Reachability-Based Forced Landing System

Anayo K. Akametalu and Claire J. Tomlin

University of California, Berkeley, CA 94720

and

Mo Chen

Simon Fraser University, Burnaby, BC V5A 1S6, Canada

DOI: 10.2514/1.G003490

A major hurdle toward the integration of unmanned aerial systems into the civilian airspace is the development of a principled methodology for handling emergency landings. Most of the prior work in the area of emergency landings for unmanned aerial systems has been concerned with using computer vision to identify potential landing sites. However, reaching these sites may not be dynamically feasible, and the maneuver needed to reach these sites may not be obvious. In this paper, a reachability-based forced landing system is proposed that uses Hamilton–Jacobi–Bellman reachability to determine the feasible landing region of a distressed aircraft. The utility of this technique is displayed on a fixed-wing aircraft with engine failure. In addition, it is also shown how to synthesize a controller that guides the aircraft to any desired landing location inside the feasible landing region.

Nomenclature

\[ B = \text{backward reachable set} \]
\[ C = \text{constraint set (general)} \]
\[ C_D = \text{drag coefficient} \]
\[ C_L = \text{lift coefficient} \]
\[ D = \text{force, } \text{N} \]
\[ \mathcal{F} = \text{forward reachable set} \]
\[ \mathcal{G}_s = \text{ground auxiliary state constraint set} \]
\[ \mathcal{G}_p = \text{ground pose constraint set} \]
\[ g = \text{gravitational constant, m/s}^2 \]
\[ \mathcal{T} = \text{initial set} \]
\[ L = \text{lift force, N} \]
\[ m = \text{mass, kg} \]
\[ \Phi = \text{aircraft pose} \]
\[ q, s = \text{state vector (general, not specific to aircraft)} \]
\[ S_w = \text{wing surface area, m}^2 \]
\[ T = \text{target set} \]
\[ v = \text{aircraft speed, m/s} \]
\[ X_s = \text{auxiliary state constraint set} \]
\[ X_p = \text{pose constraint set} \]
\[ \mathcal{Y} = \text{terminal set} \]
\[ (x, y, z) = \text{aircraft position (Earth-fixed frame), m} \]
\[ Z = \text{feasible landing set} \]
\[ \alpha = \text{angle of attack, deg} \]
\[ \gamma = \text{aircraft vertical flight-path angle, deg} \]
\[ \eta = \text{auxiliary state} \]
\[ \kappa = \text{optimal minimum time to reach feedback controller} \]
\[ \xi = \text{aircraft state} \]
\[ \phi = \text{roll angle, deg} \]
\[ \psi = \text{aircraft heading (azimuth), deg} \]

On 15 January 2009, Captain Chesley B. “Sully” Sullenberger emergency landed US Airways Flight 1549, a fixed-wing passenger aircraft, in the Hudson River after both jet engines failed due to multiple bird strikes. Remarkably, all 155 people onboard the aircraft survived [1]. The event has since been referred to as the “Miracle on the Hudson” and is widely regarded as one of the greatest emergency landings in history.

The coming years will witness large-scale incorporation of unmanned aerial systems (UASs) into the civilian airspace [2–5]. Because of the large variety of commercial applications, such as aerial photography, farming, package delivery, and provision of internet services, many key agencies and corporations are pushing for the Federal Aviation Administration (FAA) to adopt this technology into the civilian airspace [6]. As the FAA develops the regulations for UASs, one of the challenges is equipping these systems with the ability to handle emergency scenarios where a forced landing is inevitable [2].

Broadsely speaking, a forced landing procedure can be broken down into four tasks. The first task is to identify the fault that has caused the forced landing. A fault may significantly alter the dynamics of the UAS, for example, an engine failure leading to a loss of thrust. The second task is to determine feasible landing locations, which are the locations the aircraft can dynamically reach given its current state [7]. It is crucial that the second task be performed before choosing a landing location. Obtaining feasible regions limits the search space when choosing the landing location and prevents the aircraft from landing in locations that are not reachable. The third task is to choose a desired landing location, which can be chosen to minimize (maximize) a cost (utility). Finally, the aircraft must be guided to the desired location.

In manned aircraft, the responsibility of these tasks falls primarily on the pilot with some aid from the automation. For UASs, intelligent algorithms will be required to automate the entire process. In [8], the authors show how to automate tasks two, three, and four for the forced landing of a quadrotor with reduced thrust capability (descending with constant acceleration) using ideas from optimal control and statistical inference. In the last decade, there has been much work in developing the individual technologies necessary to execute an autonomous forced landing.

In regard to the third task, offline and online approaches for identifying suitable landing sites have been proposed. The works in [9, 10] focus on identifying feasible landing sites for a specified landing zone, while [11] focuses on identifying feasible landing sites for a specified landing zone with a specified landing zone resolution.
[9, 10] leverage databases to construct cost maps from terrain, population, and structure data. Others have proposed using computer vision to determine suitable landing sites from aerial imagery in real time [11–13]. In [11], an image is segmented into regions using a Canny edge detector [14], and these regions are classified based on five factors: size, shape, slope, surface, and surroundings. These factors plus an additional factor related to civilization are referred to as the “six Ss”, and they can be used with linguistic rules to identify candidate landing sites. There is also ongoing work for direct classification of landing sites using neural networks [13], which has also been applied to the fault detection problem (task one). In [15], the neural networks are trained to learn functions that map errors in the desired angular velocities to faults. The work in [15] is improved upon in [16] by using the augmented control, instead of the angular velocity errors, as the input to the neural net.

The fourth task is a trajectory planning problem, which has been well studied in the control and robotics communities. There are a number of techniques that have been proposed to solve such problems including optimal control [17], model predictive control [18], A* search [2], and rapidly expanding random trees [19]. Despite these recent developments, little work has been done toward the third task of finding the feasible landing locations. We stress here that it is important to have access to the entire feasible landing region, so as to make the best decision on where to land the aircraft. Access to the feasible landing region allows the automation to minimize the search space when determining the landing location, and it also guarantees that the aircraft will land in the best location possible. In [17], the authors propose a landing procedure that maximizes the aircraft’s altitude to maximize the feasible landing region. However, this region is never computed explicitly nor is the feasibility of individual landing locations ever determined. In [20], the authors develop a technique to determine the reachability of specific runways, given the glide range and the descent path of a fixed-wing aircraft with a failed engine, which implicitly requires that ideal landing targets (i.e., runways) are known a priori and their reachability be ascertained online. In the worst case, there may be no ideal landing site that is reachable, but the aircraft still needs to make the best landing possible, which requires knowledge of all possible locations the aircraft can be guided to. After assigning a cost (utility) to each feasible location, the location with the minimum (maximum) value would be considered the best landing location.

Finding the feasible landing region can be thought of as a reachability problem, where the task is to find the set of states that can reach or be reached from some target. In this paper, we will employ a reachability technique from the optimal control literature known as Hamilton–Jacobi–Bellman (HJB) reachability analysis. The technique is very general and can be applied to a wide range of dynamical systems. In this model, the reachability problem is a six-degree-of-freedom fixed-wing aircraft with a failed engine. The techniques we propose here can also handle constraints on the system under consideration (e.g., limits on the angle of attack or aircraft speed can easily be incorporated). Last, we will show how the same HJB techniques can be used to synthesize feedback control laws that guide the aircraft to a desired landing location.

Typically, HJB techniques are very powerful for low-dimensional systems. Various aircraft models and faults can be handled as long as we have a dynamical model describing the aircraft motion under the given fault. However, because HJB reachability requires gridding the state space, it becomes intractable in higher dimensions because computational time and memory storage scale exponentially with the system dimensionality. Recent theoretical advances in reachability for systems with lightly coupled dynamics may reduce the computational burden by decomposing high-dimensional problems into a collection of low-dimensional problems [21, 22]. In the general case, it makes practical sense to compute the feasible landing set offline to avoid the computational bottleneck. However, the feasible landing set depends on the initial configuration (position, speed, heading, etc.) of the aircraft at the onset of the forced landing, which is not known a priori. Fortunately, under certain assumptions, only the feasible landing set of a nominal initial aircraft configuration needs to be computed as a template, and it may be done so offline. The feasible landing set for arbitrary configurations can then be obtained by taking Euclidean transformations of this template. To further enhance tractability, the aircraft’s movements are restricted to specific motion primitives, which can be described by lower-dimensional models, thus improving computation time and storage. These primitives can be composed one after another, and we can relax the reachability problem to finding the feasible lands that can be achieved by transitioning through a particular sequence of primitives.

The main contribution of this paper is in demonstrating the use of HJB reachability analysis for computing the feasible landing region and landing control for a fixed-wing aircraft with loss of thrust. In particular, we demonstrate how constraints can be incorporated in this framework to ensure safe landings. HJB reachability has been applied to emergency landing scenarios for quadrotors [8], but this is the first time it has been used for fixed-wing aircraft. The large state-space dimension of the fixed-wing aircraft presents some novel challenges that are also addressed here.

The rest of the paper is organized as follows. In Sec. II, we describe the model and constraints of the aircraft. Section III formulates our two tasks of interest: finding a forced landing set and synthesizing a landing controller as reachability problems. In Sec. IV, we address the practical challenges that arise from the computational complexity of the reachability-based forced landing system (RBFLS), and finally in Sec. V, we present a practical method to handle the two tasks of interest before ending with our results and conclusion. The aircraft model and its parameters were obtained through a collaboration with NASA Neil A. Armstrong Flight Research Center.

Remark 1: Functions will be defined to accept both elements and sets as arguments. For example, define a function g(·): X → Y, which takes a point in X and maps it to a point in Y. In an abuse of notation, we also use the same function g to map between subsets of X and Y, g(·): X ⊆ X2 → Y. Furthermore, the set-to-set mapping g is always defined based on the point-to-point mapping
g(XS) = {g(x)|x ∈ XS}
(1)
where the argument XS is a subset of X.

II. Aircraft Model

UASs are broadly categorized as rotary aircraft or fixed-wing aircraft, each with plethora of possible faults. To focus the exposition, we present the methods of this paper around an example of a fixed-wing aircraft with an engine failure, although RBFLS, theoretically, is general enough to handle other faults that can be modeled in the dynamic behavior of the aircraft. Fixed-wing aircraft are suitable for long-distance travel and package delivery because their efficient aerodynamical structure allows for longer flight times with greater payloads. Engine failure is an important fault to consider because it is a common cause of forced landings in general aviation [20]. In this section, we describe the model of the aircraft and the constraints that must be satisfied during the forced landing.

A. Aircraft Dynamics

Consider a right-handed coordinate system that coincides with the local east–north–up coordinates. We define the positive x axis to be east, the positive y axis to be north, and the positive z axis to be up.

We take the aircraft state to be x = [x, y, z, p, v, ϕ]T ∈ R6, where the states are the position vector (x, y, z), heading angle ϕ (azimuth), airspeed of the aircraft v, and vertical flight-path angle γ. For convenience, we define the pose p := [x, y, z] ∈ R3 and the auxiliary state n := [v, ϕ] ∈ R2; thus, x = [p, n]T. We also define two projection maps πp(·): R6 → R3, πn(ξ) = p and πn(ξ): R3 → R2, πn(ξ) = n, which project the aircraft state onto the pose and auxiliary state, respectively. With loss of thrust, the control input to the aircraft u = [u, ε, η]T ∈ U ⊆ R2 consists of the angle of attack (AOA) α and bank angle β. Here, the control constraint U is a compact set. Following [17], the state evolves according to ξ = f(ξ, u), where f: R6 × U → R6 is given by
\[
\mathbf{f}(\xi, u) = \begin{bmatrix}
  v \cos \gamma \cos \psi \\
  v \cos \gamma \sin \psi \\
  v \sin \gamma \\
  \frac{L(a, v) \sin \phi}{m v \cos \gamma} \\
  \frac{D(a, v) \sin \phi}{m v} - g \sin \gamma 
\end{bmatrix}
\]  
\[
\text{where } L(a, v) = \frac{1}{2} \rho S u^2 C_L(a), \quad D(a, v) = \frac{1}{2} \rho S u^2 C_D(a) \tag{3}
\]

Here, \(g\) is the universal gravitational constant, \(m\) is the mass of the aircraft, and \(L\) and \(D\) are the lift and drag forces, which are expressed as

\[
X(\gamma) = [x, y, z]^T \in \mathbb{R}^3 \text{ and auxiliary state constraint set } \Sigma \subseteq \mathbb{R}^4 \text{ is}
\]

\[
X_\rho := \{(x, y, z, \psi) \in \mathbb{R}^4 \mid z \geq h_{\text{min}}(x, y) \} \tag{10}
\]

The constraints introduced thus far must be satisfied at all times during the forced landing. In addition to these constraints, there are constraints that must be satisfied upon reaching the ground. For example, the aircraft should be descending when it reaches the ground \((\gamma < 0)\) but should not contact the ground at too steep a flight-path angle. The landing speed of the aircraft should also be bounded to reduce the impact damage to the aircraft. To model these additional constraints that must be satisfied when the aircraft lands, we introduce the ground auxiliary constraint set \(G_q \subseteq \mathcal{X}_q\) and ground pose constraint set \(G_p \subseteq \mathcal{X}_p^g\):

\[
G_q := \{(v, \gamma) \in \mathcal{X}_q \mid v \leq \bar{v}, \gamma \leq \gamma_0 \leq 0 \} \tag{11}
\]

\[
G_p := \{(x, y, z, \psi) \in \mathbb{R}^4 \mid z = h_{\text{min}}(x, y) \} \tag{12}
\]

The last constraint in Eq. (12) ensures that the aircraft is physically on the ground.

Any state constituting a feasible landing must lie in the set \(G_q \times G_p \subseteq \mathbb{R}^6\). In the next section, we will explain how reachability analysis can be used to compute the set of states corresponding to feasible landings. For a practical implementation of these techniques as it relates to the forced landing problem, we will make two assumptions:

1) Nonascent: The aircraft never ascends during the forced landing, \(\gamma = 0\).
2) Flat ground: The ground profile is flat, \(h_{\text{min}}(x, y) = h_{\text{flat}} \in \mathbb{R}\).

The nonascent assumption is reasonable, given that the aircraft is experiencing an engine-out failure. The flat-ground assumption is not as reasonable, but one can take variations in the ground height into account when assigning cost to the landing locations to encourage choosing locations that are flat. In Sec. III.A, we will show that, under certain conditions, these assumptions allow us to reuse reachability computations for different initial aircraft states, which means that we can perform the computations offline and adjust the results online based on the current aircraft state. This result will allow us to determine the feasible landing set, as opposed to performing an expensive reachability computation online. We elaborate on this further in Sec. III.A.

### III. Reachability Problem Formulation

There are two problems of interest in this paper. First, given the aircraft initial state, find all feasible landing locations that the aircraft can reach. We refer to this as a forward reachability problem; given a dynamical system and a set of initial states from which the dynamical system can start, forward reachability is concerned with determining the set of states that can be reached from this set of initial states via constraint-satisfying trajectories.

Second, given a desired landing location, find a controller that will guide the aircraft there. The second problem can be solved by obtaining the solution to an appropriate backward reachability problem. Backward reachability deals with the same constraints as forward reachability but is instead concerned with finding the set of states that can reach a terminal set of states. In the remainder of this
section, we formally introduce forward and backward reachability and present the Hamilton–Jacobi–Bellman method for solving these problems. These techniques are very general, allowing us to work with nonlinear dynamics and to account for a variety of constraints. Thus, other faults can be addressed, for example partial loss of thrust or degradation in the ability to roll or pitch the aircraft due to compromised actuators.

A. Forward Reachability

A landing location is specified by a desired pose \( p \) of the aircraft. From here on, we will talk about the feasible landing pose, which is synonymous with the feasible landing location that we have referred to thus far. Given a set of possible initial states \( I \subseteq \mathbb{R}^6 \), which may be a singleton, we want to find the set of feasible landing poses that the aircraft can be guided to, without first violating the constraints defined in Sec. II.B. We call this the feasible landing set of \( I \), \( Z(I) \subseteq \mathbb{R}^4 \).

To aid in the derivation/computation of the feasible landing set, we first define the forward reachable set (FRS) of \( I \), \( \mathcal{F}(I) \subseteq \mathbb{R}^6 \), which is the set of states that the aircraft can be guided to from \( I \) within a specified time horizon \( T \), without first violating any flight constraint, and is given by

\[
\mathcal{F}(I) = \{ w \in \mathbb{R}^6 \mid \exists \xi^0 \in I, \ t \in [0, T], \ u(\cdot) \in \mathbb{U}, \ \xi(0) = \xi^0, \ \forall s \in [0, t], \ (\xi(s) \in \mathcal{X}_p \times \mathcal{X}_v; \xi(t) = w) \} (13)
\]

As presently defined, the FRS does not depend on the terminal (landing) constraints \( \mathcal{G}_p \) and \( \mathcal{G}_v \). If we wanted to account for these constraints in the FRS, then the preceding definition would be modified to read \( w \in \mathcal{G}_p \times \mathcal{G}_v \) instead of \( w \in \mathbb{R}^6 \). However, we choose to ignore these constraints here because, as will show in Sec. IV.A, doing so is required if we want the shape and size of the FRS to be invariant to the poses of the states in \( I \). Having such an invariance would allow us to reuse a FRS for different initial sets under certain conditions. The invariance is only possible because of the assumptions made in Sec. II.B.

For the invariance to hold, we require that the FRS not depend explicitly on pose constraints, which means that the expression \( \forall s \in [0, t], \ (\xi(s) \in \mathcal{X}_p \times \mathcal{X}_v) \) in Eq. (13) is troublesome. Recall, however, that the set constraint \( \mathcal{X}_v \) is meant to ensure that the aircraft stays above ground. The constraint can be read as \( \forall s \in [0, t], \ z(s) \geq h_{\text{min}}(s), v(s) \). Given the flat ground assumption, this constraint reduces to \( \forall s \in [0, t], \ (s) \geq h_{\text{flat}} \). Furthermore, the nonascending aircraft assumption implies that \( \forall s \in [0, t], \ z(s) \geq h_{\text{flat}} \Leftrightarrow z(t) \geq h_{\text{flat}} \). Thus, we can make \( \mathcal{X}_p \) a terminal constraint (it only needs to be satisfied at \( t \)). However, there is already the terminal constraint \( \mathcal{G}_p \subseteq \mathcal{X}_p \), and so we can drop the \( \mathcal{X}_p \) constraint altogether and express the FRS as

\[
\mathcal{F}(I) = \{ w \in \mathbb{R}^6 \mid \exists \xi^0 \in I, \ t \in [0, T], \ u(\cdot) \in \mathbb{U}, \ \xi(0) = \xi^0 \}, \ \forall s \in [0, t], \ (\xi(s) \in \mathcal{X}_v; \xi(t) = w) \} (14)
\]

With the FRS in place, the feasible landing set is defined as

\[
Z(I) := \pi_p(\mathcal{F}(I) \cap (\mathcal{G}_p \times \mathcal{G}_v)) \subseteq \mathbb{R}^4 (15)
\]

The explicit dependence of the feasible landing set on the ground pose constraint implies that it will not inherit the invariance properties from the FRS. To resolve this, we also introduce the feasible landing stack \( S \):

\[
S(I) := \pi_p(\mathcal{F}(I) \cap (\mathbb{R}^4 \times \mathcal{G}_p)) \subseteq \mathbb{R}^4 (16)
\]

The set in Eq. (16) contains all of the poses that can be reached by the aircraft, while simultaneously satisfying the ground auxiliary constraint. We choose to refer to it as a stack because it ignores the ground pose constraint, and so it can be thought of as stacking all of the feasible landing sets for each ground level (i.e., changes over \( h_{\text{flat}} \)). The feasible landing stack inherits the same invariance properties as the FRS, making it reusable under certain scenarios. The feasible landing set can be easily recovered from the stack as

\[
Z(I) = S(I) \cap \mathcal{G}_p (17)
\]

The set relationships are visualized in Fig. 1. The feasible landing stack (not explicitly labeled) is the conic section formed by intersecting the FRS (blue cone) and ground auxiliary constraint (vertical green plane). For visualization purposes, the auxiliary state \( \eta \) has been projected onto the aircraft speed \( v \), and the pose \( p \) has been projected onto the \( x \) position.

The challenge now becomes to find the FRS \( \mathcal{F} \), after which \( S \) and \( Z \) can be obtained by simple set operations. As stated before, we will show later on that \( \mathcal{F} \) and \( S \) have invariance properties, which will ultimately improve the practicality of RBFLS for a fixed-wing aircraft with loss of thrust.

B. Backward Reachability

The backward reachable set (BRS) is the set of states from which a prespecified set of terminal states can be reached within a time horizon \( T \), without first violating state constraints. Denote a set of desired terminal states (possibly a singleton) as \( \mathcal{Y} \subseteq \mathbb{R}^6 \); the BRS of \( \mathcal{Y} \) is

\[
B(\mathcal{Y}) = \{ w \in \mathbb{R}^6 \mid \exists \xi^* \in \mathcal{Y}, \ t \in [0, T], \ u(\cdot), \xi(0) = w, \ \forall s \in [0, t], \ (\xi(s) \in \mathcal{X}_p \times \mathcal{X}_v; \xi(t) = \xi^*) \} (18)
\]

The set in Eq. (18) contains all the states that can be steered to the target set without first violating the constraints.

Backward reachability looks very similar to forward reachability, and in fact it is. Assuming that the dynamics are time-invariant, the forward reachability problem can be thought of as a backward reachability problem where the initial set becomes a terminal set and the system dynamics evolve backward in time instead of forward [23]. Thus, we can use the same tools to solve both problems for time-invariant systems.

C. Hamilton–Jacobi–Bellman Formulation

Reachability problems are common in the control theory literature, and there are a number of methods for solving them [24–27]. In particular, one such method involves solving a time-dependent HJB variational inequality. In this section, we will show how to use this
method to compute the backward and forward reachable sets [Eqs. (18) and (14), respectively]. We begin by presenting these methods generally, that is, not necessarily tied to this particular forced landing application.

First, consider a general state vector \( q \in \mathbb{R}^n \). In an abuse of notation, we assume that \( q \) evolves according to dynamics \( f(q, u) \), where \( u \in \mathcal{U} \) is the control. We introduce two sets, the target set \( T \subseteq \mathbb{R}^n \), which may either represent a terminal set (backward reachability) or initial set (forward reachability), and the constraint set \( C \subseteq \mathbb{R}^n \), in which the state trajectory must lie for all time. For example, given the reachable sets [Eqs. (18) and (14)], \( C \) would just be \( \mathbb{R}^3 \times X_q \). However, using formulation (13) of the FRS, the constraint set is \( X_q' \times X_q' \).

Define two continuous functions \( \phi_T : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( \phi_C : \mathbb{R}^n \rightarrow \mathbb{R} \), such that \( T = \{ q \in \mathbb{R}^n | \phi_T(q) \leq 0 \} \) and \( C = \{ q \in \mathbb{R}^n | \phi_C(q) \leq 0 \} \). These functions are typically chosen to be signed distance functions to their respective sets, e.g.,

\[
\phi_T(q) = \begin{cases} 
\inf_{\bar{q} \in T} |q - \bar{q}|, & q \in \mathbb{R}^n \setminus T \\
-\inf_{\bar{q} \in \mathbb{R}^n \setminus T} |q - \bar{q}|, & q \in T 
\end{cases} 
\]

which are continuous by construction.

For a given target and constraint set, the time-dependent HJB variational inequality for backward reachability is given as

\[
D_t V(q, t) = -\min_u \left[ \inf_{\tau \in T} D_q V(q, \tau) \cdot f(q, u) \right] \\
n.s.t. \ V(q, t) \geq \phi_T(q) \quad \forall \ t 
\]

with \( V(q, 0) = \phi_T(q) \), where \( D_t V(q, t) \) and \( D_q V(q, t) \) are the partial derivatives of \( V \) with respect to \( t \) and \( q \), respectively.

Define the value function \( V_T \in C(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \) as the viscosity solution to Eq. (20) for a given constraint set \( C \) and target \( T \). The backward reachable set can be obtained as the zero sublevel set of the viscosity solution [27]; therefore, if we are considering the aircraft forced landing problem, then

\[
B(T) = \{ q | V_T(q, -T) \leq 0 \} 
\]

Note that the time interval we consider here starts at \(-T\) and ends at 0, which does not affect the reachable set because the dynamics are time-invariant.

The interpretation of Eq. (20) is surprisingly intuitive. The value function \( V(\cdot, -T) \) is used to determine which states can reach the target within a given time horizon \( T \) by assigning them with a negative value. Obviously, the only states where the value function should be negative for \( \tau = 0 \) are those in the target set, and this is achieved by initializing the value function to \( \phi_T(q) \). On the right-hand side (RHS) of Eq. (20), the control \( u \) at each state is chosen to align the dynamics in the direction of greatest decrease in the value function (i.e., the vector field is being directed to point toward the target set as much as possible given the dynamics). Increasing the horizon corresponds to decreasing \( t \) in the variational inequality, and the more a state can align its dynamics with the negative gradient (i.e., toward the target set), the more its value will decrease for decreasing \( t \). Furthermore, the minimum with zero ensures that the value function is nonincreasing for decreasing \( t \) (increasing the horizon), and so once a state can reach the target within some time horizon \( t_1 \) (has a negative value), it will be marked as reachable for all horizons \( \tau \geq t_1 \) (it will retain a negative value). Last, the constraint ensures that states not in the constraint set will always be positive in the value function and thus never be included in the BRS.

Furthermore, the value function can also be used for controller synthesis. The optimal (minimum time to reach the target) feedback controller to steer the system to \( T \) while satisfying \( C \) is given by

\[
k(\cdot; T, C) = \arg \min_{V_T(q, -T) \leq 0} \left[ \inf_{\tau \in T} D_q V_T(q, \tau) \cdot f(q, u) \right] \\
\]

The variational equality in Eq. (20) is formulated to specifically solve a backward reachability problem, but it can also be used for forward reachability with time-invariant dynamics. There are other variational equalities and partial differential equations that can be used for forward reachability and do not require time-invariant dynamics [28–30]. The forward reachability problem must first be converted to a backward reachability problem, which is done with two simple modifications; the initial set now becomes a terminal set, and the dynamics now evolve backward, such that \( f(q, u) \) is replaced with \(-f(q, u)\). Taking the resulting solution for the forward reachability problem to be \( U_T \), the \( r \)-horizon forward reachable set for the aircraft in \( \mathbb{R}^6 \) is

\[
F(T) = \{ q | U_T(q, -T) \leq 0 \} 
\]

Obtaining the sets of interest hinges on solving Eq. (20), which is typically done numerically by gridding the state space and computing numerical gradients. The finer the gridding is, the more accurate the solution will be [27]. Unfortunately, for an increasing level of accuracy, the number of grid points needed increases exponentially in the dimension of the system state, resulting in a computational burden both in time and storage. For example, obtaining a solution can take on the order of hours for a four-dimensional system in MATLAB and is considered to be computationally intractable for a system with five or more dimensions, unless the system has a special decomposable form [21, 22, 31].

To employ HJB reachability analysis, a model of the system is required, which also includes environmental factors like wind. Any conclusions drawn from the analysis (e.g., the feasible landing stack) are dependent on the particular model being used. Unfortunately, obtaining an exact model is not always possible, though we assume that it is in this paper. However, in most cases, we may have a nominal model with some uncertainty over its parameters (e.g., lift and drag coefficients). If the uncertainty is bounded, a robust assessment of the system can be performed using Hamilton–Jacobi–Isaacs reachability analysis [27], which treats the uncertainty as a worst-case disturbance.

IV. Addressing Practical Challenges

For a practical forced landing solution, the reachable sets should be available within seconds of being requested. As stated earlier, computing reachable sets can be costly, and this rules out the possibility of online computation. For arbitrary system dynamics, offline computation cannot be generalized to many initial conditions; thus, we would be forced to compute a reachable set for every possible initial condition of the aircraft at the time of fault (which is infinite!). Another issue is storing all of the reachable sets. Fortunately, for the dynamics given by Eq. (2), the resulting reachable sets have some invariance properties that allow them to be reused for a wide class of initial conditions, thus addressing the first challenge.

Next, to compute one reachable set offline still presents a challenge. The system state is six-dimensional, and depending on the desired accuracy of the solution (fineness of grid spacing), storing the solution may be impossible. In [17], the authors leverage simple motion primitives of the aircraft given by Eq. (2) that require lower-dimensional state representations of the aircraft. By restricting the movement of the aircraft to these primitives, the dimensionality of the reachability problem can be reduced. Recall that the ultimate goal is to find the feasible landing set. In Sec. VI, we provide a method for obtaining an approximation of \( Z \) that only requires computing reachable sets in four-dimensional spaces instead of six.

Many of the results that follow specifically pertain to the fixed-wing model given by Eq. (2). Though reachability analysis is general enough to handle a wide class of dynamical systems, the computational limitations may prove too difficult to overcome. One way to address this is to treat systems on a case-by-case basis and
leverage the structures in a particular model to come up with feasible solutions. The downside, of course, is that the solutions will often not generalize. Here, we settle for demonstrating a solution on a popular model class (fixed-wing aircraft without thrust). There is ongoing research on improving the computational effort needed for reachability analysis for more general classes of systems [21,22], from which this work will greatly benefit.

A. Invariance of Reachable Sets

At the onset of a forced landing, the aircraft should be able to access the set of feasible landing poses with very low latency. Failure to comply can lead to choosing a landing pose that is no longer feasible; for example, if it takes 5 min to obtain the feasible landing set for the initial state at the time of fault detection, then any state in the feasible landing set that requires a control signal different from the control applied for the past 5 min would be infeasible.

The desire for low-latency solutions rules out the possibility of solving the problem online. The alternative is to compute and store many feasible landing sets offline as a look-up table for different initial conditions. However, this solution presents its own challenges. How do we pick which initial conditions to compute solutions for? How do we approximate the solution for initial conditions different from those we chose? These problems can be mitigated offline by precomputing reachable sets for more initial conditions. However, the amount of memory needed for storage will increase as more sets are precomputed.

For initial conditions with the same velocity and flight-path angle, the set of poses that they can reach, ignoring terminal pose constraints, are the same up to a rotation and translation. For the dynamics in Eq. (2), the shape and size of the feasible landing stack are invariant to the initial pose of the aircraft, that is, given \( \xi^0 \) and \( \phi^0 \), if \( \xi(b^0) = \xi(0) \), then \( S(\xi^0) \) and \( S(\phi^0) \) will have the same shape and size. The rest of this section will focus on proving this. Because of the nonascent and flat ground assumptions being made under the engine-out fault, we make use of the FRS formulation given by Eq. (14).

We begin by defining the family of maps \( P_b : \mathbb{R}^4 \rightarrow \mathbb{R}^4 \) for \( b = [b_x \ b_y \ b_z \ b_w]^T \in \mathbb{R}^4 \):

\[
P_b(u) = A_{b,u} + \begin{bmatrix}
    b_x \\
    b_y \\
    b_z \\
    b_w
\end{bmatrix}
\]

\[
A_{b,u} = \begin{bmatrix}
    \cos b_w & -\sin b_w & 0 & 0 \\
    \sin b_w & \cos b_w & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

These maps are all Euclidean transformations in \( \mathbb{R}^4 \) (i.e., they constitute translations and rotations on vectors in \( \mathbb{R}^4 \)). We also define the family of maps \( O_b : \mathbb{R}^6 \rightarrow \mathbb{R}^6 \) for \( b \in \mathbb{R}^4 \):

\[
O_b(\xi) = \begin{bmatrix}
P_b(\pi_\phi(\xi)) \\
\pi_\xi(\xi)
\end{bmatrix}
\]

which are Euclidean transformations in \( \mathbb{R}^6 \).

For the model in Eq. (2), we have the following relationship:

\[
\frac{d}{ds} O_b(\xi(s; 0, \phi^0, u(\cdot))) = \begin{bmatrix}
    A_{b,u} & 0 \\
    0 & I
\end{bmatrix} f(\xi(s; 0, \phi^0, u(\cdot)))
\]

\[
= f(O_b(\xi(s; 0, \phi^0, u(\cdot))))
\]

where the last equality comes from Eq. (26).

From Proposition 1, we see that if two initial states \( \xi_u \) and \( \xi \) can be expressed as \( \xi = O_b(\xi_u) \) for some \( b \), then applying the same control signal for both initial conditions will yield equivalent state trajectories up to the Euclidean transformation \( O_b \). We will leverage this to show that, for any state \( \xi \), the feasible landing stack of \( \xi \) can be easily obtained from the feasible landing stack of \( [p^*, \xi(\xi)] \in \mathbb{R}^4 \), where \( p^* = [0, 0, 0, 0]^T \) is the nominal pose. This result is extremely useful because if we can guarantee that the aircraft has the same auxiliary state whenever a forced landing is initiated, then only one feasible landing stack needs to be computed, and it can be done so offline.

**Lemma 1:** Given \( \xi \) and \( \tilde{\xi} = [p^*, \xi_{\phi}(\xi)]^T \), there exists \( b \in \mathbb{R}^4 \), such that \( \xi = O_b(\tilde{\xi}) \).

**Proof:** Take \( b = \pi_\phi(\xi) \); it can be easily verified that \( \xi = O_b(\tilde{\xi}) \).

**Proposition 2:** Given \( \xi \) and \( \tilde{\xi} = [p^*, \xi_{\phi}(\xi)]^T \), there exists \( b \in \mathbb{R}^4 \), such that \( F(\xi) = O_b(F(\tilde{\xi})) \).

**Proof:** First, we will show \( O_b(F(\tilde{\xi})) \subseteq F(\xi) \). By Lemma 1, \( \exists b \) such that \( \xi = O_b(\tilde{\xi}) \). If \( w \in F(\xi) \), then \( \exists t \geq 0 \) and \( u_{\phi}(\cdot) \in \mathbb{U} \), such that \( \forall s \in [0, t] \pi_\phi(\xi(s; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot))) \in X^o \) and \( \xi(t; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot)) = w \). In addition, the trajectory \( \xi(t; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot)) \) must satisfy Eq. (5) with initial condition \( \tilde{\xi} \) and control signal \( u_{\phi}(\cdot) \).

Applying the same signal \( u_{\phi}(\cdot) \) with initial state \( \tilde{\xi} \), the resulting state trajectory must solve Eq. (5) with initial condition \( \tilde{\xi} \). By Proposition 1 (and invoking uniqueness of solutions to differential equations) the resulting state trajectory must be \( \xi_{\phi}(\tilde{\xi}; 0, \tilde{\xi}_{\phi}(\tilde{\xi}), u_{\phi}(\cdot)) \) with \( O_b(\xi_{\phi}(\tilde{\xi}; 0, \tilde{\xi}_{\phi}(\xi), u_{\phi}(\cdot))) = O_b(\xi_{\phi}(\xi; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot))) \). Furthermore \( \pi_\phi(\xi(s; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot))) = \pi_\phi(\xi(s; 0, \tilde{\xi}_{\phi}(\xi), u_{\phi}(\cdot))) \), and \( \forall s \in [0, t] \pi_\phi(\xi(s; 0, \xi_{\phi}(\xi), u_{\phi}(\cdot))) \in X^o \). Thus, \( O_b(w) \in F(\xi) \). Because \( w \) was arbitrary, we conclude that \( O_b(F(\tilde{\xi})) \subseteq F(\xi) \).

To complete the proof, note that \( \exists b \) such that \( O_b(\cdot) = O_b(\cdot)^{-1} \), in particular \( b = -A_{b,u} b \). Using the previous argument, we have \( O_b(F(\tilde{\xi})) \subseteq F(\xi) \). Combining the two set inequalities, we have \( F(\xi) \supseteq O_b(F(\tilde{\xi})) \supseteq O_b(\pi_\phi(F(\tilde{\xi}))) = F(\xi) \), and thus we conclude \( O_b(F(\tilde{\xi})) = F(\xi) \).

**The only constraints that are present are those on the auxiliary state, which are unchanged by the transformations being considered.**

In general, constraints on the pose would not be satisfied after the transformation because the pose is obviously modified. Fortunately, the flat ground and nonascending aircraft assumptions allow us to ignore the pose constraints. A similar result can be derived for the feasible landing stack.

**Corollary 1:** Given \( \xi \) and \( \tilde{\xi} = [p^*, \xi_{\phi}(\xi)]^T \), there exists \( b \in \mathbb{R}^4 \), such that \( S(\xi) = P_b(S(\tilde{\xi})) \).

**Proof:** By definition, \( S(\xi) = \pi_\phi(F(\xi) \cap (\mathbb{R}^4 \times \mathbb{G})) \). By Proposition 2, \( \pi_\phi(F(\tilde{\xi}) \cap (\mathbb{R}^4 \times \mathbb{G})) = \pi_\phi(O_b(\pi_\phi(F(\tilde{\xi})) \cap (\mathbb{R}^4 \times \mathbb{G}))) = P_b(S(\tilde{\xi})) \).

Finally, \( P_b(\pi_\phi(F(\tilde{\xi})) \cap (\mathbb{R}^4 \times \mathbb{G})) = P_b(S(\tilde{\xi})) \).
We can conclude that the shape and size of the FRS and feasible landing stack of a single state $\xi$ are invariant to the pose $p$. Note that these sets can be obtained by applying Euclidean transformations to $\mathcal{F}(\xi)$ and $\mathcal{G}(\xi)$, respectively, for $\tilde{\xi} = [p^T \cdot \pi(\xi)]^T$, which has no dependence on the pose $\pi(\xi)$. For a fixed auxiliary state $\tilde{\eta}$, the feasible landing stack only needs to be computed for $[p^T \cdot \tilde{\eta}]^T$.

Once the set is computed, it can be reused (up to a Euclidean transformation) for any initial $\xi$ such that $\pi(\xi) = \tilde{\eta}$. A visualization of the invariance property can be seen in Fig. 2. To have a three-dimensional visualization, the heading is fixed. The shape of the reachable set does not change, but in general, the feasible landing set will be different, depending on where the pose constraint (ground level) intersects the reachable set.

Analogous invariance properties can also be obtained for the BRS by making similar arguments. We state them now without proof.

**Proposition 3:** Given two terminal sets $\mathcal{Y}_1$ and $\mathcal{Y}_2 = \mathcal{O}_b(\mathcal{Y}_1)$ with $b \in \mathbb{R}^3$, $\mathcal{B}(\mathcal{Y}_2) = \mathcal{O}_b(\mathcal{B}(\mathcal{Y}_1))$. Furthermore, $w \in \mathcal{B}(\mathcal{Y}_2) \Leftrightarrow O_b(\xi)(w) \in \mathcal{B}(\mathcal{Y}_1)$.

**Proposition 4:** Given two target sets $\mathcal{Y}_1$ and $\mathcal{Y}_2 = \mathcal{O}_b(\mathcal{Y}_1)$ with $b \in \mathbb{R}^3$ and constraint set $\mathcal{C}$, $\mathcal{X}(\mathcal{Y}_2; \mathcal{Y}_1; \mathcal{C}) = \mathcal{X}(\mathcal{O}_b(\mathcal{Y}_2; \mathcal{Y}_1; \mathcal{C}))$.

**Remark 2:** The intuition behind Proposition 4 is rather straightforward. Take the singleton case, $\mathcal{Y}_1 = \{\xi\}$ and $\mathcal{Y}_2 = \{\xi\}$. Then $O_b^{-1}(w) \in \mathcal{B}(\xi)$ by Proposition 3. From the definition of BRS and FRS, $\xi \in \mathcal{F}(\mathcal{w})$ and $\tilde{\xi} \in \mathcal{F}(O_b^{-1}(w))$. Any control signal that drives the system from $O_b^{-1}(w)$ to $\tilde{\xi}$ will also drive the system from $w$ to $\xi$, and vice versa. Therefore, the minimum time to get from $w$ to $\xi$, is equal to the minimum time to get from $O_b^{-1}(w)$ to $\tilde{\xi}$. Furthermore, the optimal control in both cases will be the same.

Only one BRS and control law need to be computed. All that is left is to apply the appropriate transformation to the state before querying the control law for the correct input.

### B. Flight Primitives

To describe the motion of the fixed-wing aircraft, we use a six-dimensional model. However, given the goal of emergency landing an aircraft with engine failure (no thrust capability), we can consider motion primitives of the aircraft that require simpler models. For example, given the lack of thrust, it is likely that the aircraft will spend a good portion of its flight time gliding, which is described by a four-dimensional model.

More complex aircraft motions can be composed from the primitives. By restricting the aircraft to only being able to move according to these primitives, we can greatly simplify any analysis on the aircraft’s movement, including where it can land and how to synthesize controllers. In [17], the authors describe three motion primitives for a fixed-wing aircraft with engine failure: gliding, constant turning rate, and velocity adjustment. We describe the primitives here briefly.

1. **Gliding**

In this primitive, the aircraft moves in straight-line paths (in the horizontal plane) with constant velocity and flight-path angle. More compactly, $\dot{\gamma} = \dot{\psi} = \dot{v} = 0$. The control is fixed to the gliding control $[\phi_G, \varphi_G]$, where $\phi_G = 0$ and $\alpha_G$ maximizes the ratio of distance traveled to altitude loss, which is equivalent to maximizing the lift-to-drag ratio:

$$\alpha_G = \arg \max_{\alpha} \frac{C_L(\alpha)}{C_D(\alpha)}$$

(28)

Under this fixed control, the equilibrium velocity and flight-path angle are given as $v_G$ and $\gamma_G$, respectively. The pose dynamics are then

$$\dot{p} = \begin{bmatrix} v_G \cos \gamma_G \cos \psi \\ v_G \cos \gamma_G \sin \psi \\ v_G \sin \gamma_G \\ 0 \end{bmatrix}$$

(29)

A four-dimensional model is used to describe the motion of the aircraft within this primitive.

2. **Constant-Rate Turning**

In this primitive, the aircraft turns at a constant turning rate, keeping its speed and flight-path angle constant. More compactly, $\dot{\gamma} = \dot{\psi} = \dot{v} = 0$. The control is held fixed at the turning control $[\phi_T, \varphi_T]$, where $\phi_T$ and $\alpha_T$ are set to minimize the ratio between altitude loss and change in aircraft orientation $dz/d\psi = \dot{z}/\dot{\psi}$.

$$\begin{bmatrix} \phi_T \\ \varphi_T \end{bmatrix} = \arg \min_{\alpha, \phi} \frac{L(\alpha, \phi)}{mv^2 \cos \gamma \sin \gamma}$$

(30)

The optimization is performed numerically by indexing through equilibrium airspeed and flight-path angle pairs $(v, \gamma)$. For a given equilibrium pair $(v, \gamma)$, the corresponding controls $\alpha$ and $\phi$ are solved for by setting the last two expressions in Eq. (2) equal to zero. We then index through $(v, \gamma)$, along with the corresponding controls, to find the minimizer of Eq. (30).

The minimizing equilibrium pair is given as $v_T$ and $\gamma_T$, respectively, and the corresponding turning rate is given by $\psi = \pm \alpha_T$.

$$\alpha_T = \frac{L(\alpha_T, v_T) \sin \varphi_T}{mv_T \cos \gamma_T \sin \gamma_T}$$

(31)

The pose dynamics are

$$\dot{p} = \begin{bmatrix} v_T \cos \gamma_T \cos \psi \\ v_T \cos \gamma_T \sin \psi \\ v_T \sin \gamma_T \\ \pm \alpha_T \end{bmatrix}$$

(32)

Note that $\varphi_T$ and $\alpha_T$ can both be either positive or negative, depending on whether the aircraft is turning left or right. A four-dimensional model is used for this primitive.
3. Velocity Adjustment

In this primitive, the aircraft moves in straight-line paths (in the horizontal plane), i.e., \( \phi = 0 \), and maintains a constant orientation \( \psi_0 \). The aircraft can also adjust its velocity through the AOA \( \alpha \). The dynamics are:

\[
\dot{\xi} = \begin{bmatrix}
    v \cos \gamma \cos \psi_0 \\
    v \cos \gamma \sin \psi_0 \\
    v \sin \gamma \\
    0
\end{bmatrix}
\]

(33)

Because the velocity can only be adjusted in this primitive, it is the last primitive that the aircraft enters before landing to reduce the speed.

Here, the entire six-dimensional model is used to describe the motion of the aircraft. However, the model can be reduced by two dimensions due to the straight-line constraint (heading can be ignored, and \( (x, y) \) can be merged into one state). We will leverage the lower-dimensional model in RBFLS.

V. Forced Landing Procedure

Assuming that the aircraft will always have the same auxiliary state (i.e., velocity and flight-path angle), when it initiates a forced landing, we only need to compute one feasible landing set offline and apply the appropriate Euclidean transformation online. This result is due to Corollary 1 and the subsequent discussion.

However, because of the previously mentioned computational challenges of working with a six-dimensional model, we restrict the analysis of our system to two phases that require lower-dimensional models. We call these two phases the cruising phase and landing phase. Once in the landing phase, the aircraft will reduce its velocity while traveling along a constant heading until it has reached the desired landing with an appropriate landing speed and flight-path angle. Reachable sets can be computed for the two phases individually and then composed to obtain approximations of the feasible landing set. This scheme will result in an under-approximation of the computed feasible landing region, but this is typically not too restrictive because the cruising phase is optimized to maximize the landing region. We first describe the cruising and landing phases and present algorithms for computing the feasible landing set and synthesizing controllers for landing.

A. Cruising Phase

The cruising phase is a combination of the gliding primitive and constant-rate turning primitive. The aircraft is constrained to have a constant velocity, and it can either fly straight or turn left or right at a fixed turning rate. To make this more precise, define the set of cruising-phase control inputs \( \mathcal{U}_C \subseteq \mathcal{U} \):

\[
\mathcal{U}_C = \left\{ \begin{bmatrix} c_G \\ \alpha_T \\ \phi_T \\ 0 \end{bmatrix} \right\}
\]

(34)

For each input in \( \mathcal{U}_C \), the velocity \( v \), flight-path angle \( \gamma \), and angular velocity \( \dot{\psi} \) have an equilibrium value. We define the set of equilibrium states for the auxiliary states and angular velocity:

\[
\mathcal{E}_C = \left\{ \begin{bmatrix} v_G \\ \gamma_G \\ 0 \\ \omega_T \\ -\alpha_T \end{bmatrix} \right\}
\]

(35)

We assume that the equilibrium states still satisfy the constraints on the auxiliary state \( X_u \).

If the control is constant, \( \forall t \in [0, T], u(t) = u_* \in \mathcal{U}_C \), the resulting dynamics will be no higher than four-dimensional because at least two states will be fixed. However, this no longer holds once the controller is allowed to switch due to the transients (i.e., it takes time to move from one equilibrium condition to another). We make the assumption that the effect of these transients is negligible and allow our aircraft to switch between the primitives instantaneously. In Sec. VI, we verify this assumption empirically. Define the mappings \( [v_0(\cdot), \gamma_0(\cdot), \omega_0(\cdot)]; \mathcal{U}_C \rightarrow \mathcal{E}_C \) as the equilibrium velocity, flight-path angle, and angular velocity associated to a particular control \( u \in \mathcal{U}_C \). For example, \( [\mathcal{U}_C, 0] \rightarrow \mathcal{E}_C \). We take the cruising-phase state to be

\[
\xi_C := \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}
\]

(36)

and express the dynamics as \( \dot{\xi}_C = f_C(\xi_C, u) \), where

\[
f_C(\xi_C, u) = \begin{bmatrix}
    v_0(u) \cos \gamma \cos \psi \\
    v_0(u) \cos \gamma \sin \psi \\
    v_0(u) \sin \gamma \\
    0
\end{bmatrix}
\]

(37)

with \( u \in \mathcal{U}_C \). We adapted the model in Eq. (2) to allow for instantaneous switching between equilibrium conditions.

The set of measurable cruising-phase control signals over time horizon \( T \) is \( \mathcal{U}_C := \{ u : [0, T] \rightarrow \mathcal{U}_C \} \) is measurable. We also define the cruising-phase state trajectory \( \xi_C(\cdot) \), which solves the differential equation associated with Eq. (37) for some initial cruising state and control signal.

B. Landing Phase

The landing phase is identical to the velocity adjustment primitive. During this phase, the aircraft is restricted to flying along a straight line in the horizontal plane, but the velocity is adjustable by changing the angle of attack \( \alpha \). Thus, the set of allowable control inputs is \( \mathcal{U}_L := \{ u : [0, 0]^T \in \mathcal{U} \} \).

As we stated earlier, the dynamics of this primitive can be described with four states. Given an initial pose \( p^0 = [x^0, y^0, z^0, \psi^0] \), we define two new states \( r = \sqrt{(x - x^0)^2 + (y - y^0)^2} \) and \( z = z - z^0 \). The landing-phase state is defined as

\[
\xi_L := \begin{bmatrix} r \\ z \\ u \\ \gamma \end{bmatrix}
\]

(38)

Because the plane cannot change its heading during the landing phase, the landing-phase state evolves according to \( \dot{\xi}_L = f_L(\xi_L, u) \), where

\[
f_L(\xi_L, u) = \begin{bmatrix}
    v \cos \gamma \\
    v \sin \gamma \\
    -\frac{\alpha_T}{m} - g \sin \gamma \\
    \frac{I_A \alpha_T}{m v^2} - \frac{v}{v} \cos \gamma
\end{bmatrix}
\]

(39)
Furthermore, we can define the family of injective mappings \( \mathcal{H}_b : \mathbb{R}^2 \to \mathbb{R}^2 \) with \( b \in \mathbb{R}^2 \), which maps the landing-phase state \( \xi_L \) back to the full state \( \bar{z} \) if the aircraft began in the landing phase having pose \( b \). In particular,

\[
\mathcal{H}_b(\xi_L) = \begin{bmatrix}
    b_x + r \cos b_y \\
    b_y + r \sin b_y \\
    b_z + \bar{z} \\
    b_x \\
    b_y \\
    b_z \\
\end{bmatrix}
\]

where \( b = [b_x, b_y, b_z]^T \). In addition, if we would like to just know the pose of the aircraft while it is in the landing-phase state, then we can compose this mapping with the projection mapping \( \pi_p \):

\[
\pi_p \circ \mathcal{H}_b(\xi_L) = \begin{bmatrix}
    b_x + r \cos b_y \\
    b_y + r \sin b_y \\
    b_z + \bar{z} \\
\end{bmatrix}
\]

Similar to the cruising phase, we define the set of measurable landing-phase control signals over time horizon \( T \); \( \mathcal{U} = \{ \beta : [0, T] \to \mathcal{U}_L | \beta(\cdot) \text{ is measurable} \} \). The landing-phase state trajectory is given by \( \xi_L(\cdot) \), which solves the differential equation associated with Eq. (39) for some initial landing-phase state and control signal.

C. Underapproximating the Feasible Landing Set

Our objective is to approximate the feasible landing set and stack for an engine-out fixed-wing aircraft by assuming that the aircraft executes its landing by transitioning from the cruising phase to the landing phase. To not introduce new notation, we still use \( Z \) and \( S \) to refer to the approximations of the feasible landing set and stack, respectively. We also assume that, when the forced landing is initiated, \( \pi_r(\bar{z}) = [v_G, \gamma_0] \), that is, the auxiliary state is in the gliding equilibrium, which is achieved by applying the gliding control \( u = [\alpha_G, \phi_G] \) until the auxiliary state reaches the gliding equilibrium. Under this assumption, the feasible landing stack is invariant to the position, and we only have to compute it for the nominal emergency state \( \bar{z}^* = [p^*, v_0, \gamma_0] \). For any other pose, we simply apply the appropriate Euclidean transformation and use Eq. (12) to obtain the feasible landing set.

The feasible landing stack can be constructed as follows.

1) For the cruising phase, compute the reachable for \( \bar{z}^* \).
2) For the landing phase, compute the reachable set for each state in the cruising-phase reachable set.
3) Take the union of all the landing-phase reachable sets. The feasible landing stack is then obtained by projecting this set onto the pose.

Given the nominal emergency state \( \xi^* = [v_G, \gamma_0]^T \) and \( \pi_r(\bar{z}) = \bar{p} \), we can compute the feasible landing set by simply applying a Euclidean transformation on \( S(\xi^*) \) and taking an intersection with the ground pose constraint \( G_p \):

\[
(\xi^*) = \bigcup_{b \in F_C(p^*)} \pi_p \circ \mathcal{H}_b(F_L(\xi_L^*)) \cap (\mathbb{R}^2 \times G_p)
\]

This set represents poses that the aircraft can reach in the cruising phase starting from the pose \( p^* \).

There is no explicit constraint on the cruising-phase state trajectory. The constraint on the auxiliary state is accounted for by Eq. (35) because we assume that the auxiliary state switches between equilibriums. The cruising-phase FRS can be computed via variational inequality (20) and Eq. (23). However, appropriate substitutions must be made (i.e., \( \mathcal{U}_L, f, \xi, \Phi_{0} \) is replaced by \( \mathcal{U}_L, f, \xi_C, \Phi_{0} \)). In addition, the constraint in Eq. (20) would be set to \( \mathcal{X}_p \) or removed entirely under the nonascent and flat ground assumptions.

For any pose in \( F_C(p^*) \), the aircraft can switch into the landing phase. If we define the landing-phase state \( \xi_L \) with respect to the current pose at the time of switching, then initially \( r = \bar{z} = 0 \). Furthermore, we assume that the aircraft is gliding before it enters the landing phase. This assumption is fine because, in the cruising phase, we allow for instantaneous switching between turning and gliding. The assumption also implies that, upon entering the landing phase, the initial landing-phase state is \( \xi_L^* = [0, 0, v_G, \gamma_0]^T \).

It now remains to analyze reachability for the landing phase. The landing-phase forward reachable set for \( \xi_L^* \) is given as

\[
F_L(\xi_L^*) = \{ w \in \mathbb{R}^4 | \forall t \in [0, T], u(t) \in U_L, \xi_L(0) = \xi_L^*, \xi_L(t) = w \}
\]

This set can be computed via variational inequality (20) and Eq. (23). Again, appropriate substitutions must be made (i.e., \( \mathcal{U}_L, f, \xi, \Phi_{0} \) is replaced by \( \mathcal{U}_L, f, \xi_L, \xi_L^* \)). In addition, the constraint set \( C \) would be set to \( C_L = \mathbb{R}^2 \times \mathcal{X}_p \).

Not every state in the landing phase constitutes a safe landing. For this, we need to incorporate the ground auxiliary constraint \( G_p \), by intersecting \( F_L(\xi_L^*) \) and \( \mathbb{R}^2 \times G_p \). This constraint ensures that we only consider states that have the landing speed and flight-path angle within the appropriate limits.

To construct the feasible landing stack, the set \( F_L(\xi_L^*) \cap (\mathbb{R}^2 \times G_p) \) must be mapped to poses. Given the initial pose upon entering the landing phase, this can be achieved via Eq. (41), and thus the feasible landing stack is given by

\[
S(\xi^*) = \bigcup_{b \in F_C(p^*)} \pi_p \circ \mathcal{H}_b(F_L(\xi_L^*) \cap (\mathbb{R}^2 \times G_p))
\]

In a nutshell, this equation describes how to obtain the feasible landing stack. First, compute the cruising-phase FRS, then take the union of the landing-phase FRS (intersected with the ground auxiliary constraint) for all the states in the cruising-phase FRS, and last extract the pose of these states.

For any aircraft state \( \bar{z} \) with \( \pi_r(\bar{z}) = [v_G, \gamma_0]^T \) and \( \pi_r(\bar{z}) = \bar{p} \), we can recover the feasible landing set by first applying a Euclidean transformation on \( S(\xi^*) \) and taking an intersection with the ground pose constraint \( G_p \):

\[
Z(\bar{z}) = \mathcal{P}_p(S(\xi^*)) \cap G_p
\]

Recall that the ground pose constraint ensures that we only consider poses that correspond to being on the ground.

D. Controller Synthesis

Now we want to synthesize a controller to guide the aircraft to a desired pose \( p \). Again, we impose that the aircraft begins in the cruising phase and transitions to the landing phase. To land safely, we also require that the auxiliary state satisfy the ground auxiliary constraint; thus, the goal is equivalent to guiding the aircraft state to the set \( \{ p \} \times G_p \). Earlier, we showed that a controller can be synthesized for a terminal set \( \bar{Y} \) by computing its BRS. All terminal sets of interest have the form \( \{ p \} \times G_p \) and so we just focus on computing the BRS for \( \bar{Y} = \{ p \} \times G_p \). By Proposition 3, any BRS of interest can be obtained through a Euclidean transformation of \( B(\bar{Y}) \).

Because the aircraft must terminate in the landing phase, the desired landing pose must be achieved in the landing phase. Thus, \( B(\bar{Y}) \) should at least include the set of states that can reach \( \bar{Y} \) in the landing phase. If the landing-phase state \( \xi_L \) is defined with respect to \( p^* \), then reaching \( \bar{Y} \) is equivalent to reaching a landing-phase state in \( \mathcal{Y}_L^* = \{ [0, 0]^T \} \times G_p \subseteq \mathbb{R}^4 \). The landing-phase backward reachable set of \( \mathcal{Y}_L^* \) is given by
Aircraft speed (normalized by glide velocity)

The optimal feedback controller to reach $\gamma^c_r$ while satisfying $C_L$ in the landing phase is given by $\kappa_C(\gamma^c_i; C_L) : \mathbb{R}^2 \rightarrow U_L$:

\[
\kappa_C(\gamma^c_i; C_L) = \arg \min_{\theta \in \Theta} D_{\gamma^c_i} V_{\gamma^c_i}(\gamma^c_i, -\theta) \cdot f(\gamma^c_i, u)
\]

\[
\hat{\gamma} = \min \{ t | V_{\gamma^c_i}(\gamma^c_i, -\theta) \leq 0 \}
\]

Once the aircraft state $\gamma_i$ is in $H_{\gamma^c} (B_1(\gamma^c_i))$, this implies that the landing-phase state is in $B_1(\gamma^c_i)$, and the control in Eq. (47) can be applied to reach the target. Because the aircraft begins in the cruising phase before initiating the landing phase, we must properly represent $H_{\gamma^c} (B_1(\gamma^c_i))$ as a target in the cruising phase. Recall that, upon exiting the cruising phase, we assume that the aircraft is in the gliding equilibrium. Thus, the aircraft can only reach a certain subset of $H_{\gamma^c} (B_1(\gamma^c_i))$, which is $H_{\gamma^c} (B_1(\gamma^c_i)) \cap \mathbb{R}^3 \times \{(v_G, \gamma_G) \}$. Projecting this set onto the poses yields the target set for the cruising phase $\gamma^c_r = \pi_p \circ H_{\gamma^c} (B_1(\gamma^c_i) \cap \mathbb{R}^3 \times \{(v_G, \gamma_G) \}) \subseteq \mathbb{R}^4$. Even though $H_{\gamma^c}$ maps from $\mathbb{R}^4$ to $\mathbb{R}^6$, we never explicitly go to a six-dimensional space because

\[
\pi_p \circ H_{\gamma^c} (\xi_L) = \begin{bmatrix} r \\ \dot{z} \\ 0 \\ 0 \end{bmatrix}
\]

where $\xi_L = [r, \dot{z}, v, \gamma]^T$.

To steer the aircraft to the desired landing pose $p^*$ (while satisfying the auxiliary constraint), the aircraft needs to have a pose in the set $\gamma^c_r$ before switching into the landing phase. Thus, we construct a controller that will steer the aircraft to $\gamma^c_r$ during the cruising phase. The cruising-phase backward reachable set of $\gamma^c_r$ is given by

\[
B_C (\gamma^c_r) = \{ w \in \mathbb{R}^4 | \exists t \in [0, T], u(t) \in U_C, \xi(t) = 0 \} 
\]

\[
\forall s \in [0, t], \xi_C(t) \in \gamma^c_r 
\]

which we compute using Eqs. (20) and (21). We make the appropriate substitutions (i.e., $[\xi(t), f, a, \gamma]$ is replaced by $[U_L, f_C, \xi_C, \gamma^c_C]$). There is no explicit constraint for the cruising phase on the cruising-phase state.

The optimal feedback controller to reach $\gamma^c_r$ in the cruising phase is given by $\kappa_C(\cdot; \gamma^c_r) : \mathbb{R}^2 \rightarrow U_C$:

\[
kappa_C(\gamma^c_i; \gamma^c_r) = \arg \min_{\theta \in \Theta} D_{\gamma^c_i} V_{\gamma^c_i}(\gamma^c_i, -\theta) \cdot f(\gamma^c_i, u)
\]

\[
\hat{\gamma} = \min \{ t | V_{\gamma^c_i}(\gamma^c_i, -\theta) \leq 0 \}
\]

Combining the cruising-phase and landing-phase controllers yields the cruise-landing controller for reaching $\gamma^c_r$:

\[
kappa_{CL}(\gamma^c_i; \gamma^c_r; C_L) = \begin{cases} \kappa_C(\xi_L^C; C_L), & \pi_p(\xi) \in \gamma^c_C \\ \kappa_C(\pi_p(\xi); \gamma^c_C), & \text{o.w.} \end{cases}
\]

In addition, Proposition 4 also applies to the cruise-landing controller. The cruise-landing controller for reaching any terminal set of the form $\mathcal{O}_d(\gamma^c_r)$ is

\[
kappa_{CL}(\gamma^c_i; \mathcal{O}_d(\gamma^c_r); C_L) = \kappa_{CL}(\mathcal{O}_d(\gamma^c_r); \gamma^c_r; C_L)
\]

VI. Results and Discussion

Inspired by the Miracle on the Hudson, we consider an aircraft with a loss of thrust that must make an emergency landing in New York City from an altitude of 1220 m. For the dynamical system, we use the NASA Transport Class Model (TCM) Aircraft [32]. The reachable sets are computed using the Level Set Toolbox in MATLAB [33].

A. Parameters

Before computing the feasible landing set, a number of specified and computed parameters must be obtained. The physical parameters and constraints for the aircraft were obtained from NASA Neil A. Armstrong Flight Research Center and are given in Table 1.

The drag and lift coefficients are both approximated as piecewise linear functions of $\alpha$:

\[
C_D(\alpha) = \begin{cases} 0.0173\alpha + 0.1254 & \alpha < 5.3078 \text{ deg} \\ 0.0831\alpha - 0.227 & \text{o.w.} \end{cases}
\]

\[
C_L(\alpha) = \begin{cases} 0.0936\alpha + 0.6040 & \alpha < 5.946 \text{ deg} \\ 0.0238\alpha + 1.019 & \text{o.w.} \end{cases}
\]

Given the physical parameters of the aircraft, the optimal glide velocity $v_G$ and glide flight-path angle $\gamma_G$ were determined to be 148.74 m/s and $-5.46$ deg, respectively. For the constant-rate turning primitive, Adler et al. [17] optimized the aircraft speed and flight-path angle numerically to maximize Eq. (30). The correctness of the analysis in the cruising phase hinges on the ability to switch
between the glide and constant-rate turning primitives with low settling time. Keeping this in mind, we set $v_T/v_G = 0.0136$ and only optimize over the flight-path angle, which yields $\gamma_T = -10.56$ deg and $\omega_T = 6.03$ deg/s. The rationale is that keeping the two equilibria “close” together will minimize the settling time. In Fig. 3, we validate the low-latency assumption empirically by investigating the effect of switching between the two primitives (at 0.1 Hz) on the aircraft speed and turning rate. The blue curve represents the ideal curve assumed in the analysis of the cruising phase, and the red curve is the actual response seen in simulation.

**B. Feasible Landing Set**

Once the parameters are obtained, the cruise–land feasible landing set is computed using the method outlined in Sec. V.C. A projection of the feasible landing stack onto the $(x, y, z)$ space is shown in Fig. 4. The projection is formed by taking a union over all the possible headings $\psi$. The black region represents the initial location, the red region represents the reachable locations during the cruising phase, and the blue region represents the locations reachable during the landing phase and thus is the feasible landing stack. Slices of the stack along the $z$ axis correspond to the feasible landing set at different ground heights. The $(x, y, z)$ axes are normalized by $4.2 \times 10^4$, $4.2 \times 10^4$, and $4.0 \times 10^3$ ft, respectively. The feasible landing stack is four-dimensional, but for visualization purposes, this set is projected onto the position space $(x, y, z)$. The projection displays the positions that the aircraft can reach but gives no information as to what the heading shall be for a given position. In general, the heading of the
fixed-wing is important for assessing the cost that it will incur because the aircraft will slide before coming to a complete stop.

Recall that the feasible landing stack differs from the feasible landing set, in that the ground pose constraint is not taken into account. Once the ground height is specified, the feasible landing set can be obtained by taking a slice of the stack along the $z$ axis. A projection of a feasible landing set (again ignoring heading) can be seen on the New York cost map in Fig. 5. In this example, the aircraft is 4000 ft above ground.

At first glance, it may seem counterintuitive in Fig. 4 that the set of reachable positions in the cruising phase (the red set) is completely detached from the feasible landing set (the blue set). This result is due to the constraints on the auxiliary states. Though the poses in the cruising phase are reachable, the auxiliary states will not satisfy the ground auxiliary constraints. This constraint violation is primarily because the glide velocity $v_g$ during the cruising phase is greater than the maximum allowable landing speed velocity $v_G$. Furthermore, once the aircraft switches into the landing phase, it takes some time before the auxiliary state can be driven into the ground auxiliary constraint set.

C. Controller Synthesis

Using the method outlined in Sec. V.D, we design a feedback controller to land the aircraft at the nominal pose $p^\ast$. The resulting backward reachable set can be seen in Fig. 6. The black region is the target set, which represents the intended landing state. The green region is the landing-phase backward reachable set. The red region constitutes the subset of the green region that can be reached when the aircraft is in the gliding primitive. The red region serves as the target set for the cruising phase, from which the cruising-phase backward reachable set, shown as the blue region, is calculated. The $(x, y, z)$ axes are normalized by $4.2 \times 10^4$, $4.2 \times 10^4$, and $4.0 \times 10^3$, respectively. There is also a sample trajectory from a location in the backward reachable set to the black target that uses the feedback control generated from the backward reachable set.

It can be seen that the nonascending assumption of the aircraft is actually not satisfied during the landing phase. There are two things to be said about this. First, the proposed algorithm for computing the feasible landing set needs to be slightly modified because the invariance properties hinge on this assumption. This modification will be discussed later in the section.

Second, reachability analysis can provide strong insights about the system dynamics. When first computing the backward (and forward) reachable set, we imposed the nonascent assumption because it allowed for the invariance properties discussed in Sec. IV.A, and it also made practical sense. However, this resulted in extremely small (essentially negligible) sets. Removing that constraint yields the sets
in Figs. 4 and 6. From the simulated trajectories, it is clear that the aircraft needs to first ascend during the landing phase to reduce its speed, given the system model and parameters. The insights obtained from the analysis can then be used to inform a more appropriate forced landing algorithm.

Now we address the nonascent assumption. Rather than only assuming an initial $\eta$ when constructing the feasible landing set and controller, we instead have to assume an initial $\eta$ and altitude $z$. The aircraft would then access the feasible landing set once it has reached the auxiliary state and altitude that the computation was performed for. The feasible landing set would still retain its translational invariance in the $(x, y)$ plane because the flat ground assumption holds. Multiple feasible landing sets can be computed for different altitudes and used as a look up table online. However, the computation will be $m$ times as expensive (offline) as the former algorithm, where $m$ is the number of altitudes the feasible landing set is computed for.

VII. Conclusions

With UAVs becoming more ubiquitous, forced landing systems will be an integral part of the civilian airspace. This paper considered the problem of executing an emergency forced landing for a fixed-wing aircraft with loss of thrust. The issues of identifying feasible landing locations and constructing controllers for landing such aircraft were specifically addressed. The proposed solution, based on HJB reachability, is general enough to account for various kinds of aircraft dynamics and constraints. Such an analysis can be very informative in facilitating the design of a forced landing algorithm, especially when the results differ from the heuristic assumptions. Assessing the feasibility of different landing locations cannot be left solely to intuition or heuristics, or else there is the risk of attempting to reach locations that are, in fact, not reachable or not safely reachable. Analysis tools provide guarantees about where it is the aircraft can and cannot reach, and this informs the decisions that will be made during the forced landing.

Acknowledgments

Anayo K. Akametalu and Mo Chen were equally contribute on this work. This paper was funded through the University of California, Berkeley Chancellor’s Fellowship, and a NASA Neil A. Armstrong Flight Research Center grant, award NNX12AR18A. This work was made possible through a collaboration with NASA Neil A. Armstrong Flight Research Center. In particular, we would like to thank Jinu Idicula for many discussions and for providing us with model parameters and figures.

References


