Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Attacker Dispersal Surface in the Turret Defense Differential Game

Alexander Von Moll [*,**] **Zachariah Fuchs** [**]

[*] *Air Force Research Laboratory, WPAFB, OH 45433 USA*
(email: alexander.von_moll@us.af.mil)
[**] *University of Cincinnati, Cincinnati, OH 45221 USA*
(email: fuchsze@ucmail.uc.edu)

Abstract: The characteristics for the solution to the Turret Defense Differential Game are explored over the parameter space. We collapse the five natural parameters of capture radius, attacker speed, turret turn rate, time penalty constant, and look-angle penalty constant into two composite parameters in order to facilitate the analysis. There exist three singular surfaces in the game, two of which have an analytic form, and one which can only be obtained numerically. We focus on the latter: the Attacker Dispersal Surface, wherein the attacker can choose between an indirect or direct route to capture. For certain parameter settings, the Attacker Dispersal Surface is present, while for others, the surface is absent. These regions in the parameter space are identified, and the numerical procedures to do so are detailed. Backwards shooting of the optimal state and adjoint dynamics features prominently in the procedures. Two pieces of numerical evidence are utilized to indicate the presence or absence of the Attacker Dispersal Surface.

Keywords: Differential or dynamic games; Singularities in optimization; Autonomous systems; Navigation, Guidance, and Control;

1. INTRODUCTION

In differential games, parameters, often representing physical properties of the system, can play a significant role in the overall nature of the solutions. In particular, the number, type, and shape of singular surfaces in the state space are all subject to change for different settings of parameters. Isaacs identified different types of singular surfaces including dispersal, universal, barrier, equivocal, and transition surfaces (Isaacs, 1965). Later, Merz discovered focal surfaces, whereby the optimal trajectories approach tangentially (Merz, 1971; Breakwell and Bernhard, 1990). There is, perhaps, no better example demonstrating the effects that parameters can have on the singular surfaces than the Homicidal Chauffeur Differential Game (HCDG) (Merz, 1971). See, for example, Fig. 1 which portrays a partitioning of the parameter space into regions in which the overall solution, comprised of regular trajectories and singular surfaces, is fundamentally different. The singular surfaces, themselves, are important in that they often divide the state space into regions of win for either agent or regions of different control. Therefore, the importance of understanding singular surfaces and their relationship to the game's parameters cannot be overstated. Much of the work on singular surfaces by Isaacs, Breakwell, Bernhard, Melikyan, and Lewin has focused on analytical techniques (c.f. Isaacs, 1965; Breakwell and Bernhard, 1990; Melikyan, 1994; Lewin, 1994). However, the focus in this paper is on numerical techniques, for which there is a dearth of literature.

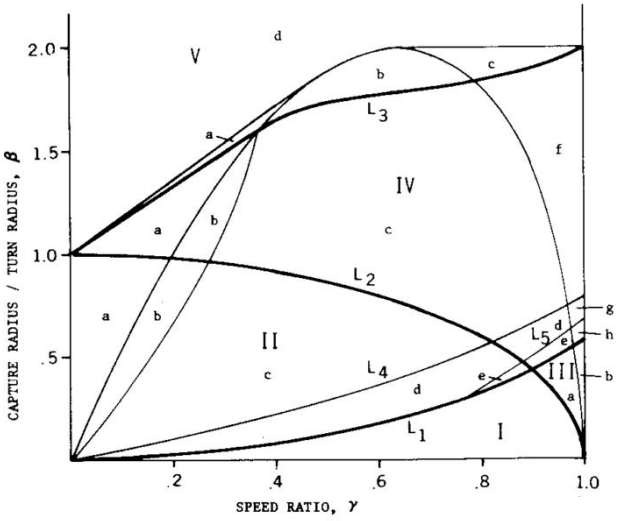

Fig. 1. Solution characteristics over the space of parameters in the Homicidal Chauffeur Differential Game, reproduced from Merz (1971).

The Turret Defense Differential Game (TDDG) was introduced by Akilan and Fuchs (2017) and solved for one particular setting of parameters. The TDGG involves an engagement between a mobile Attacker with simple motion and a stationary Turret with a fixed turn rate. An integral

⋆ The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

cost is imposed on the Attacker based on the look-angle of the Turret in addition to a cost based on the amount of time it takes the Attacker to come within the capture distance, d_c, of the Turret. The Turret seeks to maximize the Attacker's cost and does so by turning to keep the Attacker in its line of sight. Three singular surfaces are present in the TDDG: (1) the Turret Dispersal Surface (TDS), (2) the Turret Universal Surface (TUS), and (3) the Attacker Dispersal Surface (ADS) (Akilan and Fuchs, 2017). These names describe the type of the surface as well as which agent has control authority on the surface. In the case of the Dispersal Surfaces, the agent with control authority must make a choice between two (or more) optimal actions. Here, the Value function, which gives the saddle-point equilibrium cost of the game for a particular initial condition, is not differentiable (Başar and Olsder, 1982). The Value function is also not differentiable on the Turret Universal Surface; and in this case, the Turret's optimal control is not defined. Fortunately, both the Turret Dispersal Surface and the Turret Universal Surface can be handled analytically, as was done in Akilan and Fuchs (2017). Neither of these surfaces are affected by the parameter settings as they arise due to inherent symmetries in the game. Our focus, in this paper, is on the Attacker Dispersal Surface which does not have an analytical expression and thus must be treated numerically (see Patsko et al., 2018, §2.5.1). In particular, we show that the parameters' settings affect not only the shape of the surface but also whether or not the surface is even present. The numerical process we develop to accomplish this task is enabled by collapsing the five natural parameters into two composite parameters.

The paper is organized as follows. Section 2 contains the problem formulation and details the conversion from the natural parameter space to the collapsed parameter space. Section 3 formally describes Dispersal Surfaces and includes necessary and sufficient conditions which are used in the following section. Section 4 details two different numerical approaches used to characterize the Attacker Dispersal Surface. Section 5 concludes the paper and discusses future research directions.

2. KINEMATICS AND SCALING

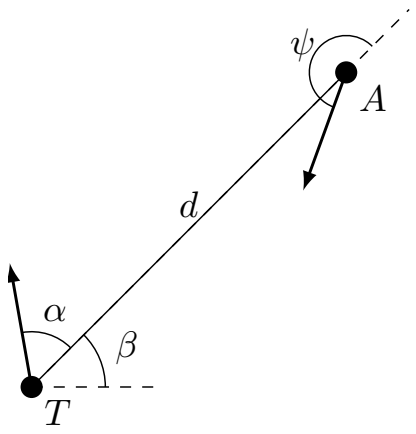

Fig. 2. Coordinate system

Figure 2 shows the coordinate system used throughout the remainder of the paper. Note this is a relative coordinate system. As presented in Akilan and Fuchs (2017), the coordinate system cannot be reduced further than three. Note, the 'global rotation' of the system, β, is necessary for the conversion of the relative states back to the natural states (i.e. Cartesian coordinates of both agents), but not necessary for computing the equilibrium trajectories of the other states. The motivation to reduce the parameter set is similar to the motivation for reducing the state dimension. Different parameter sets may yield quite different player behaviors across the state space. Characterizing these different regions of optimal play over the space of possible parameter settings is an important part of solving the game (in a full sense). Doing so for large (> 3) numbers of parameters may be difficult or intractable. Fortunately, it is often possible to reduce the number of parameters via various scalings and non-dimensional quantities.

For inspiration, consider the famous Buckingham π Theorem, which is based upon unit compatibility, and is used (particularly within the aerospace community) to derive physical laws centered around non-dimensional numbers (e.g., Reynolds number, Mach number, etc.). Without an understanding of Reynolds number, $Re = \frac{\rho u L}{\mu}$, one may unnecessarily repeat experiments with different length scales (L) and speeds (u), but constant Re.

Such a collapsing of physical quantities can also be seen in the HCDG wherein we see (Fig. 1) that it is the *ratio* of capture radius to Chauffeur turn radius that is pertinent to the different solution regions. Merz (1971) characterized the entire parameter space, dividing it into 20 distinct regions with different characteristics.

Here, the collapsing of the physical parameters is facilitated by scaling time and distance. For notational convenience, the dimensional variables are barred; the non-barred variables denote the scaled quantities. The Turret Defense Differential Game kinematics were expressed by Akilan and Fuchs (2017) originally and are given here with a small correction and a simplification, along with the optimal costate kinematics:

$$\begin{aligned}
\dot{\bar{d}} &= v_A \cos\bar{\psi}, && \bar{\psi} \in [0, 2\pi] \\
\dot{\bar{\alpha}} &= \bar{\omega} - v_A \frac{1}{\bar{d}} \sin\bar{\psi}, && \bar{\omega} \in [-\Omega, \Omega] \\
\dot{\bar{\beta}} &= v_A \frac{1}{\bar{d}} \sin\bar{\psi} \\
\dot{\lambda}_{\bar{d}} &= -\lambda_{\bar{\alpha}} v_A \frac{1}{\bar{d}^2} \sin\bar{\psi} \\
\dot{\lambda}_{\bar{\alpha}} &= -c_1 \frac{1}{2} \sin\bar{\alpha} \\
\dot{\lambda}_{\bar{\beta}} &= 0.
\end{aligned} \tag{1}$$

where $\bar{d}(\bar{t}_f) = d_c$ and $\bar{d}(\bar{t}) > d_c$, $\forall \bar{t} < \bar{t}_f$ – that is, the game terminates when the attacker reaches the capture distance d_c. From (1) and the boundary conditions, the four parameters are apparent: attacker speed v_A, maximum turret slew rate Ω, capture distance d_c, and cost parameter c_1. The fifth parameter c_2 appears in the cost function (Akilan and Fuchs, 2017),

$$\bar{J} = \int_{\bar{t}_0}^{\bar{t}_f} \left(c_1 \frac{1}{2} \left(1 + \cos(\bar{\alpha}(\bar{t}))\right) + c_2 \right) \mathrm{d}\bar{t} \tag{2}$$

as well as in the equilibrium terminal value of λ_d (Akilan and Fuchs, 2017),

$$\lambda_{\bar{d}}(\bar{t}_f) = -\frac{c_1 \frac{1}{2}(1+\cos(\bar{\alpha}(\bar{t}_f))) + c_2}{v_A}. \tag{3}$$

Fact 1. The cost in (2) is already dimensionless and altering its value with an additive offset or multiplicative scaling does not change the equilibrium strategies.

The presence of the parameters in (1)–(3) suggests the need to consider each of the equations when performing any type of scaling. The methodology employed here is described more fully by Langtangen and Pedersen (1985). First, consider the following scaling of distance and time,

$$d = \frac{\bar{d}}{d_c}, \qquad t = \frac{\bar{t}}{t_c} \tag{4}$$

where d and t are the scaled/non-dimensional versions of distance and time, d_c is, as before, the capture distance, and t_c is a time constant representing a characteristic time associated with the problem. Let t_c be the time it would take the Attacker to travel a distance d_c in a straight line,

$$t_c = \frac{d_c}{v_A} \tag{5}$$

Note that the independent variable, time, has been scaled, and all of the states are now essentially dimensionless since α and β are angles. Scaling in time affects all of the states in (1), but scaling in distance requires special care in obtaining $\dot{d}$. The closing velocity can now be expressed in terms of non-dimensional distance and time,

$$\begin{aligned} \dot{\bar{d}} = \frac{\mathrm{d}\bar{d}}{\mathrm{d}\bar{t}} = \frac{\mathrm{d}(dd_c)}{\mathrm{d}(tt_c)} = \frac{d_c}{t_c}\frac{\mathrm{d}d}{\mathrm{d}t} = \frac{d_c}{t_c}\dot{d} \\ \implies \dot{d} = t_c \frac{1}{d_c}\dot{\bar{d}} \end{aligned} \tag{6}$$

For α and β states we have,

$$\begin{aligned} \dot{\bar{\theta}} = \frac{\mathrm{d}\bar{\theta}}{\mathrm{d}\bar{t}} = \frac{\mathrm{d}\bar{\theta}}{\mathrm{d}(tt_c)} = \frac{1}{t_c}\frac{\mathrm{d}\theta}{\mathrm{d}t} \\ \implies \dot{\theta} = t_c\dot{\bar{\theta}}, \qquad \theta = \alpha, \beta \end{aligned} \tag{7}$$

as a form for the dynamics. The relationship between the scaled (non-barred) versions of the angles and their natural counterparts is simply,

$$\theta(t) = \bar{\theta}(\bar{t}), \qquad \theta = \alpha, \beta \tag{8}$$

Substituting (1), (4), and (5) into (6) and (7) yields the following scaled state kinematics,

$$\begin{aligned} \dot{d} &= t_c \frac{1}{d_c} v_A \cos\psi = \cos\psi, && \psi \in [0, 2\pi], \\ \dot{\alpha} &= t_c\left(\bar{\omega} - \frac{v_A}{\bar{d}}\sin\psi\right) = \omega - \frac{1}{d}\sin\psi, && \omega \in [-\rho, \rho], \\ \dot{\beta} &= t_c v_A \frac{1}{\bar{d}}\sin\psi = \frac{1}{d}\sin\psi, \end{aligned} \tag{9}$$

where,

$$\rho = \frac{\Omega d_c}{v_A}. \tag{10}$$

Concerning the c_1 and c_2 parameters, let $c_1 = 1$ without loss of generality (see Fact 1), and $c_2 = c$. The "natural" cost $\bar{J}$ may be recovered by $\bar{J} = c_1 J$. Define the set of admissible initial conditions as

$$\mathbf{X} := \{\mathbf{x} = (d, \alpha, \beta) \;\mid\; d \geq 1\}. \tag{11}$$

Now, the scaled Hamiltonian is written, using new, non-barred versions of the costates, as,

$$\begin{aligned} \mathscr{H} = \lambda_d \cos\psi + \lambda_\alpha\left(\omega - \frac{1}{d}\sin\psi\right) + \\ \lambda_\beta \frac{1}{d}\sin\psi - \frac{1}{2}(1+\cos\alpha) - c \end{aligned} \tag{12}$$

The non-barred costate kinematics are then obtained by taking the partial of $\mathscr{H}$ w.r.t. each (scaled) state,

$$\begin{aligned} \dot{\lambda}_d &= -\frac{\partial\mathscr{H}}{\partial d} = (\lambda_\beta - \lambda_\alpha)\frac{1}{d^2}\sin\psi, \\ \dot{\lambda}_\alpha &= -\frac{\partial\mathscr{H}}{\partial\alpha} = -\frac{1}{2}\sin\alpha, \\ \dot{\lambda}_\beta &= -\frac{\partial\mathscr{H}}{\partial\beta} = 0. \end{aligned} \tag{13}$$

To obtain the saddle point control strategies the Hamiltonian is minimized and maximized w.r.t. ω and ψ, respectively. Incidentally, the expressions to follow are identical to those given by Akilan and Fuchs (2017) except with scaled/non-barred versions of the variables in place.

$$\omega^* = \arg\min_{\omega} \mathscr{H} = -\rho\,\mathrm{sign}\,\lambda_\alpha \tag{14}$$

$$\begin{aligned} \psi^* = \arg\max_{\psi} \mathscr{H} \\ \implies \cos\psi^* = \frac{\lambda_d}{\sigma_A}, \qquad \sin\psi^* = \frac{-\lambda_\alpha}{d\sigma_A}, \end{aligned} \tag{15}$$

where,

$$\sigma_A = \sqrt{\lambda_d^2 + \left(\frac{\lambda_\alpha}{d}\right)^2}. \tag{16}$$

Another consequence of the scaling in distance is that the terminal condition is,

$$\Gamma(\mathbf{x}) := d - 1, \tag{17}$$

where $\mathbf{x} := (d, \alpha, \beta)^\top$. Termination occurs when the state enters the terminal surface,

$$\mathscr{C} := \{\mathbf{x} \mid \Gamma(\mathbf{x}) = 0\}. \tag{18}$$

Using the Lagrange multiplier ν, the adjoined terminal Value function is written,

$$\Phi(\mathbf{x}_f) = \nu\Gamma(\mathbf{x}_f) = \nu(d-1). \tag{19}$$

Just as was done by Akilan and Fuchs (2017), differentiating (19) w.r.t. the states yields the terminal costate variables,

$$\begin{aligned} \lambda_d(t_f) &= \frac{\partial\Phi(\mathbf{x}_f)}{\partial d} = \nu, \\ \lambda_\alpha(t_f) &= \frac{\partial\Phi(\mathbf{x}_f)}{\partial\alpha} = 0, \\ \lambda_\beta(t_f) &= \frac{\partial\Phi(\mathbf{x}_f)}{\partial\beta} = 0. \end{aligned} \tag{20}$$

Because the game is independent of time, the Hamiltonian, (12), must be equal to zero at all time. Substituting the equilibrium control strategies, (14) and (15), and terminal costate values, (20), into (12) and setting time to t_f gives

$$\mathscr{H}^*\big|_{t=t_f} = |\nu| - \frac{1}{2}(1+\cos\alpha_f) - c = 0. \tag{21}$$

Solving for ν gives

$$|\nu| = \frac{1}{2}(1+\cos\alpha_f) + c. \tag{22}$$

All together, (9), (13)–(16), (20), and (22) fully express the regular solutions (i.e., the non-singular saddle point equilibrium dynamics) of the TDDG in the scaled space.

Of primary importance is the fact that the five natural parameters (v_A, d_c, Ω, c_1, and c_2) have been replaced with two: ρ and c. In a general sense, ρ represents the control authority of the Turret w.r.t. the Attacker: increasing ρ favors the Turret (in terms of Value, given some initial condition), whereas decreasing ρ favors the Attacker. The cost parameter, c, functions in much the same way as in the original (natural) representation of the game – it is merely a weight affecting the relative importance of time versus the Turret-induced cost to the Attacker. For small settings of c, the Attacker will prefer to avoid the Turret's gaze even if it takes longer to reach termination (and vice versa for large settings of c). Akilan and Fuchs (2017) considered one particular set of natural parameters throughout: $d_c = 1$, $v_A = 1$, $\Omega = 0.05$, $c_1 = 1$, and $c_2 = 0.01$. This set corresponds to $\rho = 0.05$ and $c = 0.01$, which we will henceforth refer to as the canonical parameters for the TDDG. The process of filling the state space with equilibrium trajectories is mostly standard for the TDDG: backwards integration from the terminal set and from points along the Turret Universal Surface. However, the ADS requires special care as it does not have an analytical expression.

3. GENERAL DISPERSAL SURFACE CHARACTERISTICS

In this section, we formally state the characteristics of a Dispersal Surface and establish criteria to be used in the computation of the ADS. First, we define the Value function as

$$\begin{aligned} V(\mathbf{x}(t)) &= \min_{\psi} \max_{\omega} J(\mathbf{x}(t); \psi(t), \omega(t)) \\ &= \max_{\omega} \min_{\psi} J(\mathbf{x}(t); \psi(t), \omega(t)) \\ &= \max_{\omega} \min_{\psi} \int_t^{t_f} \left(\frac{1}{2} (1 + \cos \alpha(\tau)) + c \right) \mathrm{d}\tau \,. \end{aligned} \tag{23}$$

The Dispersal Surface is characterized by one (or both) of the agents' equilibrium actions being non-unique (Isaacs, 1965). That is, the agent whose equilibrium control is non-unique may choose which equilibrium action to take, leaving the Value of the game unaffected. Let $\mathbf{X}_{DS}$ denote the set of points on the Dispersal Surface. Let this be a Dispersal Surface in which the Attacker's equilibrium action is non-unique, but the Turret's equilibrium action is uniquely defined. Let ψ_A and ψ_B represent two different headings the Attacker can choose at particular state $\mathbf{x}'$.

Criterion 1. The condition

$$V(\mathbf{x}'(t)) = J(\mathbf{x}'(t); \psi_A(t), \omega^*(t)) = J(\mathbf{x}'(t); \psi_B(t), \omega^*(t)) \tag{24}$$

where $\omega^* = \arg\max_{\omega} J(\mathbf{x}'(t); \psi(t), \omega(t))$ is necessary for $\mathbf{x}' \in \mathbf{X}_{DS}$ to hold.

That is, the Dispersal Surface is characterized by two equilibrium trajectories intersecting at the same state with the same Value. Because the system dynamics are autonomous (i.e. not time-dependent), it is not necessary for this intersection to occur at the same time.

4. ATTACKER DISPERSAL SURFACE

In this section we focus specifically on the Attacker Dispersal Surface in the TDDG. At a conceptual level, the ADS arises due to the interplay between the $(1 + \cos \alpha)$ and the c terms of the cost functional's integrand. On the ADS, the Attacker has a choice between taking a quicker route in which the $(1 + \cos \alpha)$ term's contribution to the cost is higher, and a more round-about route in which the c term's contribution to the cost is higher. We refer to these as the direct (D) and indirect (I) paths, respectively. Based on the analysis in the previous section, we develop two numerical procedures to characterize the ADS. The first procedure computes the extent of the state space that is covered by equilibrium trajectories emanating from (in forward time) the ADS. This procedure provides the primary evidence for the presence or absence of the ADS, depending on the parameters. The second procedure computes the surface itself and is included here to confirm the results of the first procedure. Note that because of the lack of an analytical expression for the ADS both procedures are necessary to fill the state space with equilibrium trajectories, which is synonymous with solving the TDDG for particular parameter settings.

Before proceeding into the discussion of the procedures themselves, we include some background on the other singular surfaces and their interactions with the ADS. Akilan and Fuchs (2017) showed that the Turret Dispersal Surface is defined by

$$\mathbf{X}_{TDS} := \{\mathbf{x} \mid \alpha = \pi\} \,, \tag{25}$$

which is the configuration in which the Turret is looking directly away from the Attacker. In this symmetrical configuration, the Turret has the choice of turning clockwise or counter-clockwise at its maximum turn rate: $\omega^* = \pm\rho$ – both choices are optimal. Upon the Turret making a choice, the state of the system immediately departs the TDS and both agents' equilibrium controls are uniquely defined. Akilan and Fuchs (2017) also showed that the Turret Universal Surface is defined by

$$\mathbf{X}_{TUS} := \{\mathbf{x} \mid \alpha = 0\} \,. \tag{26}$$

On the TUS, the Turret is looking directly at the Attacker; the Turret's control is $\omega^* = 0$ and the Attacker heads directly towards the Turret, $\psi^* = \pi$. The agents remain in this configuration for the remainder of the game – neither gains an advantage by deviating from this "locked on" configuration.

Note that the TUS does not interact with the ADS, but its tributaries must also be computed in order to fill the state space with equilibrium trajectories. The TDS is particularly important in the present context because it coincides with the ADS.

4.1 ADS Envelope Computation

In this section, we present a procedure for computing the envelope of the ADS. Because of the symmetrical nature of the TDDG's solution (c.f Akilan and Fuchs, 2017), we can, without loss of generality, consider the part of the state space in which $0 \leq \alpha \leq \pi$. For the remainder of the paper, we continually make use of the solution of the TDDG (9), (13)–(16), (20) and (22) to backwards integrate from the terminal surface (18). This backwards integration is carried out until the trajectory reaches the TDS, where $\alpha = \pi$ since the trajectories can go no further and we are interested in computing the maximal envelope of the ADS.

We define the following mapping between points on the terminal surface and points along the trajectory obtained by backwards integration of the solution:

$$B : \alpha_f, t_f \rightarrow \mathbf{x}_0, \quad \text{s.t. (9), (13)–(16), (20) and (22),} \tag{27}$$

where $\mathbf{x}_0 = \mathbf{x}(0)$. Making use of (24) we define the envelope of the ADS as the pair

$$(\alpha_{f_L}, \alpha_{f_U}) \text{ s.t. } \begin{cases} d_{0_L} = d_{0_U}, \\ \alpha_{0_L} = \alpha_{0_U} = \pi, \\ \alpha_{f_L} < \alpha_{f_U}, \\ V(\mathbf{x}_{0_L}) = V(\mathbf{x}_{0_U}) \end{cases} \tag{28}$$

where $\mathbf{x}_{0_L} = B(\alpha_{f_L}, t_{f_L}) = [d_{0_L}\ \alpha_{0_L}\ \beta_{0_L}]^\top$, etc. In general, t_{f_L} and t_{f_U} are not known *a priori*. However, this is of little consequence since we can simply integrate trajectories backwards in time until the condition $\alpha = \pi$ is met. Indeed, the trajectories ought not go any further, due to the TDS. The purpose of defining and computing (28) is twofold. First, for $\alpha_{f_L} \leq \alpha_f \leq \alpha_{f_U}$ the trajectories terminate (in retrograde time) on the ADS while for $\alpha_f < \alpha_{f_L}$ and $\alpha_f > \alpha_{f_U}$ the trajectories terminate on the TUS and do not interact with the ADS. Second, when $\alpha_{f_L} = \alpha_{f_U}$, i.e., there is no solution to (28) the ADS is not present. We state the following, without proof:

Proposition 1. For the Attacker Dispersal Surface to exist in the solution of the Turret Defense Differential Game, (28) must be satisfied.

That is, if the ADS exists in the solution, we must be able to compute the ADS's envelope $(\alpha_{f_L}, \alpha_{f_U})$.

Procedure Now, we describe a process by which the ADS envelope, (28), is computed. Figures 3 and 4 contain the results for this procedure which are illustrative for describing the procedure itself. First, the terminal surface, (18), is swept along a grid of α_f, $0 < \alpha_f < \pi$. The mapping B is computed for each α_f by backwards integration until $\alpha = \pi$, and the corresponding d_0 and V values are recorded. Figure 3 shows the d_0 values along the horizontal axis associated with the α_f values. From (28), we require that we have two different α_f with the same d_0. Thus

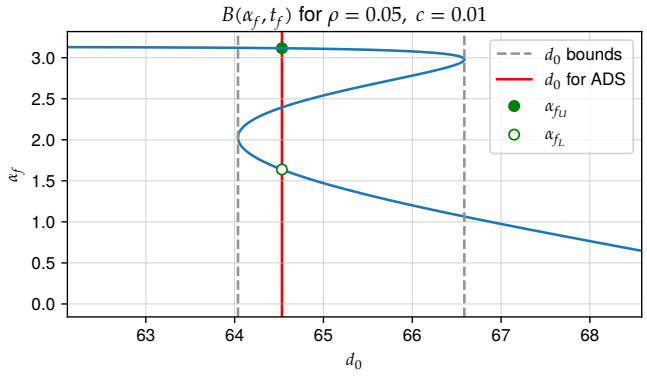

Fig. 3. Mapping of α_f to d_0 along the TDS indicating the region of d_0 (dashed gray lines) that may satisfy the ADS envelope conditions and the d_0 and $(\alpha_{f_L}, \alpha_{f_U})$ for which the Value of the L and U trajectories are equal. Initially 4,000 samples were used in the α_f sweep, and then 1,000 samples were used in the d_0 sweep.

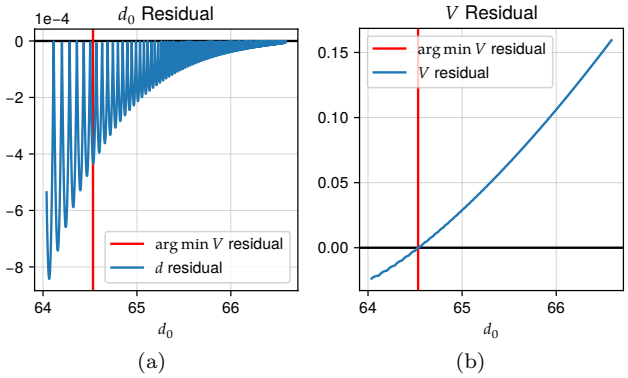

Fig. 4. Residuals between the L and U trajectories corresponding to Fig. 3.

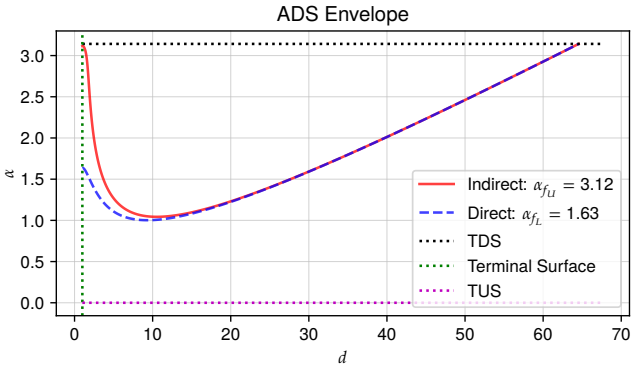

Fig. 5. Depiction of the ADS envelope in the state space for $\rho = 0.05$ and $c = 0.01$. Equilibrium trajectories emanating from the ADS fill the space between the red and blue curves (i.e. the upper and lower extent of the ADS envelope).

the gray dashed lines in the figure denote the bounds for which this condition can be satisfied. All of the d_0 within this range satisfy the first three conditions in (28). Satisfaction of the last condition, $V(\mathbf{x}_{0_L}) = V(\mathbf{x}_{0_U})$, is the distinguishing criteria by which the ADS envelope is determined. Then, a line of constant d_0 (i.e., the red line in the figure) is swept within the bounds. Because we started with a grid of α_f, the spacing of points in the d_0 axis is non-uniform. A new grid over the d_0 is used and the backwards "S" curve is interpolated to get approximate candidate values for α_{f_L} and α_{f_U}. The mapping B is recomputed for the approximate candidate α_f values and the initial distance residual $d_{0_L} - d_{0_U}$ (Fig. 4a) and Value residual $V(\mathbf{x}_{0_L}) - V(\mathbf{x}_{0_U})$ (Fig. 4b) are recorded. Note the d_0 residual is larger for smaller d_0 because the upper part of the curve is very flat here – a small error in α_{f_U} produces a large deviation in d_{0_U}. Finally, when the number of d_0 samples is sufficiently large, the Value residual in Fig. 4b is fairly smooth and monotonic with a unique zero crossing (marked with the red line). This value for d_0, and the corresponding $(\alpha_{f_L}, \alpha_{f_U})$ satisfies (28) and thus defines the envelope of the ADS. Figure 5 shows the ADS envelope trajectories in the state space along with the terminal surface, the TDS, and the TUS.

Remarks Note that the ADS envelope conditions in (28) can be determined by setting up an appropriate nonlinear

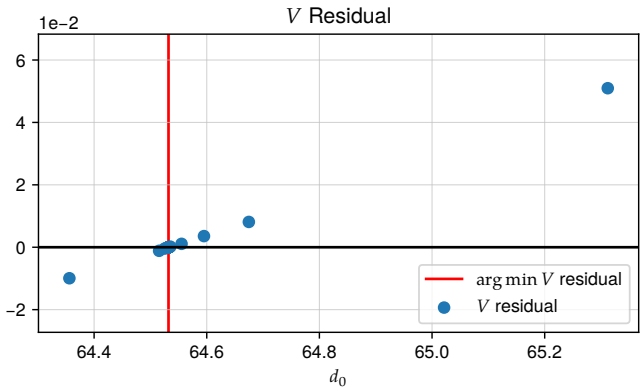

Fig. 6. Value residual for the binary search algorithm.

program (NLP). However, from our experience, it can be difficult to enforce the condition $\alpha_{f_L} < \alpha_{f_U}$ as the optimizer tends to end up in the situation where $\alpha_{f_L} = \alpha_{f_U}$ and dwells there. Also, the second stage of the procedure, in which d_0 is swept, may be replaced by a binary search process due to the monotonicity of the Value residual w.r.t. d_0. This drastically reduces the number of backwards shots needed in the second stage. For example, to obtain a Value residual $< 1 \times 10^{-12}$ only 37 backwards shots are needed for the example in Fig. 3. In this way, the Value residual tolerance can be directly specified, and the d_0 residual is directly affected by the number of samples in the initial α_f grid. Figure 6 shows the result of using binary search to find the zero crossing of the Value residual.

The trajectories generated within the range $\alpha_{f_L} < \alpha_f < \alpha_{f_U}$ do not actually reach back to the TDS, since this would require passing through the ADS. Therefore, the middle section of the backwards "S" curve in Fig. 3 is meaningless for our purposes. Algorithm 1 summarizes the procedure used to compute the ADS envelope $(\alpha_{f_L}, \alpha_{f_U})$.

Algorithm 1 ADS Envelope (with binary search)

$d_0 \leftarrow$ empty vector
for α_{f_i} **in** α_f grid **do**
  $d_{0_i} \leftarrow B(\alpha_{f_i}, t_f)$ ▷ backwards integrate
end for
$d_{0_{\min}}$, $d_{0_{\max}} \leftarrow$ local min & max of d_0
$V_{\text{resid}} \leftarrow \infty$
while $V_{\text{resid}} > \varepsilon$ **do** ▷ binary search
  $d_{\text{mid}} \leftarrow (d_{0_{\min}} + d_{0_{\max}})/2$
  $\alpha_{f_L}, \alpha_{f_U} = B^{-1}(d_{\text{mid}})$ ▷ see Fig. 3
  $\mathbf{x}_{0_L} \leftarrow B(\alpha_{f_L}, t_f)$ ▷ backwards integrate
  $\mathbf{x}_{0_U} \leftarrow B(\alpha_{f_U}, t_f)$ ▷ backwards integrate
  $V_{\text{resid}} \leftarrow V(\mathbf{x}_{0_L}) - V(\mathbf{x}_{0_U})$
  if $V_{\text{resid}} > 0$ **then**
    $d_{0_{\max}} \leftarrow d_{\text{mid}}$
  else
    $d_{0_{\min}} \leftarrow d_{\text{mid}}$
  end if
end while

4.2 Different Parameters

In this section, we demonstrate the utility of the previously described numerical procedure in determining the regions of the parameter space in which the ADS is present. As

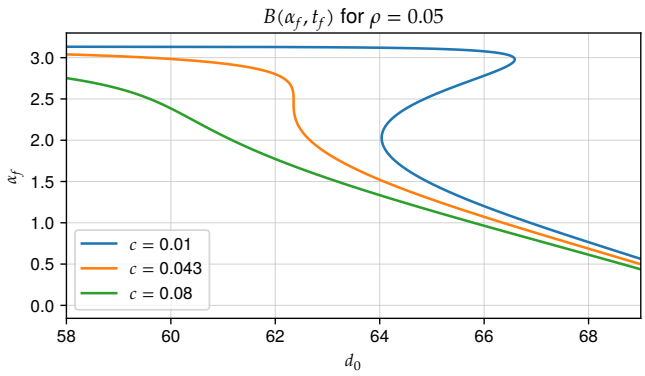

Fig. 7. Mapping of α_f to d_0 for different settings of the cost parameter c.

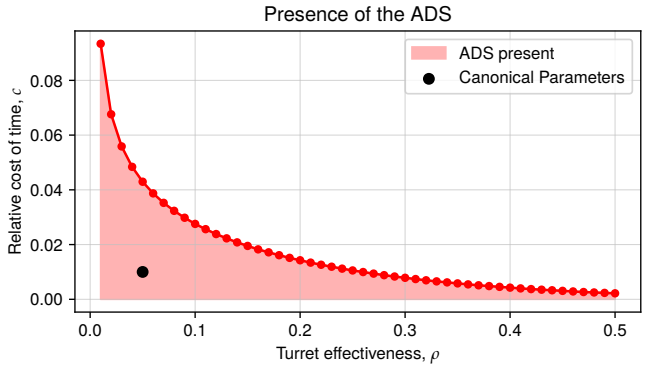

Fig. 8. Presence of the ADS in the solution of the TDDG over the parameter space.

Proposition 1 suggests, the inability for the procedure to find $(\alpha_{f_L}, \alpha_{f_U})$ that satisfies (28) indicates that there is no ADS in the solution to the TDDG. Figure 7 shows the results of the backwards integration process (i.e. the backwards "S" curves) for different settings of the cost parameter c for a particular Turret effectiveness setting, ρ. For $c = 0.043$, the curve flattens out to the point that for each d_0 there is a unique α_f. Thus, the conditions for the ADS envelope (28), in particular the requirements that $\alpha_{f_L} < \alpha_{f_U}$ and $d_{0_L} = d_{0_U}$, cannot be met. It is precisely at this setting of c, i.e., where there exists a point on the curve s.t. $\mathrm{d}d_0/\mathrm{d}\alpha_f = 0$, that the ADS disappears from the solution of the TDDG. As suggested by the curve in Fig. 7 corresponding to $c = 0.08$, the ADS is absent for all $c > 0.043$, in this case. Repeating the process across a range of ρ yields the curve in Fig. 8.

4.3 ADS Termination Point Computation

In order to gain a better understanding of the ADS and how it is affected by the parameter settings, we develop here a procedure for computing the surface itself. We have observed that one of the endpoints of the ADS (if it exists) lies on the TDS. Akilan and Fuchs (2017) showed that, for the canonical parameter settings, the other endpoint of the ADS is *not* coincident with any other surface. The approach for computing points on the ADS is based upon solving for a pair of trajectories satisfying

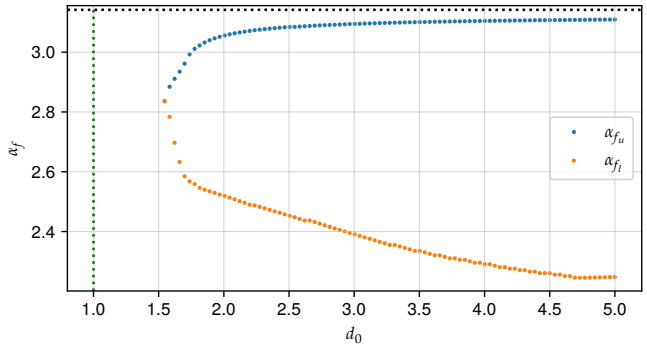

Fig. 9. Solutions to (29) over a d_0 grid for $c = 0.01$.

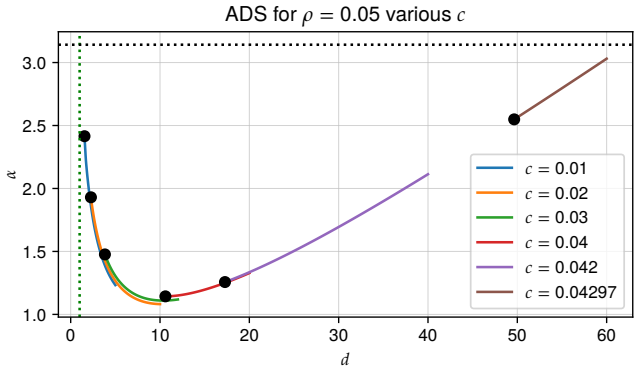

Fig. 10. ADS for $\rho = 0.05$ and various c showing the recession of the ADS into the TDS as $c \to 0.043$.

$$(\alpha_{f_l}, \alpha_{f_u}) \text{ s.t. } \begin{cases} d_{0_l} = d_{0_u} = d_0, \\ \alpha_{0_l} = \alpha_{0_u} = \alpha_0, \\ \alpha_{f_l} < \alpha_{f_u}, \\ V(\mathbf{x}_{0_l}) = V(\mathbf{x}_{0_u}) \end{cases} \tag{29}$$

where d_0 is the specified distance associated with the point on the ADS for which we are solving. The condition (29) is similar to (28) but with a free α_0. Again, the t_f associated with these trajectories is not known *a priori*. However, using integrator callbacks (c.f. Rackauckas and Nie, 2017), we can integrate backwards until the specified distance d_0 is reached. To solve for $(\alpha_{f_l}, \alpha_{f_u})$, we employ an NLP solver which seeks to minimize the following residual

$$r = \left\| \begin{matrix} V(\mathbf{x}_{0_l}) - V(\mathbf{x}_{0_u}) \\ \alpha_{0_l} - \alpha_{0_u} \end{matrix} \right\|$$

To generate a series of points on the ADS, we solve (29) along a grid of d_0. We begin with some appropriate d_0 and proceed in decreasing order of d_0 along this grid. A homotopy approach is used in which the NLP solver is given the $(\alpha_{f_l}, \alpha_{f_u})$ of the previous d_0 as an initial guess. Figure 9 contains the results of this procedure for a relative time cost setting $c = 0.01$. Note on the left side of the plot the two curves quickly converge s.t. $\alpha_{f_l} \not< \alpha_{f_u}$. Thus for $d_0 \leq 1.5$ the condition $\alpha_{f_l} < \alpha_{f_u}$ in (29) cannot be satisfied. The leftmost d_0 point corresponds to the non-TDS endpoint of the ADS. For each solution in Fig. 9, we record the α_0 to which the upper and lower trajectories integrate back to. The points (d_0, α_0) represent the ADS itself. We repeat the procedure for different settings of c up until $c \approx 0.043$ whereupon, based on Fig. 8, we expect the ADS to disappear. Figure 10 shows the Attacker Dispersal Surfaces for $\rho = 0.05$ and several different c. As c is increased from 0.01 to ≈ 0.43, the endpoint of the ADS recedes away from the terminal surface and towards the TDS. These results corroborate our earlier statements about the disappearance of the ADS from the solution of the TDDG for high settings of c. Algorithm 2 summarizes the process for computing the ADS once for a particular setting of c.

Algorithm 2 ADS Computation

$\mathbf{X}_{ADS} \leftarrow \emptyset$
$i \leftarrow$ LENGTH (d_0 grid)
$\alpha_{f_l} \leftarrow 0$ and $\alpha_{f_u} \leftarrow \pi$
$guess \leftarrow (\alpha_{f_l}, \alpha_{f_u})$
while $\alpha_{f_u} - \alpha_{f_l} > \varepsilon$ and $i \geq 0$ **do**
  $d_0 \leftarrow d_0$ grid $[i]$
  $\alpha_{f_l}, \alpha_{f_u}, \alpha_0 \leftarrow$ solution to (29) with $guess$ ▷ NLP
  if $\alpha_{f_u} - \alpha_{f_l} > \varepsilon$ **then**
    $\mathbf{X}_{ADS} \leftarrow \mathbf{X}_{ADS} \cup \{(d_0, \alpha_0)\}$
    $guess \leftarrow (\alpha_{f_l}, \alpha_{f_u})$
  end if
  $i \leftarrow i - 1$
end while

5. CONCLUSION

In this paper we have analyzed the Turret Defense Differential Game – a game with three states, only two of which affect the solution. Despite the simplicity of its dynamics, the full solution of the game is difficult to obtain due to the presence of a non-analytical singular surface: the Attacker Dispersal Surface. Numerical methods are required to characterize the Attacker Dispersal Surface as well as to compute the surface itself. Furthermore, the original description of the game contains five natural parameters, which makes understanding the impact of parameters on the solution (particularly the Attacker Dispersal Surface) difficult. We have collapsed the five natural parameters into two composite parameters: the Turret effectiveness, ρ, and the relative time cost c. The scaling of the kinematics facilitated the exploration of the parameter space. We developed two numerical approaches, based on the general definition of a dispersal surface, to characterize the Attacker Dispersal Surface. These approaches allowed us to expose an interesting feature of the game's solution: the Attacker Dispersal Surface is only present over a portion of the parameter space.

In general, analysis of non-analytic singular surfaces, even relatively "benign" types as in the dispersal surface, is difficult. The numerical approaches developed here make extensive use of the unique aspects of the game and are built upon existing general purpose algorithms such as numerical integration, root finding, binary search, nonlinear programming, etc. An interesting research direction would be to generalize some of these concepts to better address non-analytic singular surfaces in other games.

ACKNOWLEDGEMENTS

This paper is based on work performed at the Air Force Research Laboratory (AFRL) *Control Science Center of Excellence*. Distribution Unlimited. 29 Aug 2019. Case #88ABW-2019-4250.

REFERENCES

Akilan, Z. and Fuchs, Z. (2017). Zero-sum turret defense differential game with singular surfaces. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2041–2048. doi:10.1109/CCTA.2017.8062754.

Başar, T. and Olsder, G.J. (1982). *Dynamic Noncooperative Game Theory*, volume 160 of *Mathematics in Science and Engineering*. Elsevier, 2nd edition. URL `https://www-sciencedirect-com.wrs.idm.oclc.org/science/journal/00765392/160/supp/C`.

Breakwell, J.V. and Bernhard, P. (1990). A simple game with a singular focal line. *Journal of Optimization Theory and Applications*, 64, 419–428. doi:10.1007/bf00939457. URL `http://dx.doi.org/10.1007/bf00939457`.

Isaacs, R. (1965). *Differential Games: A Mathematical Theory with Applications to Optimization, Control and Warfare*. Wiley, New York.

Langtangen, H.P. and Pedersen, G.K. (1985). *Scaling of Differential Equations*. Springer-Verlag. URL `https://hplgit.github.io/scaling-book/doc/pub/book/pdf/scaling-book-4print.pdf`.

Lewin, J. (1994). *Differential Games: Theory and Methods for Solving Game Problems with Singular Surfaces*. Springer-Verlag London Limited.

Melikyan, A.A. (1994). Singular paths in differential games with simple motion. 125–135. URL `https://link-springer-com.wrs.idm.oclc.org/chapter/10.1007/978-1-4612-0245-5_7`.

Merz, A.W. (1971). *The Homicidal Chauffeur - A Differential Game*. Ph.D. thesis, Stanford.

Patsko, V., Kumkov, S., and Turova, V. (2018). *Pursuit-Evasion Games*, 1–87. Springer International Publishing. doi:10.1007/978-3-319-27335-8_30-2. URL `http://dx.doi.org/10.1007/978-3-319-27335-8_30-2`.

Rackauckas, C. and Nie, Q. (2017). Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5. doi:10.5334/jors.151. URL `http://dx.doi.org/10.5334/jors.151`.