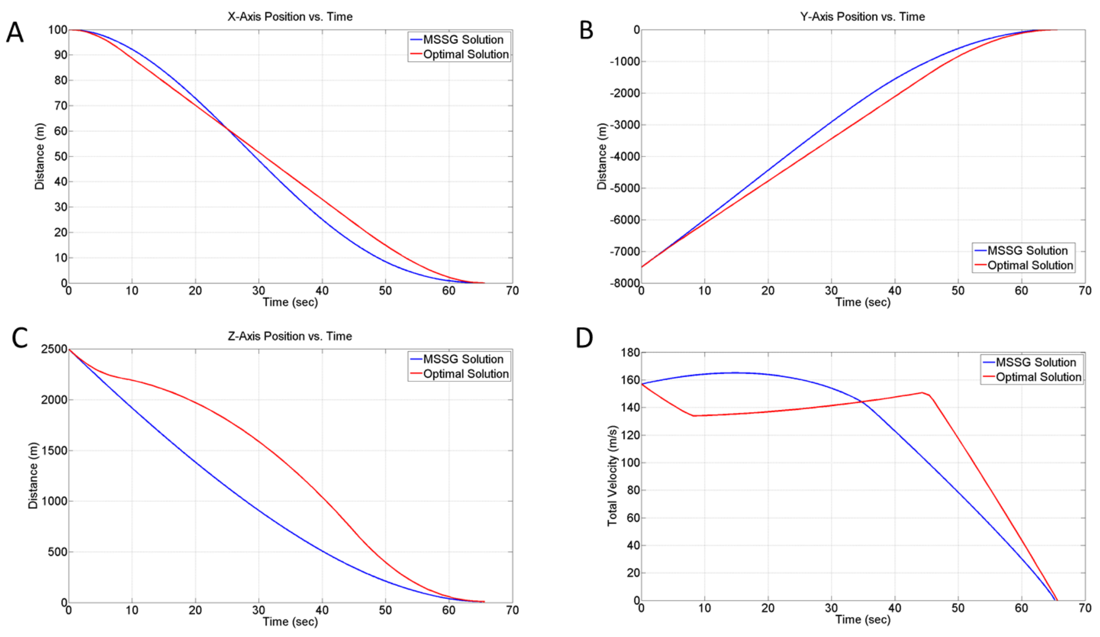

Figure 3. Comparison of Reinforcement Learning Optimized MSSG System Trajectories to Optimal Solution

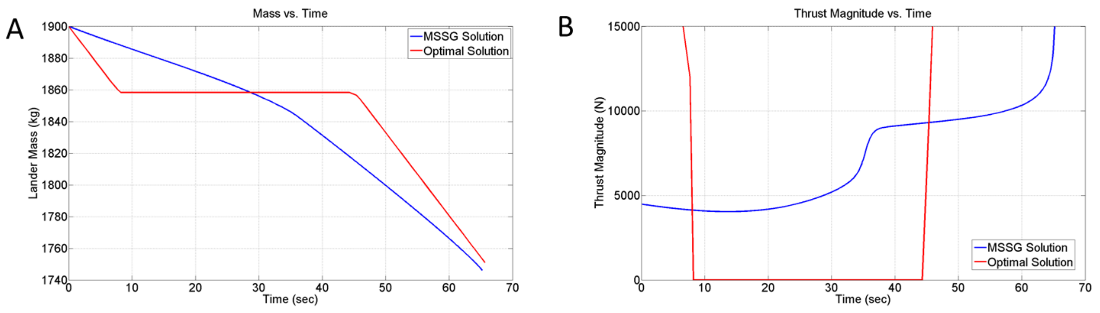

Figure 4. A) Thrust Magnitude and B) Lander Mass Histories Comparing Optimal Solution to Reinforcement Learning Optimized MSSG

Figure 3 shows a comparison of the open-loop optimal trajectory and the reinforcement learning optimized MSSG closed-loop trajectories, and a comparison of the velocity magnitude histories of the two results. As expected, both results bring the spacecraft to the desired target point with minimal residual velocity. Figure 4 compares both the fuel consumption and the thrust magnitude histories of the two solutions. While the optimized result still provides a suboptimal solution, the fuel consumption of the MSSG-generated trajectory is less than 4% off of the optimal, as shown explicitly in Table 3. Figure 4 also shows that the MSSG solution has an approximate minimum thrust level of more than 4000 N, which is much larger than the typical constraint of a minimum thrust level of $0.2T_{max}$. Importantly, the fuel consumption demonstrated by the MSSG guidance is very close to the optimal open-loop solution, but as a real guidance solution, it also accounts for closing the guidance loop, something that is not included in the open-loop optimal solution, which is included here simply as a reference for comparison. Further, the inherent robustness of the MSSG algorithm allows it to be more flexible than the provided deterministic solution, providing the possibility of retargeting and optimization about off-nominal conditions.

Table 4. Comparison Between the Open-Loop Fuel-Optimal Guidance and the Reinforcement Learning Optimized MSSG Guidance (Propellant Mass)

	Optimal (GPOPS)	**RL Optimized MSSG**
Mass of Propellant/Optimal Value	1	1.032
Mass of Propellant	$148.87\ kg$	$153.65\ kg$

Monte Carlo Analysis of Optimized MSSG

In order to further validate and analyze the robustness and optimality of the learned MSSG parameters, a Monte Carlo analysis has been performed to test the system under off-nominal conditions. The Monte Carlo analysis has been conducted by running 1000 simulations of the MSSG algorithm in the described 3-DOF framework. Table 5 shows the parameters used in the simulations, as well as their dispersions. All values are assumed to follow a normal (Gaussian) distribution described by their respective means and standard deviations. The MSSG parameters and gains are exactly those described in Table 3 and are the result of the Reinforcement Learning optimization. The nominal case for the Monte Carlo simulation is the same as seen in the previous analysis. Further, for each thruster firing, the mass-flow rate is perturbed using a normal distribution which is set assuming an upper value of 10% deviation from the mean value.

Table 5. Initial Condition Dispersions for Monte Carlo Simulation

Initial Condition	**Mean Value**	**Standard Deviation**
X-Axis Position	$0\ m$	$100\ m$
Y-Axis Position	$-7500\ m$	$-100m$
Z-Axis Position	$2500\ m$	$100\ m$
X-Axis Velocity	$0\ m/sec$	$10\ m/sec$
Y-Axis Velocity	$145\ m/sec$	$10\ m/sec$
Z-Axis Velocity	$-60\ m/sec$	$-10\ m/sec$
Navigation Error – Position	N/A	$1\ m$
Navigation Error - Velocity	N/A	$0.1\ m$

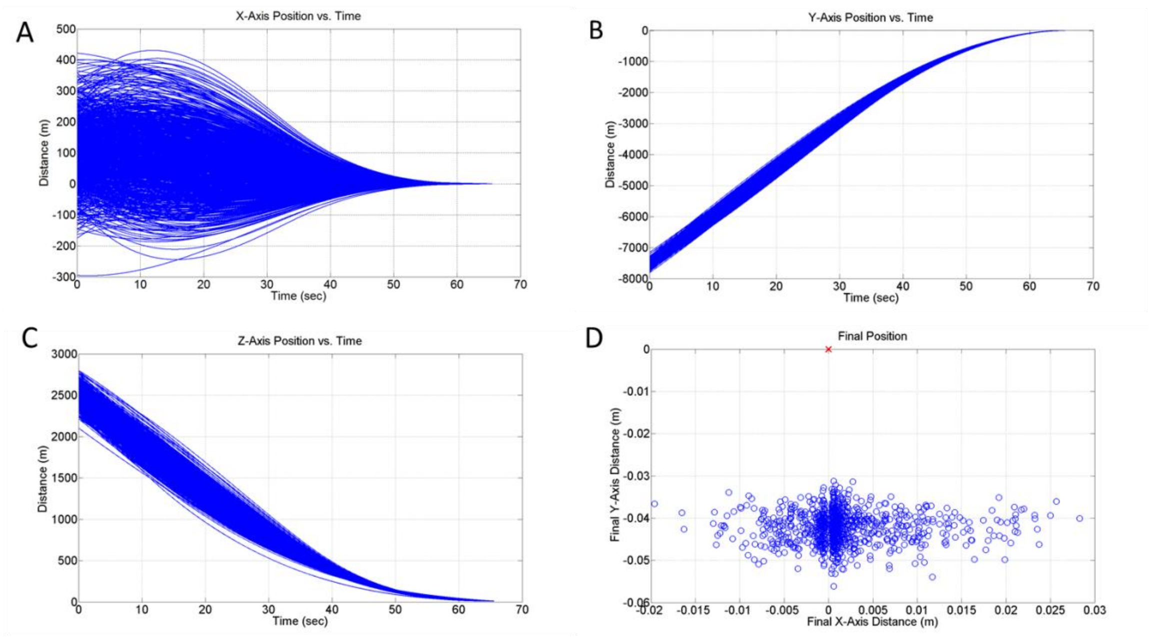

Figure 5. Monte Carlo Simulation Results: A) X-Axis Position; B) Y-Axis Position; C) Z-Axis Position; and D) Final Landing Location Dispersion

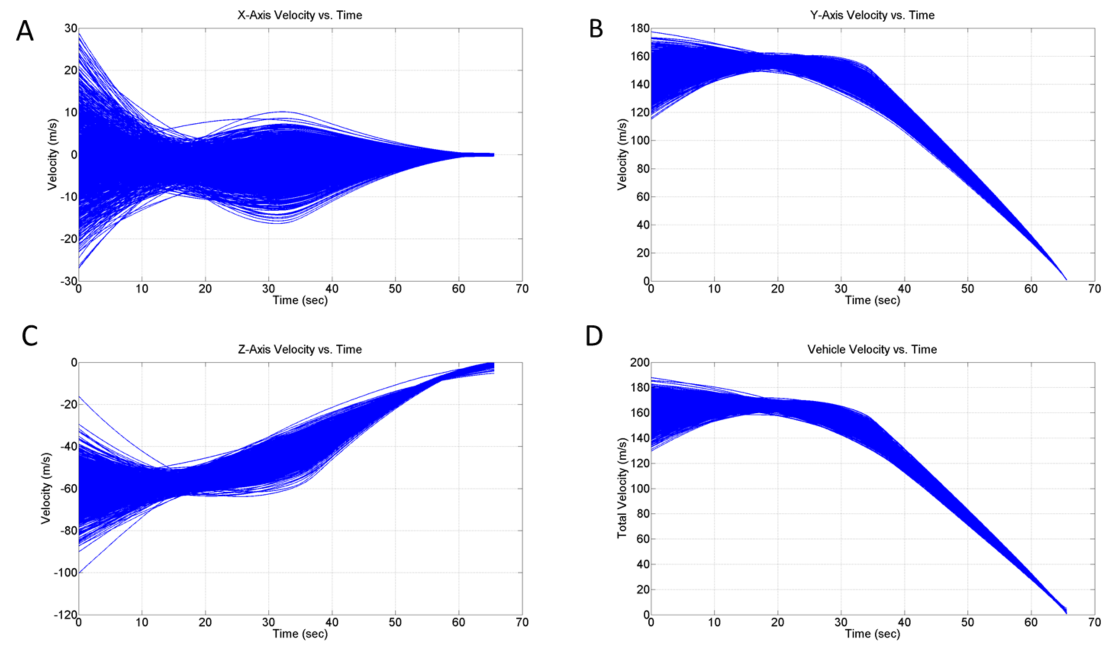

Figure 6. Monte Carlo Simulation Results: A) X-Axis Velocity; B) Y-Axis Velocity; C) Z-Axis Velocity; and D) Velocity Magnitude

Figures 5 and 6 show the position and velocity histories for the set of 1000 Monte Carlo simulations. The terminal state statistics for position and velocity are reported in Fig. 7 and 8. Additionally, Fig. 9 shows the mass fuel consumption and thrust command histories for the 1000 Monte Carlo cases. These results show that not only is the chosen set of gains very near optimal from a fuel consumption perspective, but that they are also robust to perturbations and off nominal conditions. Figures 7 and 8 shows that the residual errors in position and velocity are very near zero. The results of the Monte Carlo simulation are gathered in Table 6.

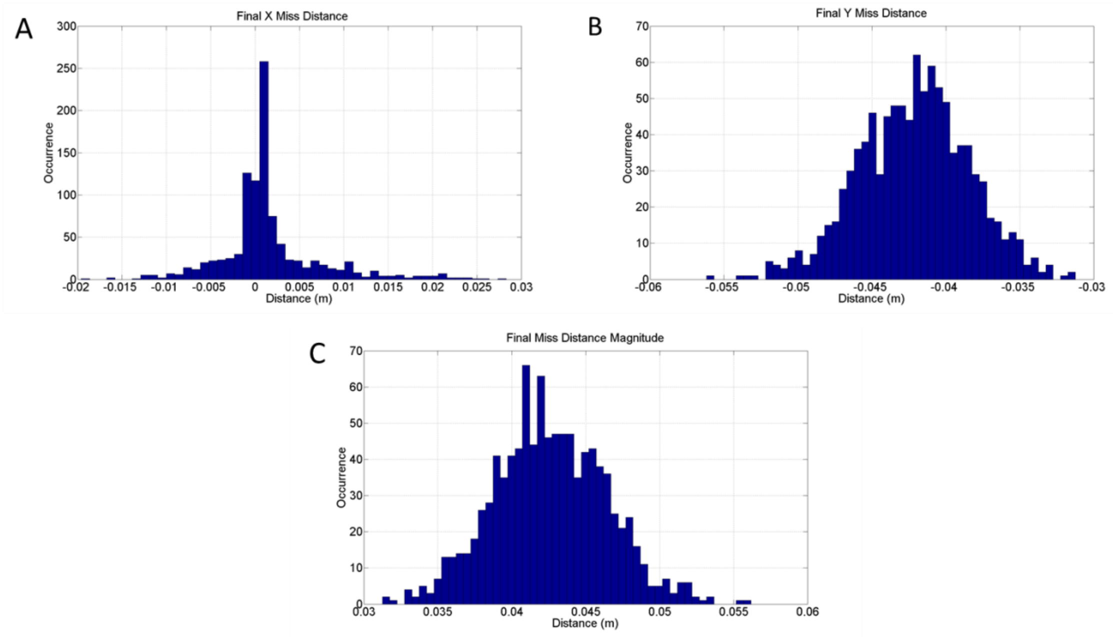

Figure 7. Monte Carlo Simulation Results: A) X-Axis Miss Distance; B) Y-Axis Miss Distance; and C) Magnitude of Miss Distance

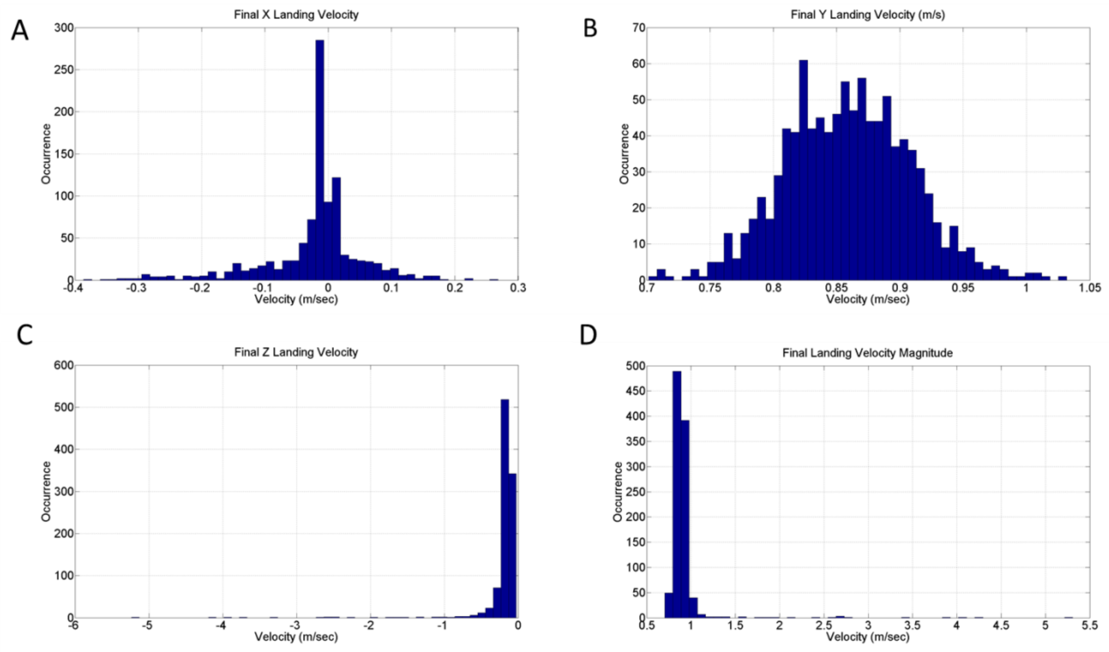

Figure 8. Monte Carlo Simulation Results: A) Residual X-Axis Velocity Error; B) Residual Y-Axis Velocity Error; C) Residual Z-Axis Velocity Error; and D) Magnitude of Residual Velocity Error

Table 6. Monte Carlo Simulation Result Statistics

Initial Condition	Mean Final Value	Standard Deviation
X-Axis Position	$0.0016\,m$	$0.0057\,m$
Y-Axis Position	$-0.0422\,m$	$0.0037\,m$
Z-Axis Position	$10\,m$	$0\,m$
X-Axis Velocity	$-0.0215\,m/sec$	$0.0787\,m/sec$
Y-Axis Velocity	$0.8590\,m/sec$	$0.0498\,m/sec$
Z-Axis Velocity	$-0.2144\,m/sec$	$0.3649\,m/sec$

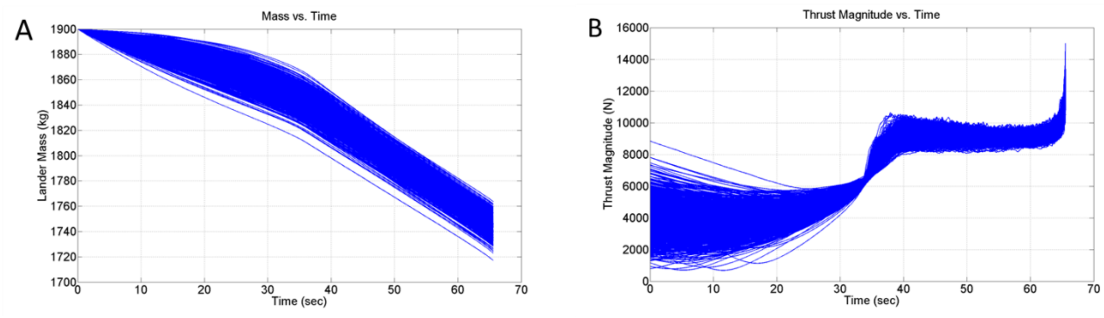

Figure 9. Monte Carlo Simulation Results: A) Lander Mass History; and B) Thrust Magnitude History

Clearly the performance of the algorithm with the given parameters is quite good. Figure 5d shows that the final miss distance is very near zero for all cases, demonstrating the accuracy of the algorithm with the optimized parameters. However, as can be seen in Fig. 8, and Table 6, some cases did contain significant residual velocity error. The residual final velocity has a maximum value of $5.3\,m/sec$, which is mostly in the $-Z$ direction, as seen in Fig. 8c. However, most of the cases exhibit residual velocity errors much smaller than this. Further, this error can be attributed in part due to the amount of training that was performed within the reinforcement learning environment, and also that the MSSG parameters were determined for the nominal initial condition as opposed to a set of parameters that work best over the entire space defining all possible initial conditions. It is likely that this case was very near the edge of the design space that was used in the learning environment, and as such further training will likely remove these errors.

CONCLUSIONS

A novel non-linear guidance algorithm using Higher Order Sliding Mode (HOSM) control is presented and analyzed. In previous work, the algorithm has been theoretically proven to be globally stable under the condition that an upped bound for the perturbing forces is known, and has been shown to be both robust and accurate under perturbations and unmodeled dynamics. This algorithm has also been previously examined from a fuel consumption perspective where it has been shown that, with any given set of guidance parameters, the MSSG algorithm is clearly sub-optimal. In order to examine the optimality of the MSSG algorithm, a machine learning technique known as reinforcement learning has been employed to find a set of parameters that not only include the performance and robustness inherent to the MSSG algorithm, but also optimizes them in terms of fuel consumption. Indeed, the results of the reinforcement learning process has provided a set of parameters that bring the performance of the MSSG algorithm very close to that provided by an optimal open-loop solution. Further, the solution does not require the generation of a reference trajectory, and as such is more flexible under off-nominal conditions and perturbations. The features of accuracy, flexibility, and good fuel usage make the MSSG algorithm very applicable for future lunar missions.

Future work for the MSSG algorithm include a) inclusion of additional constraints in the reinforcement learning process, including guide-slope constraints, b) shifting the paradigm of one set of guidance parameters to that of a more adaptive approach, which will allow the reinforcement learning algorithm to learn an optimal set of guidance parameters that update during the descent phase, and c) the application of reinforcement learning of the MSSG algorithm to other scenarios, including asteroid proximity operations and Mars hypersonic reentry and terminal powered landing guidance. These additional features and analysis will allow the MSSG algorithm to be further refined and analyzed under more strenuous conditions and provide further understanding into the true application of the algorithm for future mission architectures.

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] Phinney, W.C., Criswell, D., Drexler, E., Garmirian, J. "Lunar Resources and Their Utilization", *Space-Based Manufacturing from Nonterrestial Materials*, AIAA p. 97-123, 1977.

[4] 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).

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

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

[17] 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.

[18] A., Levant, Construction Principles of 2-Sliding Mode Design, Automatica, Vol. 43, No. 4, (2007), pp. 576–586.

[19] Levant, A., Sliding order and sliding accuracy in sliding mode control. *International Journal of Control*, *58*(6), 1993, 1247–1263.

[20] A. Levant, Higher-order sliding modes, differentiation and output feedback control. *International Journal of Control*, *76*(9/10), (2003), 924–941.

[21] Levant, A. Homogeneity approach to high-order sliding mode design. *Automatica*, *41*(5), (2005a). 823–830.

[22] Levant, A., Quasi-continuous high-order sliding-mode controllers. *IEEE Transactions on Automatic Control*, *50*(11), (2005b), 1812–1816.

[23] Y. B., Shtessel, & I. A., Shkolnikov, Aeronautical and space vehicle control in dynamic sliding manifolds. International Journal of Control, 76(9/10), 1000–1017, 2003.

[24] M., U., Salamci, M., K., Ozgoren, S., P., Banks, Sliding Mode Control with Optimal Sliding Surfaces for Missile Autopilot Design, *Journal of Guidance, Control and Dynamics*, Vol. 23, No. 4, 2000.

[25] Y., Shtessel, and C., Tournes, Integrated Higher-Order Sliding Mode Guidance and Autopilot for Dual-Control Missiles, *Journal of Guidance, Control and Dynamics*, Vol. 32, No. 1, 2009.

[26] C., Tournes, Y., Shtessel, I., Shkolnikov, Missile Controlled by Lift and Divert Thrusters Using Nonlinear Dynamic Sliding Manifolds, *Journal of Guidance, Control and Dynamics*, Vol. 29, No. 3, 2006.

[27] A., Koren, M., Idan, O., M., Golan, Integrated Mode Guidance and Control for a Missile with On-Off Actuators, *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 1, 2008.

[28] R. Furfaro, S. Selnick, M. L. Cupples and M. W. Cribb, Non-Linear Sliding Guidance Algorithms for Precision Lunar Landing, in Advances in the Astronautical Sciences, Volume 140, Proceedings of the 21st AAS/AIAA Space Flight Mechanics Meeting held February 13-17, 2011, New Orleans, Louisiana.

[29] R. Furfaro and Daniel R. Wibben, Asteroid Precision Landing Via Multiple Sliding Surfaces Guidance Techniques, Proceedings of the annual AAS/AIAA Astrodynamics Specialists Conference, AAS 11-647, July 31-August 4, 2011, Girdwood, Alaska.

[30] R. Furfaro, D. O., Cersosimo and J. Bellerose, Close Proximity Asteroid Operations using Sliding Control Modes, Proceedings of the annual AAS/AIAA Space Flight Mechanics Conference, AAS 12-132, Jan 31-Feb 4, 2012, Charleston, Louisiana.

[31] N., Harl, and S., N., Balakrishnan, Reentry Terminal Guidance Through Sliding Control Mode, Journal of Guidance, Control and Dynamics, Vol. 33, No. 1, January–February 2010.

[32] Slotine, J., and Li, W., Applied Nonlinear Control, Prentice Hall, 1991.

[33] 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.

[34] P.E. Gill, M.A. Saunders, and W. Murray. SNOPT: An SQP algorithm for large scale constrained optimization. Technical Report NA 96-2, University of California, San Diego, 1996.