3-14-2014

Fragment Capture Simulation for MANPADS Test Arena Optimization

Michael J. Garee

FRAGMENT CAPTURE SIMULATION FOR MANPADS TEST ARENA OPTIMIZATION

THESIS

Michael J. Garee, 1st Lt, USAF

AFIT-ENS-14-M-09

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-14-M-09

FRAGMENT CAPTURE SIMULATION FOR MANPADS TEST ARENA OPTIMIZATION

THESIS

Presented to the Faculty

Department of Operations Research

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Michael J. Garee, B.S.

1st Lt, USAF

March 2014

AFIT-ENS-14-M-09

FRAGMENT CAPTURE SIMULATION FOR MANPADS TEST ARENA OPTIMIZATION

Michael J. Garee, B.S.
1st Lt, USAF

Approved:

//signed//	20 Feb 2014
Raymond R. Hill, PhD (Chairman)	Date
//signed//	20 Feb 2014
Darryl Ahner, PhD (Member)	Date

AFIT-ENS-14-M-09

Abstract

The assessment of aircraft survivability against explosive munitions is an expensive undertaking. Test articles for both aircraft and weapon are scarce due to their high costs, leading to a limited supply of test data. The development of newer, hopefully more effective weaponry and protection measures continues despite the short supply of test data. Therefore, test organizations need to explore methods for increasing the quality of test results while looking for ways to decrease the associated costs. This research focuses on the Man-Portable Air-Defense System (MANPADS) as the weapon of choice and live-fire arena testing as the experimental data source. A simulation infrastructure is built and used to examine how to optimize the arena configuration to maximize the test information obtained. Several research questions are explored: measuring potential data quality, comparing arena designs, and improving arena configurations based on fragment pattern predictions.

Table of Contents

Page

List of Figures

Figure Page

List of Tables

FRAGMENT CAPTURE SIMULATION FOR MANPADS TEST ARENA OPTIMIZATION

I. Introduction

The assessment of aircraft survivability against explosive munitions is an expensive undertaking. Test articles for both aircraft and weapon are scarce due to their high costs, leading to a limited supply of test data. The development of newer, hopefully more effective weaponry and protection measures continues despite the short supply of test data. Therefore, test organizations should explore methods for increasing the quality of test results while looking for ways to decrease the associated costs. This research focuses on the Man-Portable Air-Defense System (MANPADS) as the weapon of choice and live-fire arena testing as the experimental data source.

1.1 Background

1.1.1 MANPADS overview.

MANPADS are a class of shoulder-fired anti-aircraft missiles measuring one to two meters in length and weighing 13 to 25 kilograms (Bureau of Political-Military Affairs, 2011). Examples of MANPADS include the SA-7b and the SA-14. A MANPADS missile is guided, unlike the superficially similar but unguided rocket-propelled grenade (RPG), and is best-suited for use against ground targets or low-flying aircraft. The most common guidance system for MANPADS is infrared seekers, though laser-guided and command line-of-sight systems also exist (Schmitt, 2013). MANPADS are designed to destroy helicopters and small aircraft and can be operated by an individual or a small team (Schroeder, 2007). The ease with which these weapons are transported and hidden

facilitates their proliferation and makes them attractive weapons for criminal and terrorist activities. An estimated 500,000-750,000 MANPADS remain stockpiled worldwide, making them a persistent threat to both military and civilian aircraft (Schroeder, 2007).

According to the U.S. State Department's Bureau of Political-Military Affairs, curbing the spread of MANPADS is a top priority for national security (Bureau of Political-Military Affairs, 2011). The State Department reports that forty civilian aircraft were hit by MANPADS between 1975 and 2011, causing over 800 deaths worldwide. U.S. civilian airliners have never been targeted in a missile attack, but a 2005 RAND Corporation study concluded that "... one anti-aircraft missile purchased for as little as a few thousand dollars on the black market could kill hundreds of people and cause economic damage exceeding $16 billion" (Schmitt, 2013). Unclassified data on U.S. military aircraft losses caused by MANPADS is not available.

The Department of Defense and the Department of Homeland Security outline four methods of protection for U.S. aircraft against MANPADS (Czarnecki et al., 2008):

1. "Denial of weapons to potential threat organizations and individuals
2. "Denial of opportunity to fire the weapon at an aircraft
3. "Prevention of impact of the missile on the aircraft
4. "Withstanding MANPADS impacts and landing the aircraft without system loss or casualties."

This study supports the fourth point by exploring data collection and analysis techniques useful for aircraft survivability assessments.

1.1.2 Aircraft survivability testing.

"What is the potential for an aircraft kill given a hit?" is the key question in aircraft survivability analysis and is supported by the answers to other more fundamental questions (Czarnecki et al., 2011a). These questions cover survivability aspects such as the amount

of blast damage sustained by an aircraft and the penetration depth of missile fragments. These questions must be answered using Test & Evaluation (T&E) methods. Live-fire testing using actual aircraft and weapons produces the most realistic results, but comes at the highest resource cost. Costs can be reduced by using representative test articles, such as helicopter tail boom skeletons instead of production-quality helicopters (Calapodas, 2001).

Simulation software can offer further cost savings, although developing and maintaining complex software can become expensive. Simulations also require accurate test data and sufficient mathematical models to produce reliable results. Physical testing can be used to produce simulation inputs directly, or it can aid in developing models to use within a simulation. For example, MANPADS fragment mass and velocity data can be captured in a test arena with no aircraft present, and the data can be supplied to simulation modelers for use in survivability simulations (Czarnecki et al., 2011a). Described in greater detail in Section 2.3, a fragment capture arena is a full or partial enclosure of an explosive weapon with walls made of catch bundles used for capturing fragments and make-screens used for calculating fragment velocities. Modelers can use physical test data to make recommendations for improving future tests, forming a feedback loop. The Department of Defense (DoD) recognizes Modeling & Simulation (M&S) as a "key enabler of DoD activities" (Under Secretary of Defense (AT&L), 2007), and Survivability/Vulnerability Information Analysis Center (SURVIAC) maintains and distributes eleven computer models with military applications, including ALARM, BRAWLER, JSEM, and their Vulnerability Tool Kit (SURVIAC, 2013). The collection of models in SURVIAC clearly indicates that computer simulations play a vital role in the survivability discipline.

1.2 Problem statement

Simulation authors have made specific requests for data improvements from physical MANPADS testing. In particular, they wish to have missile fragment masses mapped to their velocities with a higher degree of fidelity than in the past: "... complete threat model

assembly requires warhead fragment and missile body debris distribution and detailed velocity information" (Czarnecki et al., 2011a). Thus, the ideal fragment capture arena for MANPADS testing would completely surround the weapon, capturing every fragment and letting researchers build a complete three-dimensional fragment distribution. Physical and logistical constraints make a complete arena infeasible. A fully-enclosed arena would be destroyed by the blast pressure, and the man-hours required to extract and measure all weapon fragments would be cost-prohibitive. Therefore partial enclosures must be used, resulting in incomplete fragment data. Smaller arenas are more affordable in both material and labor but yield lower data quality. Establishing the existence of a "knee" in the curve of arena size versus data quality would benefit MANPADS test design by allowing test developers to make informed trade-offs between the two factors.

Each arena test will capture only a fraction of the weapon's fragment distribution as determined by arena size and weapon orientation. Survivability models benefit from having data on the full effect of the weapon, but must currently make extrapolations from limited data. Simulation and experimental design techniques could allow conduct of physical tests in a way that improves the accuracy of such projections.

For a given arena shape, there are numerous ways to place velocity-measuring make-screens on the arena walls. Smaller screens capture fewer fragments on average, giving better estimates of individual fragment velocities. Each make-screen has a material cost and requires a data channel; both money and data channels are limited resources, so an upper bound exists on the number of make-screens available for a test. Predicting fragment distributions that might be seen on the arena walls allows make-screens to be sized and placed more efficiently – smaller screens can be placed in areas of higher fragment density and larger screens in areas of lower fragment density.

Modeling the distributions of fragment mass, position, and velocity for a MANPADS detonation can support existing threat models used by program offices and the warfighter.

Simulated tests can significantly reduce costs and decrease analysis time when compared to physical arena tests. Physical testing can then confirm simulation results and examine scenarios of high uncertainty. Simulated testing supports physical test planning, improving the quality of data gathered from live tests – an important contribution considering the resource constraints placed on physical testing.

1.3 Scope

The scenario under study involves a MANPADS detonated within a fragment capture arena used to measure fragment masses and velocities. Arena design constraints cause only a portion of the weapon's fragments to be captured. Data extrapolation is required to construct a complete distribution. This research develops methods to perform simulated arena tests and to enhance data quality from physical tests by suggesting arena configuration improvements. A realistic simulation model for MANPADS testing facilitates the exploration of all problem components discussed in Section 1.2. Developing a suitable model and the supporting data analysis toolkit are the overarching goals of this research effort.

While this study focuses on MANPADS test analysis, the methodology is extensible to fragment capture tests for other types of small-warhead munitions. Further study would be required to assess if these findings are applicable to large explosives, such as the Mark 80 series general purpose bombs. Explosives testing also typically involves measuring blast pressures. Although an important test component, blast characterization is outside the scope of this problem.

1.4 Research questions

Several research questions are developed and answered in Chapter 4. The first explores the notion of the data quality captured by an arena configuration. A data quality metric is formulated and applied to several make-screen configurations. The second research

question focuses on comparing different arena configurations. The data quality metric from the first question is used in making comparisons, along with a cost metric developed for this research question. The final research question covers the improvement of arena configurations based predicted fragment distributions. A manual technique is developed and demonstrated, and a programmatic technique is discussed.

1.5 Overview

Chapter 2 provides a review of relevant literature and necessary background for the problem at hand. Chapter 3 presents the development of the baseline MANPADS simulation model. Chapter 4 advances several research questions related to fragment capture arena test design and addresses them using the simulation model. Chapter 5 concludes the thesis by offering key research findings, recommendations, and proposed areas for future research.

II. Background

2.1 Introduction

The purpose of this chapter is to survey the relevant literature on the problem and to develop the context needed for the remainder of the thesis, with an emphasis on weapons test methodology.

2.2 Literature review

This literature review covers mathematical modeling of explosives and test methodology for explosive weaponry, two areas pertinent to the problem at hand.

2.2.1 Explosives modeling.

As early as World War II, scientists were working to model the behavior of explosives. English physicist Nevill F. Mott investigated the fragmentation of exploding shell casings during the war and is responsible for developing the Mott distribution, "the hallmark relation for the representation of exploding munitions fragmentation data" (Grady, 2004). Mott's equations are used to predict the mass distribution of fragments from a cylindrical bomb and are defined as

$$N(m) = \frac{M_0}{2M_K^2} \exp\left(-\frac{\sqrt{m}}{M_K}\right) \tag{2.1}$$

and

$$M_K = Bt^{5/16} d^{1/3} \left(1 + \frac{t}{d}\right) \tag{2.2}$$

where $N(m)$ gives the total number of fragments with mass greater than m; M_0 is the total cylinder mass; M_K is a distribution factor; B is a constant related to the explosive and metal used in the bomb; t is the cylinder wall thickness; and d is the inner diameter (Cooper, 1996). Therefore, with precise information about the weapon construction, a mass distribution can be generated.

A simpler expression of Mott's equations is

$$N(m) = N_0 \exp\left(-\sqrt{\frac{m}{\mu}}\right) \tag{2.3}$$

where $N(m)$ and M_0 are as before, μ is one-half the mean fragment mass, and $N_0 = M_0/\mu$ (Gold, 2007). This form lacks the predictive ability of Equation 2.1 since μ depends on measurements of the fragments. However, it is useful for validating sample data and for implementing in computer code, such as the MOTT code presented by Vladimir Gold (Gold, 2007).

In the same decade as Mott's seminal work, R. W. Gurney derived a model for predicting the initial velocity of fragments produced by an exploding bomb (Cooper, 1996, Chapter 27). The Gurney equation for a cylindrical shell filled with a cylindrical explosive charge is

$$\frac{V}{\sqrt{2E}} = \left(\frac{M}{C} + \frac{1}{2}\right)^{-1/2} \tag{2.4}$$

where V is the initial outward velocity of the metal (before breakup), M is the mass of the shell, C is the mass of the charge, and $\sqrt{2E}$ is the Gurney velocity coefficient, a constant associated with each particular explosive. This equation is based on the transfer of energy from the charge to the metal shell. As the shell fragments, energy is lost and confined detonation gases escape more easily, halting further fragment acceleration. Thus, initial fragment velocities are less than V; Cooper cites 80% of V as a typical value. The Gurney equation takes on slightly different forms for different explosive shapes and configurations. However, the cylindrical configuration is most relevant here since it matches the shape of MANPADS missiles.

Modelers need to know the degree of deceleration fragments experience as they travel from the weapon to the screen, since make-screens in fragment capture arenas are some distance away from the weapon itself. Gravity and drag are the two forces acting on fragments after the explosive charge has been consumed. Atmospheric drag on a fragment

is expressed as

$$F = \frac{1}{2} C_D A_f \rho_a V_f^2 \tag{2.5}$$

where F is the drag force, C_D is the drag coefficient, A_f is the fragment's face area, ρ_a is the air density, and V_f is the fragment velocity. Note that fragment velocity is time-varying in this equation. Basic physics relates mass, acceleration, and force by

$$F = ma. \tag{2.6}$$

Cooper uses Equations 2.5 and 2.6, the definitions of acceleration and velocity, and integration to calculate the velocity lost by a fragment of mass m due to drag during a flight of distance d:

$$\ln\left(\frac{V}{V_0}\right) = -\frac{C_D A_f \rho_a d}{2m}. \tag{2.7}$$

Incorporating gravity into the model produces nonlinear equations, but gravity will have a minor effect on fast fragments traveling short distances (such as those found in a test arena), therefore it is reasonable to ignore the effect of gravity in this research.

2.2.2 Explosives testing.

The Joint Aircraft Survivability Program Office (JASPO) and the 96th Test Group at Wright-Patterson Air Force Base (WPAFB) are responsible for much of the published information on MANPADS testing in the realm of aircraft survivability. Their work includes the design of tactics, techniques, and procedures (TTPs) for protecting against MANPADS attacks near airfields (Czarnecki et al., 2003); assessment of large aircraft control surface and engine vulnerability (Czarnecki et al., 2008, 2011b); and use of MANPADS miss distances to evaluate low-altitude aircraft survivability (Bestard and Czarnecki, 2009).

A series of tests performed by the group and reported on in 2011 yielded the highest-fidelity data to date for both missile blast analysis and fragment capture, responding to specific requests by the modeling community for better data (Czarnecki et al., 2011a).

Their fragment capture procedures are the motivation for the research questions in Chapter 4. Tests were conducted with the missile stationary (static) or in motion at constant velocity (dynamic); each test was instrumented for either blast analysis or fragment capture. In total, the group was responsible for eight blast tests (six static, two dynamic) and four fragment capture tests (two static, two dynamic). In the blast tests, pressure gauges were placed as close as twelve inches from the warhead and "maximized in number," an improvement over previous tests that used few sensors placed relatively far from the weapon. The team constructed a fragment capture arena for each of the fragment characterization tests (refer to Section 2.3 for arena details). In these tests, the team collected data on all captured fragments weighing more than four grains (approximately 0.26 grams). They focused their efforts in this phase of testing on mapping fragment masses to velocities as measured with make-screens. The group's results can be used to update threat characterization models supporting live-fire Test & Evaluation (T&E) in aircraft acquisition programs.

2.3 Arena construction and operation

In the static configuration, the weapon is centered in an arena consisting of fragment capture bundles, make-screens for recording fragment velocities, and other data collection equipment. Figure 2.1 depicts this kind of arena configuration; the rectangular make-screens cover the walls in the background, while the missile in the foreground is attached to a platform to elevate it from the floor (Czarnecki et al., 2011a). Alternatively, the weapon can be launched into the arena at a controlled velocity and set to detonate at the exact center of the arena, in order to capture dynamic test data. The shape of the arena is determined by the testers but is typically a square or half-square. Capture bundles are thick layers of fibrous material used for arresting weapon fragments. After the test, researchers record the position where each fragment entered the capture bundle and remove layers from the bundle until the fragment itself is located and its mass recorded (Czarnecki,

2013). Collecting fragments in this manner is a very slow process, causing significant delay between conducting the test and analyzing the results.

A make-screen is a device used to record the timing of a weapon fragment impact. In its simplest form, it consists of an insulating material sandwiched between two charged, conductive surfaces (e.g. a thin piece of cardboard placed between two sheets of aluminum foil) that are attached to a data channel. These can be made fit any dimension as required by the test; Czarnecki et al. (2011a) report using screens one to four square-feet in surface area. When a fragment penetrates the make-screen, it creates an electrical connection between the front and back surfaces. An attached data channel registers the connection and records the time of impact. By recording the time of detonation, the time-of-flight for the fragment can be calculated. The fragment velocity is

$$\text{velocity} = \frac{\text{distance traveled}}{\text{time of impact} - \text{time of detonation}}. \tag{2.8}$$

However, this velocity is an estimate, as there is currently no way to know where on the missile body each fragment originated. For simplicity, all travel distances are measured from the weapon centroid.

There are several drawbacks with this design. First, the make-screens will not register an impact from a non-conductive fragment. Second, when a make-screen is struck more than once, multiple impact times are recorded, and there is no way to determine which impact time is associated with a particular fragment. To mitigate this, screen sizes can be selected to keep the expected number of fragments per screen low, and the average velocity of all fragments incident on a screen can be used as a measurement. Third, a make-screen can be rendered inoperable if a fragment becomes lodged in the make-screen while completing the circuit. If that occurs, any subsequent fragment impacts on that screen will not register. Fourth, make-screens require a small gap on all sides to avoid forming electrical connections with adjacent screens. Using smaller screens requires more screens

to cover the same arena walls, which causes more of the wall to be exposed by gaps, reducing the actual coverage area.

This arena design is the basis of the MANPADS simulation model described in the next chapter.

Figure 2.1: A MANPAD missile centered in a fragment capture arena (Czarnecki et al., 2011a). Rectangular patterns on the walls show the edges of make-screens.

III. Fragment Capture Model Design

3.1 Overview

This chapter discusses the fragment capture arena model developed for MANPADS simulation and the data structures used to map physical reality to a suitable software model. The model incorporates representations of fragment trajectories, the arena configuration, and fragment impact data. It produces output data during each simulation run for later analysis. All parts of the simulation are written in MATLAB using object-oriented programming principles.

3.2 Baseline model components

The key elements of the model are implemented as class objects. The primary classes used in the model are:

`Arena` Container for all aspects of the simulation configuration. The Arena object specifies a test weapon, a collection of make-screens, and methods for constructing the arena from external input. Methods for transforming data between Cartesian and spherical coordinate systems are also defined here.

`Screen` Representation of an individual make-screen, containing the screen's position within the arena and a list of fragments that impact the screen after the simulated weapon is detonated.

`Weapon` Model for the MANPADS missile being tested. This class specifies the fragmentary mass of the weapon, the distributions from which fragment masses and velocities are sampled, methods for generating fragments at the time of detonation, and a list of the fragments themselves.

`Frag` Model for an individual weapon fragment that specifies its weight, the impact position, and its impact velocity.

These objects have a hierarchical organization within the software (Figure 3.1); objects lower in the hierarchy are contained in some way by those higher in the hierarchy. For example, an `Arena` contains many `Screen` instances but only one `Weapon`, and a `Weapon` contains many `Frag` instances.

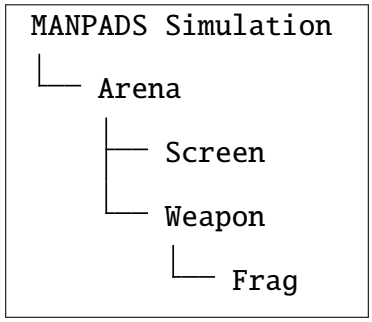


Figure 3.1: Object hierarchy within MANPADS simulation.

Positions within the arena are principally specified in Cartesian coordinates using the arena coordinate system. The weapon centroid is the origin of this system; the y-axis follows the long axis of the weapon (nose to tail); the x-axis is perpendicular to the y-axis and parallel to the arena floor; and the z-axis is the arena's vertical axis (Figures 3.2 and 3.3). Note that the y- and z-axes are not affected by the missile's roll angle within the arena, although the roll angle may affect the data collected from a test.

Regarding arena construction, rectangular or square arena walls are easiest to construct and use in physical tests. However, the exploding missile casts fragments in all directions, including above and below the weapon where make-screens are not typically located. This suggests that a spherical arena and coordinate system are better suited for fragment capture analysis (Figures 3.4 and 3.5). Also, fragment measurements on the walls of a rectangular arena are subject to a particular kind of distortion that can be removed via a spherical projection. For example, in a rectangular arena, two fragments that have a certain angular

separation will be closer to one another, in straight-line distance, when impacting the center of a wall than when striking near a corner. This occurs simply because the weapon's distance to the corners is greater than its distance to the wall centers. In a spherical arena, all points of the sphere's surface are equidistant from the origin, which eliminates distortion (see Figure 3.7 for an example of this effect). Finally, fragment trajectory distributions are better described by the angles with respect to the weapon than by positions within the arena. A spherical coordinate system lends itself well to this purpose.

During the simulation, positions in Cartesian coordinates (x, y, z) are converted to spherical coordinates (r, θ, ϕ), where radius r is the straight-line distance from the origin, θ is the azimuth angle measured counter-clockwise from the positive x-axis in the x-y plane (i.e. from the right side of the missile) as in standard polar coordinates, and ϕ is the elevation angle measured from the x-y plane (Figure 3.6). These conventions are chosen to conform with those used in MATLAB's own functions. For convenience, the unit sphere is used for all spherical projections within this model. The MATLAB functions `cart2sph` and `sph2cart` are used for converting between Cartesian and spherical coordinates, internally using the following equations:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{3.1}$$

$$\theta = \arctan\left(\frac{y}{x}\right) \tag{3.2}$$

$$\phi = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \tag{3.3}$$

and

$$x = r\cos\phi\cos\theta \tag{3.4}$$

$$y = r\cos\phi\sin\theta \tag{3.5}$$

$$z = r\sin\phi\,. \tag{3.6}$$

The inverse tangent function used by MATLAB (`arctan2`) includes quadrant checks, so the result is in the interval $[-\pi, \pi]$ instead of the usual $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ (MATLAB, 2012). Since the simulation works with unit spheres, r is typically set equal to unity instead of being calculated.

Spherical coordinates are used internally by the model for programming convenience. First, mapping the arena to the unit sphere keeps r constant. Second, constant movement about the perimeter of the arena can be measured by constant changes to θ; movement about the perimeter requires nonlinear changes in x and y. Third, the corners of the arena where two walls meet require no special handling after projecting to the unit sphere – a corner simply becomes the border between make-screens. In Cartesian coordinates, the two walls need to be processed differently on either side of the corner. Finally, fragment trajectory distributions are more naturally developed in spherical coordinates due to the spherical behavior of an explosion.

Customizing distributions is the only time the analyst needs to work in spherical coordinates. All other inputs and outputs are specified in Cartesian coordinates with the weapon at the origin, the positive y-axis at the missile's nose, and the positive z-axis pointed straight up – the arena coordinate system. The model is capable of producing graphical output in the spherical system if desired by the user, but the standard operating modes display data in arena coordinates.

3.3 Fragment modeling

Weapon fragments in the MANPADS model have three key attributes: mass, position on the arena walls, and velocity at time of impact. These are set as properties within each `Frag` object.

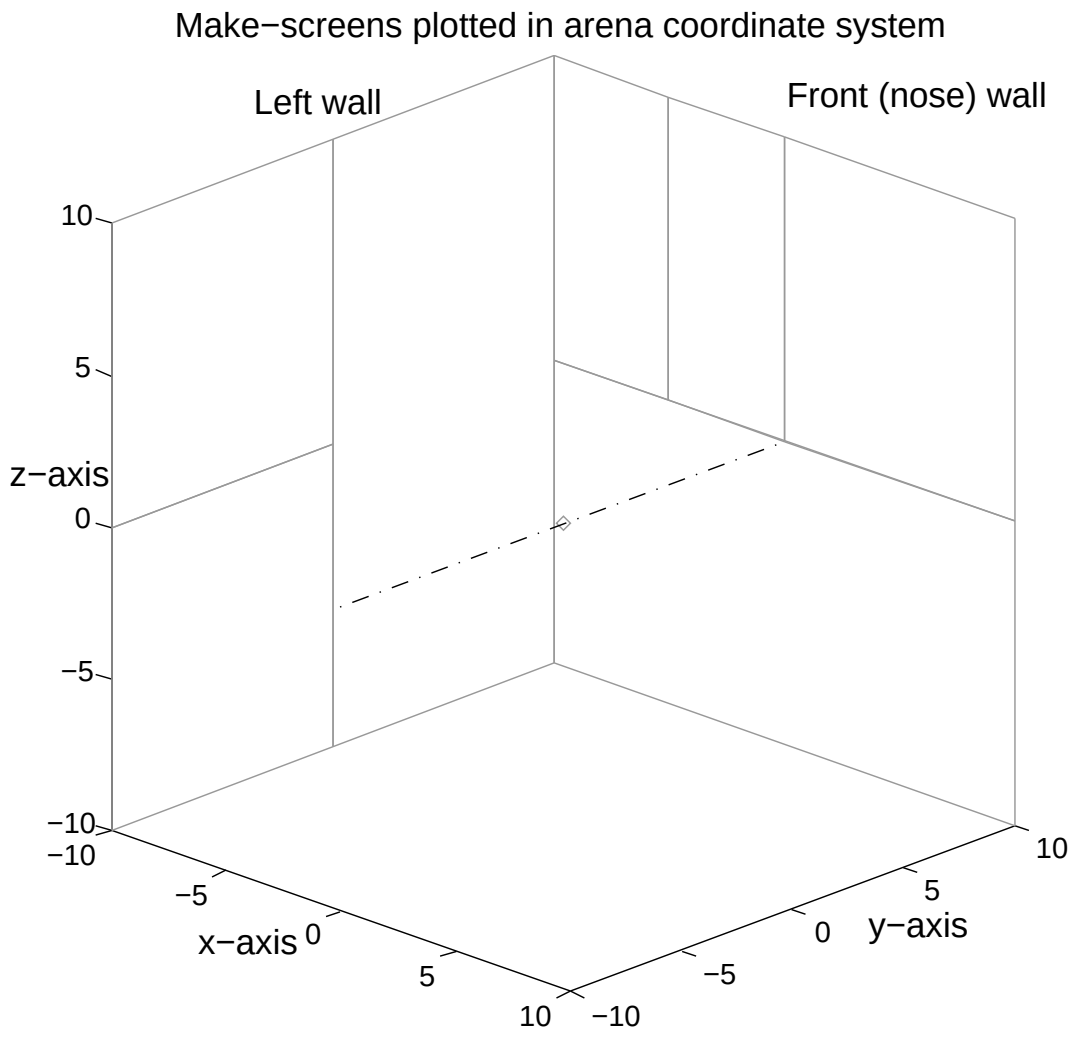


Figure 3.2: A simple two-walled arena with three make-screens placed on the left wall and four screens on the front wall. The weapon is placed in the arena with its long axis on the y-axis (denoted with the dashed line) and the nose pointed toward the front wall.

Mass is assigned to fragments by solving the simpler form of Mott's equation for mass m. Thus Equation 2.3 is rewritten as

$$m = \mu \left(\ln \left[\frac{N}{M_0 \mu^{-1}} \right] \right)^2 . \tag{3.7}$$

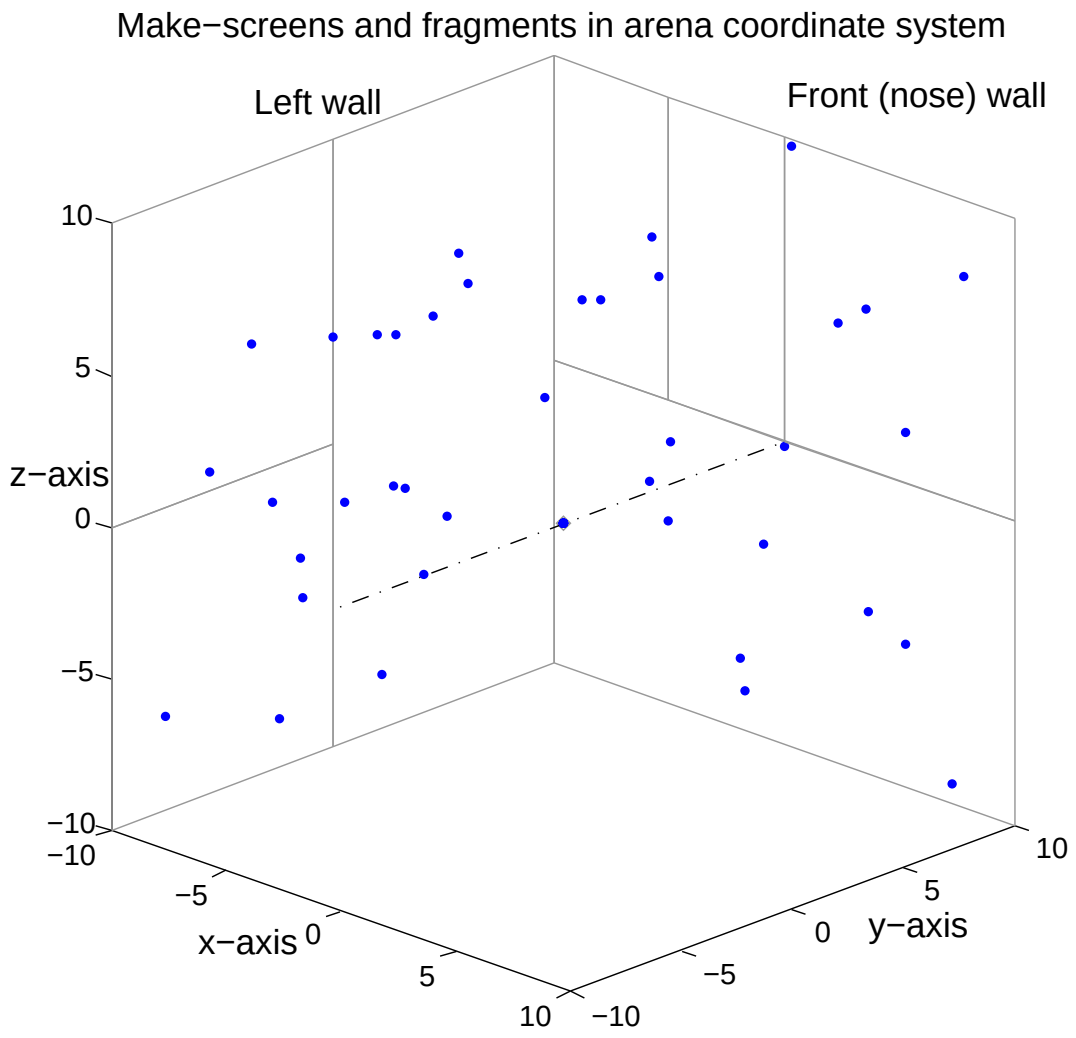


Figure 3.3: Sample arena with fragments. The same arena shown in Figure 3.2 is populated with fragment impacts.

The `Weapon` object defines the properties `numFrags` for the number of fragments to be produced by the weapon and `fragMass` as the total mass of those fragments. N, the number of fragments of mass m or greater, is treated as a uniformly-distributed random number for producing random variates and is bounded to the range $[0, \texttt{numFrags}]$. Note

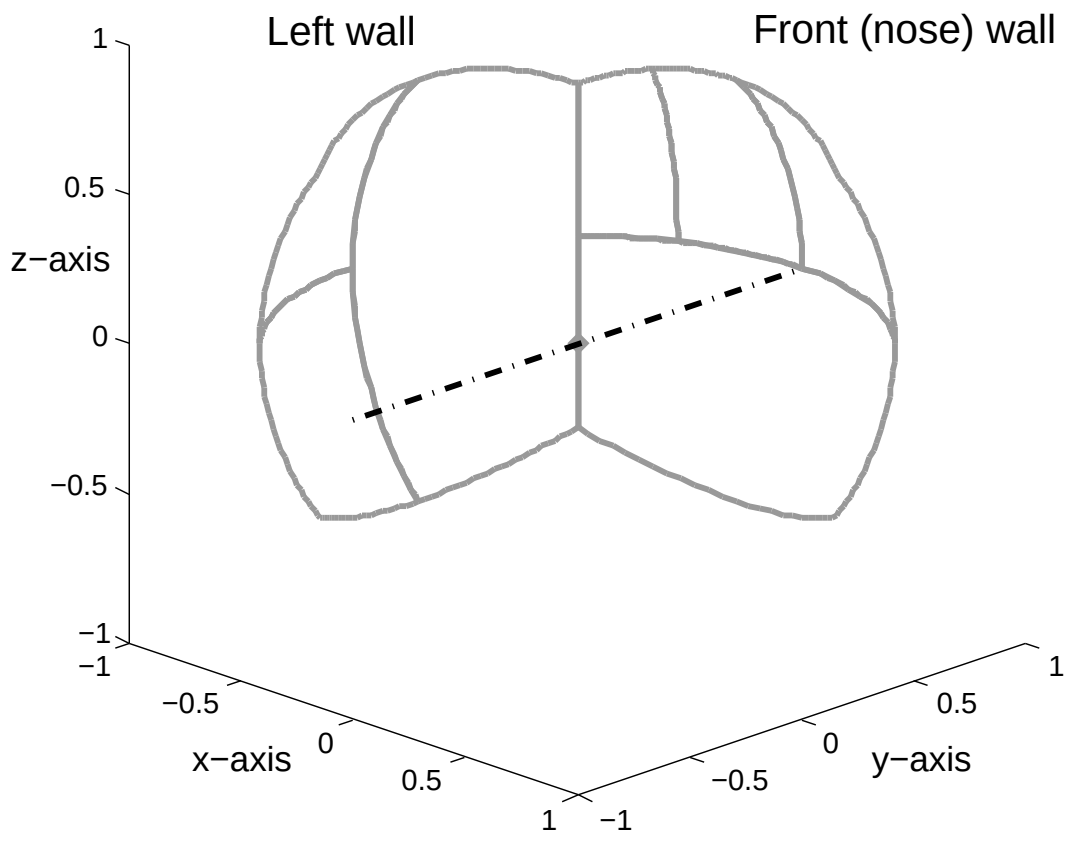


Figure 3.4: Sample arena projected onto unit sphere. The arena shown in Figure 3.2 is projected onto a unit sphere and viewed from the same camera angle.

that $M_0 = \texttt{fragMass}$, so one-half of the mean fragment mass, μ, is

$$\mu = \frac{1}{2}\frac{\texttt{fragMass}}{\texttt{numFrags}} \tag{3.8}$$

and

$$M_0\mu^{-1} = \frac{\texttt{fragMass}}{\frac{1}{2}\frac{\texttt{fragMass}}{\texttt{numFrags}}} = 2 \cdot \texttt{numFrags}. \tag{3.9}$$

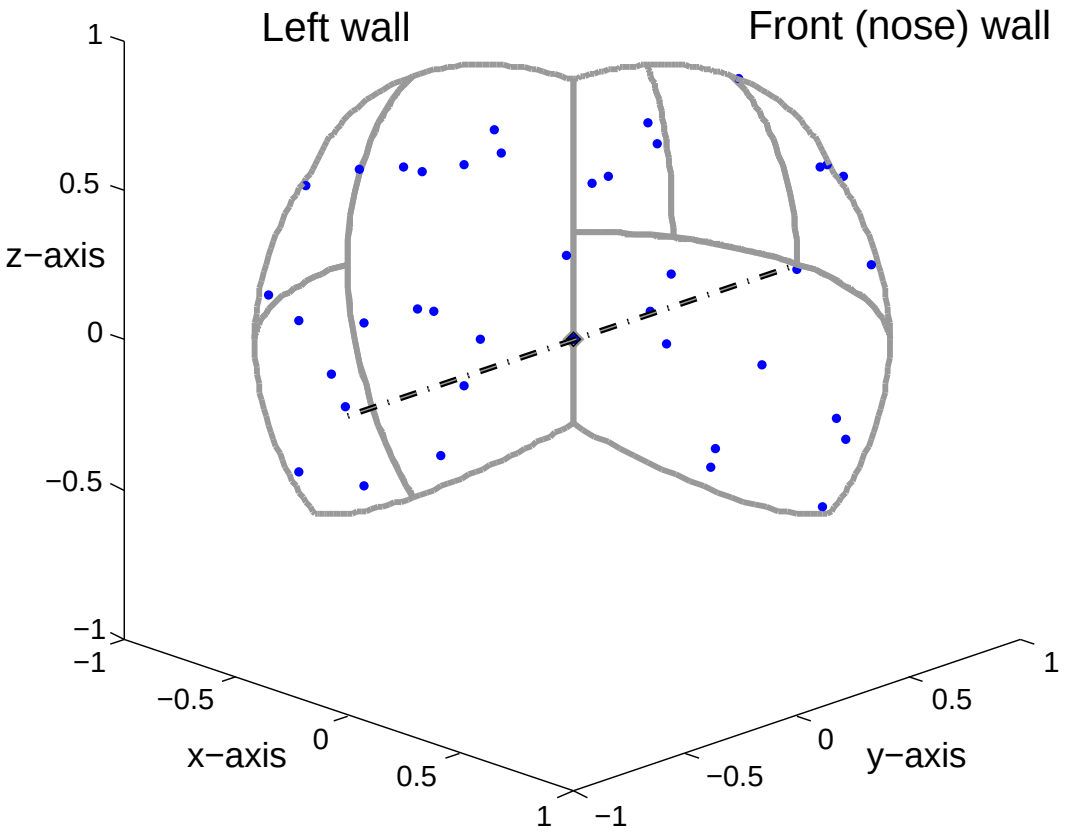


Figure 3.5: Sample arena with fragments projected onto unit sphere. The same arena configuration shown in Figures 3.2 and 3.3 is projected onto the unit sphere and viewed from the same camera angle. Positions of fragments appear distorted by this projection, but angular separation is preserved.

Therefore, Equation 3.7 becomes

$$m = \frac{1}{2}\frac{\texttt{fragMass}}{\texttt{numFrags}}\left(\ln\left[\frac{N}{2\cdot\texttt{numFrags}}\right]\right)^2. \tag{3.10}$$

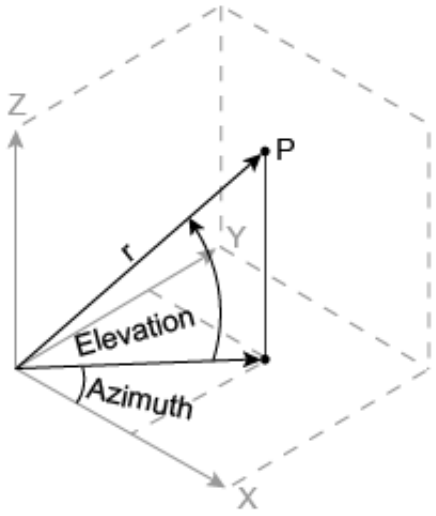


Figure 3.6: Spherical coordinate system used in model. Azimuth is θ and elevation is ϕ (MathWorks, 2013). Note that the definitions of θ and ϕ are often reversed in physics applications.

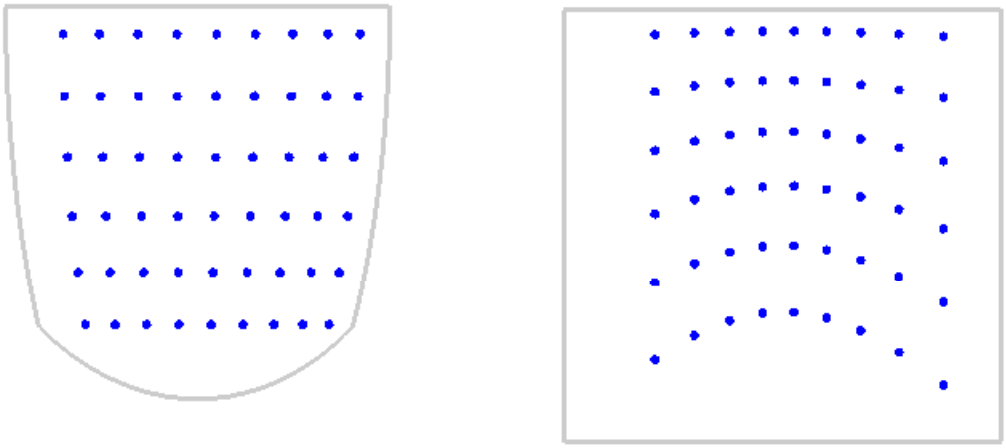


Figure 3.7: Example of distortion caused by coordinate conversion. (Left) A make-screen with fragments projected onto a unit sphere. The rows have equal ϕ separation and the columns have equal θ separation. (Right) The same data projected onto a flat surface ten meters from the origin. Heavy distortion occurs.

As each `Frag` object is generated, a uniform random draw for $N \in [0, \texttt{numFrags}]$ is used with Equation 3.10 to assign the fragment mass m. If the user has fragment mass data from physical testing, an empirical distribution could be used here instead.

Fragment trajectories are sampled from user-specified distributions. Current data collection techniques in physical MANPADS testing does not allow for a fragment to be traced back to its point of origin from the weapon. Thus, all fragments in this model originate from the weapon's center of mass (the arena origin). Trajectories are designated using spherical coordinates and stored in the `Frag` properties `theta` and `phi`. The properties `x`, `y`, and `z` are used to record the fragment's impact position on a make-screen, calculated later in the simulation.

The Gurney equation (Equation 2.4) is used to calculate the initial fragment velocity V, a common value for all fragments produced by a particular weapon. The output from the Gurney equation is reduced by twenty percent due to the average initial energy loss suggested by Cooper (Cooper, 1996, Chapter 27). This reduced value is stored in the `Weapon` object as `initialVelocity`. (In this model, velocity refers to the magnitude of the velocity vector, not the vector itself.) Part of the weapon specification includes the shell mass, the charge mass, and an explosive constant. The following parameter values are used in the model:

- shell mass $M = 15$ kg, based on estimating a 20 kilogram weapon;
- charge mass $C = 5$ kg, based on same estimate as shell mass; and
- explosive constant $\sqrt{2E} = 2.67$ km/s, from the mean of $\sqrt{2E}$ values cited by Cooper (1996) in Table 27.1.

Aerodynamic drag likely has a strong effect on fragment velocities in physical tests. However, using notional values with Equation 2.7 yields very small velocity changes. Also, the drag equation requires the face area of the weapon fragment. MANPADS missiles

are highly non-homogeneous in composition, so no good theoretical distribution exists for sampling fragment size and shape. Fragments may change their face area by tumbling while in flight. The combination of these factors leads to the exclusion of the drag force from the model.

Without drag, all fragments in the model would have the same final velocity, since all fragments are assigned the same initial velocity. Brief study of physical test data shows that fragments have a wide range of impact velocities. This variation is in part due to drag and the fact that all fragments do not originate from the weapon centroid. To account for this, a velocity reduction factor is imposed on each fragment. The user provides a value for the model (`maxReduxFactor`) that designates the maximum percent reduction allowed; `maxReduxFactor` must be in the closed interval [0, 1]. Impact velocity is calculated as

$$V_f = V_0 \left(1 - U\left(0, \texttt{maxReduxFactor}\right)\right) \tag{3.11}$$

where $U(a, b)$ is a sample from the uniform distribution between a and b. Like with fragment mass, an empirical velocity distribution could be used in place of the reduction factor. Final velocity, in $^{m}/_{s}$, is stored as a property in the `Frag` object.

3.4 Arena modeling

A physical arena has a collection of make-screens; the `Arena` object has a corresponding set of `Screen` objects. Each make-screen in the arena is represented with a `Screen` class instance, which has properties for the screen's ID number, the positions of its upper-left and lower-right corners (in the arena coordinate system), and a list of fragments that impact the make-screen (populated after the `Weapon` object is detonated). Arena configuration data can either be provided by the user as an input into the model or created by the simulation as an output, explained in greater detail in Section 3.6.

3.5 Fragment impact calculations

Impact data for each fragment has three components:

- impact position in arena coordinates,
- identification of the make-screen struck by the fragment, and
- velocity at time of impact.

Impact velocity is determined at the time of fragment creation in this model and is not given further treatment in this section. In reality there is a dependence on travel distance due to drag forces. If the user chooses not to provide make-screen data, arena wall information is used instead. The algorithms in this section apply equally to both make-screens and walls.

A fragment strikes a make-screen at a point given by the intersection of the fragment's trajectory vector (defined by θ and ϕ) and the plane containing the make-screen. The plane of the make-screen is defined by any three points on the screen. The model uses the upper-left, upper-right, and lower-right corners of the screen, denoting them as $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2$, and $\vec{\mathbf{x}}_3$, respectively. A fragment intersects the unit sphere at the point $(1, \theta, \phi)$ in spherical coordinates, which is converted to Cartesian coordinates (x_s, y_s, z_s) using MATLAB's `sph2cart` function (the subscript s here designates the unit sphere). Let $\vec{\mathbf{x}}_4 = (x_4, y_4, z_4) = (0, 0, 0)$ and $\vec{\mathbf{x}}_5 = (x_s, y_s, z_s)$ define two points on the line following the fragment's trajectory. The intersection point $\vec{\mathbf{x}}_\cap = (x_\cap, y_\cap, z_\cap)$ of the line and the make-screen plane is found by solving the four simultaneous equations

$$0 = \begin{vmatrix} x_\cap & y_\cap & z_\cap & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} \tag{3.12}$$

$$x_\cap = x_4 + (x_5 - x_4)\, t \tag{3.13}$$

$$y_\cap = y_4 + (y_5 - y_4)\, t \tag{3.14}$$

$$z_\cap = z_4 + (z_5 - z_4)\, t \tag{3.15}$$

for $x_\cap$, $y_\cap$, and $z_\cap$ (Weisstein, 2013). This yields

$$t = -\frac{\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & x_5 - x_4 \\ y_1 & y_2 & y_3 & y_5 - y_4 \\ z_1 & z_2 & z_3 & z_5 - z_4 \end{vmatrix}}, \tag{3.16}$$

which can be substituted into the previous set of equations to solve for $\vec{\mathbf{x}}_\cap$. The intersection point is stored as a property of the fragment object. Since $\vec{\mathbf{x}}_4$ and $\vec{\mathbf{x}}_5$ are already defined, Equation 3.16 is simplified slightly to

$$t = -\frac{\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & 0 \\ y_1 & y_2 & y_3 & 0 \\ z_1 & z_2 & z_3 & 0 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & x_s \\ y_1 & y_2 & y_3 & y_s \\ z_1 & z_2 & z_3 & z_s \end{vmatrix}}. \tag{3.17}$$

and Equations 3.13 through 3.15 are simplified greatly to

$$x_\cap = x_s t \tag{3.18}$$

$$y_\cap = y_s t \tag{3.19}$$

$$z_\cap = z_s t. \tag{3.20}$$

The line defined by $\vec{\mathbf{x}}_4$ and $\vec{\mathbf{x}}_5$ extends in both directions, so the preceding calculations produce an intersection even if the fragment's trajectory vector points directly away from the make-screen. In fact, two anti-parallel trajectories have identical intersections for a given make-screen. To account for this, the dot product of the trajectory vector (x_s, y_s, z_s) and the make-screen intersection vector $(x_\cap, y_\cap, z_\cap)$ is calculated. If the dot product is zero, then the vectors are aligned in the same direction. Otherwise, the fragment trajectory points away from the screen and the intersection is ignored.

Equation 3.17 does not take into account the size of the make-screen, only the plane within which the make-screen lies. The impact point must lie within the boundaries of the make-screen for impact to occur. If it does, then the impact position and screen of the fragment are updated and the fragment is added to the screen's list of fragments. A series of simple inequalities is used to compare the impact position to the extreme points of the make-screen.

The impact calculation procedure is summarized in Figure 3.8. Once an impact is recorded for a fragment, the procedure skips to the next fragment on the list to avoid wasted computations – make-screens are non-overlapping, so at most one impact can occur. If no impact is found, the fragment is not assigned a final position.

3.5.1 Example of fragment impact calculation procedure.

To clarify the process of assessing fragment impacts detailed in Figure 3.8, consider a simple arena with two make-screens on the front (nose) wall. Table 3.1 defines the placement of the two screens. This arena contains a weapon that produces a single fragment whose trajectory intersects the unit sphere at $(r, \theta, \phi) = (1, 1, 0.5)$; this point is generated by the `Weapon` object's fragment trajectory distribution function. This is converted to Cartesian coordinates to give $\vec{\mathbf{x}}_5 = (0.47, 0.74, 0.48)$.

Next, the line-plane intersection of the fragment's trajectory and the plane of the first make-screen is computed. For the first make-screen, $\vec{\mathbf{x}}_1 = (-1, 1, 1)$, $\vec{\mathbf{x}}_2 = (0, 1, 1)$, and

```
for each frag in fragments
    for each screen in screens
        point = line-plane intersection of frag and screen;

        if (point is within bounds of screen)
            and (point is in same direction as frag trajectory)

            frag.position = point;
            frag.screen = screen;
            screen.frags += frag;
            break to next frag;
```

Figure 3.8: Pseudo-code for calculating fragment impact points. Equation 3.17 generates an intersection point for each line-plane pair. That point must be checked against the boundaries of the make-screen. Also, the point must be in the same direction as the fragment's trajectory. If these conditions are met, then the fragment strikes the make-screen at the calculated point.

$\vec{\mathbf{x}}_3 = (0, 1, 0)$. These three vectors plus $\vec{\mathbf{x}}_5$ are used in Equation 3.17:

$$t = -\frac{\begin{vmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0.47 \\ 1 & 1 & 1 & 0.74 \\ 1 & 1 & 0 & 0.48 \end{vmatrix}} = 1.35\,. \tag{3.21}$$

The fragment's intersection point – with the plane, not neccessarily the make-screen – is found with Equations 3.18 - 3.20 to be $\vec{\mathbf{x}}_\cap = (0.64, 1, 0.65)$.

The boundary of the first make-screen is compared to $\vec{\mathbf{x}}_\cap$. Since the intersection's x-component of 0.64 lies outside the make-screen's x-bounds of $[-1, 0]$, the fragment does not impact the first make-screen. The procedure advances to the next make-screen.

Since the two make-screens are placed on the same wall, they are coplanar, and therefore $\vec{\mathbf{x}}_\cap$ is the same for the second make-screen. The x-bounds of this screen are $[0, 1]$, which contains 0.64. The y-bounds of $[1, 1]$ contain $y_\cap = 1$; the screen's z-bounds of $[0, 1]$ contain $z_\cap = 0.65$. Thus the intersection point $\vec{\mathbf{x}}_\cap$ lies within the boundary of the second make-screen.

Finally, the angle is found between vectors $\vec{\mathbf{x}}_5$ and $\vec{\mathbf{x}}_\cap$. If the angle is zero, the vectors are parallel and an impact occurs. (Near-zero angles are treated as zero in the software due to round-off errors. The only angles this procedure can produce are zero and π radians.) In this example, the angle between the vectors is zero. The screen number in the `Frag` object is set to two, and the fragment is added to the list of impacts maintained by the second `Screen` object.

Table 3.1: Specifications for make-screens used in impact procedure example. Positions are in the arena coordinate system.

ID	Upper-left			Lower-right		
	x	*y*	*z*	*x*	*y*	*z*
1	-1	1	1	0	1	0
2	0	1	1	1	1	0

3.6 Model input/output design

The baseline model described above has two modes of operation. The goal of Mode A is to scatter weapon fragments on defined walls of an arena, with no concern given to make-screens. The user provides the model with characteristics of the weapon under

test, positions of the arena walls (not make-screens), and the desired velocity reduction factor. Using these configuration values, the model generates fragment positions on the arena walls and the corresponding impact velocities. Also, the model provides graphical representations of the arena and fragments. This mode is useful for generating fragment distribution data for external analysis and is summarized in Figure 3.9.

Mode B is used for performing fragment impact analysis on a given make-screen configuration. The inputs required for this mode are the same as Mode A, with the addition of make-screen position information. Instead of scattering fragments on blank arena walls, the model places fragments on the make-screens and reports the velocity and position of each fragment impact with respect to the affected make-screen (Figure 3.10). Once all fragment impacts are computed, the make-screen velocities (the mean velocity of all fragments on a particular make-screen) are available. Graphical outputs of the arena include the make-screen borders. The second mode is well-suited for postmortem evaluation of physical test data.

Inputs		*Outputs*
- Weapon characteristics — Shell mass — Charge mass — Explosive constant — Mass distribution (if other than Mott's) — Number of fragments - Arena wall positions - Velocity reduction factor (or velocity distribution) - Trajectory distribution	$\Longrightarrow$	- Fragment impacts on arena walls — Velocity — Position - Post-detonation graphics of arena - Arena summary data

Figure 3.9: Operation Mode A input/output summary.

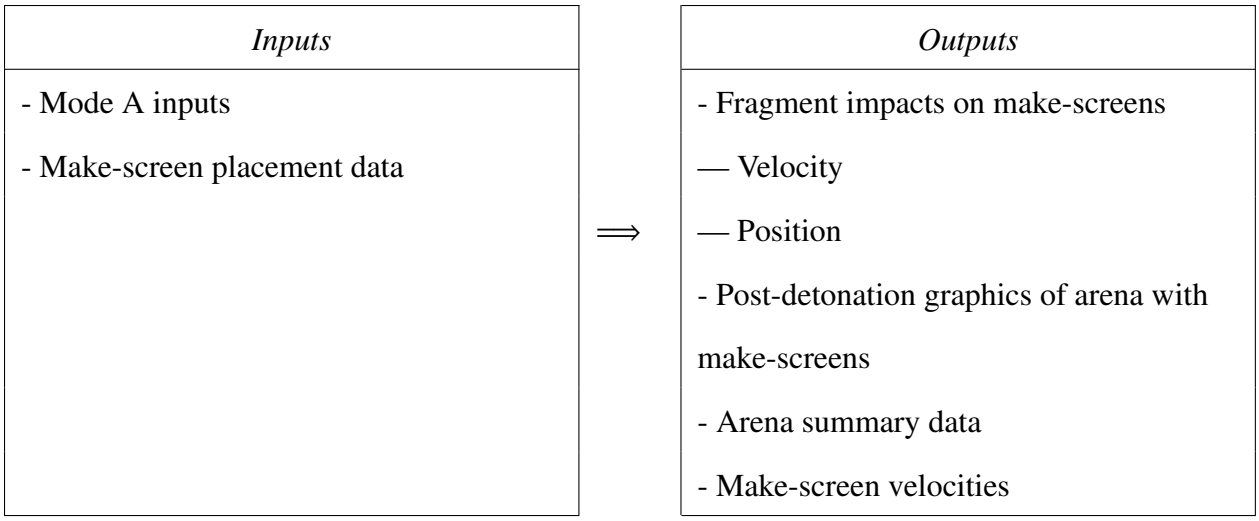


Figure 3.10: Operation Mode B input/output summary.

IV. Research Questions

The framework previously described provides a means to model the MANPADS arena. Model results can be evaluated empirically or quantitatively. Unfortunately, there is no useful quantitatively-based assessment method. The chapter develops a quantitative measure of arena configuration quality and then uses that measure to evaluate potential arena configurations.

4.1 How is data quality defined?

High-quality test data describes reality with a high degree of accuracy. Data from fragment capture tests includes fragment impact positions and make-screen velocity measurements. There is little error in the positions of fragment impacts, apart from human measurement error. However, there is great potential for error in measuring fragment velocities. The most notable source of error is in the way make-screens function – make-screens collect fragment impact times, compute times of flight, average those times, and use the screen's distance from the weapon to provide a mean fragment velocity. This velocity is then attributed to each fragment on the make-screen. The mean velocity for a large group of fragments is less representative of individual fragment behavior than the mean velocity for a smaller group. Velocity data quality is therefore maximized by using enough make-screens so that each screen captures a single fragment at most. This gives the most accurate estimate of fragment velocity possible but is completely impractical. Conversely, the lowest quality data is obtained by using one make-screen to cover an entire wall. Note that the actual measure of interest is the error between the make-screen velocity (estimated) and the corresponding fragment velocities (often unknown), not simply the number of make-screens.

Predicting the quality of data that can be produced by an arena configuration allows decision makers to choose between competing designs. The high resource costs of each physical test encourages careful arena design to maximize the utility of test data. The development of a data quality metric supports these goals and contributes to research questions addressed later in this chapter.

A measure of data quality should satisfy the following criteria:

1. encourages arena configurations with low mean numbers of fragments per make-screen, achievable by adding screens or repositioning screens based on expected fragment density;

2. encourages arena configurations with low error between a make-screen's measured velocity and the corresponding individual fragment velocities;

3. does not penalize configurations with make-screens that are not hit by any fragments – while wasteful of resources, they do not affect data quality;

4. does not reward configurations for adding make-screens without decreasing measurement error (e.g. using two screens to capture two same-speed fragments instead of one); and

5. allows comparison with other configurations.

Criteria 1 and 2 can be combined, since fewer fragments per make-screen naturally leads to more representative make-screen velocities, and more representative make-screen velocities correspond to reduced error. They are separated here for clarity.

Consider the arena configuration problem as follows. A configuration consists of $S = \{1, ..., n^s\}$ make-screens, each of which captures the group of fragments T^s, $s \in S$. Each screen captures $n^s = |T^s|$ fragments, and a total of $N = \sum_{s \in S} n^s$ fragments are recorded with $T = \bigcup_{s \in S} T^s$. We require that $T^i \bigcap T^j = \emptyset \, \forall \, i, j \in S, i \neq j$. Every fragment $f \in T$ has some

true velocity v_f and an estimated velocity $\hat{v}_f = v_s, f \in T^s, s \in S$ where v_s is the average velocity for all fragments impacting screen s. The mean absolute velocity error for screen s is

$$MAVE^s = \frac{1}{n^s} \sum_{f=1}^{n^s} \left| v_f - v_s \right|. \tag{4.1}$$

$MAVE$ achieves a maximum value when a screen captures only the fastest and slowest of all fragments captured by the arena[1]. $MAVE$ achieves a minimum value of zero when a screen captures a single fragment; this aligns $MAVE$ with criteria 1 and 2.

The actual magnitude of $MAVE$ depends on the number of fragments captured by screen s. It is normalized by dividing by the worst-case value $MAVE^*$; this allows comparisons among screens with different numbers of fragments, since the ratio is in the unit interval and $MAVE^*$ is the same for all screens in the arena. The per-screen quality score is thus defined as

$$Q^s = 1 - \frac{MAVE^s}{MAVE^*}. \tag{4.2}$$

The complement is taken so that better performance yields a higher score. It is possible for a make-screen to not be hit by fragments; such screens are deemed "inactive", while all

[1]For a make-screen with only the fastest and slowest captured fragments, with speeds v_{max} and v_{min},

$$MAVE^* = \frac{1}{2}\left(|v_{min} - v_*| + |v_{max} - v_*|\right) = \frac{1}{2}\left(v_* - v_{min} + v_{max} - v_*\right) = \frac{1}{2}\left(v_{max} - v_{min}\right).$$

If a fragment with velocity $v_f \in [v_{min}, v_{max}]$ is added to the screen, the average velocity becomes v_s. Note that $\left|v_f - v_s\right| \leq \frac{1}{2}\left(v_{max} - v_{min}\right)$ due to the calculation of mean v_s, so

$$\begin{aligned} MAVE^s &= \frac{1}{3}\left(v_{max} - v_{min} + \left|v_f - v_s\right|\right) \\ &\leq \frac{1}{3}\left(v_{max} - v_{min} + \frac{1}{2}\left(v_{max} - v_{min}\right)\right) \\ &\leq \frac{1}{2}\left(v_{max} - v_{min}\right) \\ &\leq MAVE^*. \end{aligned}$$

This process can be repeated for any number of fragments. Therefore $MAVE^*$ is the maximum (and worst-case) $MAVE$ value for a particular scenario.

screens with fragments are "active." Q^s is not computed for inactive screens, in support of criterion 3 above.

The quality scores for each active make-screen contribute to the overall arena score. The arena data quality score for arena configuration a is a measure of how well the arena configuration records the true velocities of the captured fragments. It is defined as

$$Q^a = \frac{1}{|S|} \sum_{s \in S_{active}} Q^s, \tag{4.3}$$

where $S_{active} \subset S$ includes only active make-screens. Dividing the sum by the number of screens weights each screen's contribution to Q^a according to the number of active screens. Adding screens to a configuration causes each screen to contribute less to the arena score (provided they are all active), so the additional screens must improve the actual data quality in order to improve the arena score. This aspect of Q^a supports criterion 4. Since $Q^s \in [0, 1]$, the maximum sum of screen scores is $|S|$; therefore Q^a is also in the unit interval. An arena data quality score of unity indicates all fragment velocities are measured correctly, and a score of zero means all fragment velocities are estimated with maximum error. Bounding the quality score enables natural comparisons between arenas (criterion 5).

Several demonstrations are now presented of the quality score in action. In the following runs, the model is configured with `maxReduxFactor` = 0.90. The same fragment pattern of fifty fragments is depicted in each of the graphics in this section unless noted otherwise. The arenas have a single wall; the number of make-screens vary.

Figure 4.1 shows arena configurations with one, four, sixteen, and sixty-four evenly-tiled make-screens. Qualitatively, the data quality improves with increasing screen count since the mean fragment count per screen decreases. Q^a is calculated for each layout and annotated below the plots. The calculated values support the qualitative relationship. In the sixteen and sixty-four screen arenas, several make-screens capture only one fragment.

These screens report the true velocity of the fragment (zero error) and make the greatest contribution to the overall arena scores.

To demonstrate that data quality is not a direct function of make-screen quantity, a new sixteen screen configuration is created such that each screen captures two to four fragments (Figure 4.2). Arena data quality increases about two percent. A final sixteen make-screen arena is constructed (Figure 4.3) to provide an example of an arena with the maximum data quality possible. All active screens capture exactly one fragment, so each make-screen velocity corresponds exactly to its fragment's velocity. This arena only has eleven captured weapon fragments, so five screens are inactive. The perfect data quality indicates that the inactive screens do not detract from the quality score. The scores from the experiments in this section are summarized in Table 4.1.

These results indicate that the quality metric provides a means to compare arena configurations for a given fragment pattern. These results are focused purely on solution quality – which configuration yields the most accurate fragment velocity estimates. The quality metric does not do anything with respect to arena cost or complexity considerations.

Table 4.1: Summary of data quality scores for Section 4.1.

Configuration	Q^a	# of screens hit	# of fragments
1 screen	0.4111	1	50
4 screens	0.4354	4	50
16 screens	0.5583	11	50
64 screens	0.7450	23	50
16 screens (custom)	0.5788	16	50
16 screens (perfect)	1	11	11

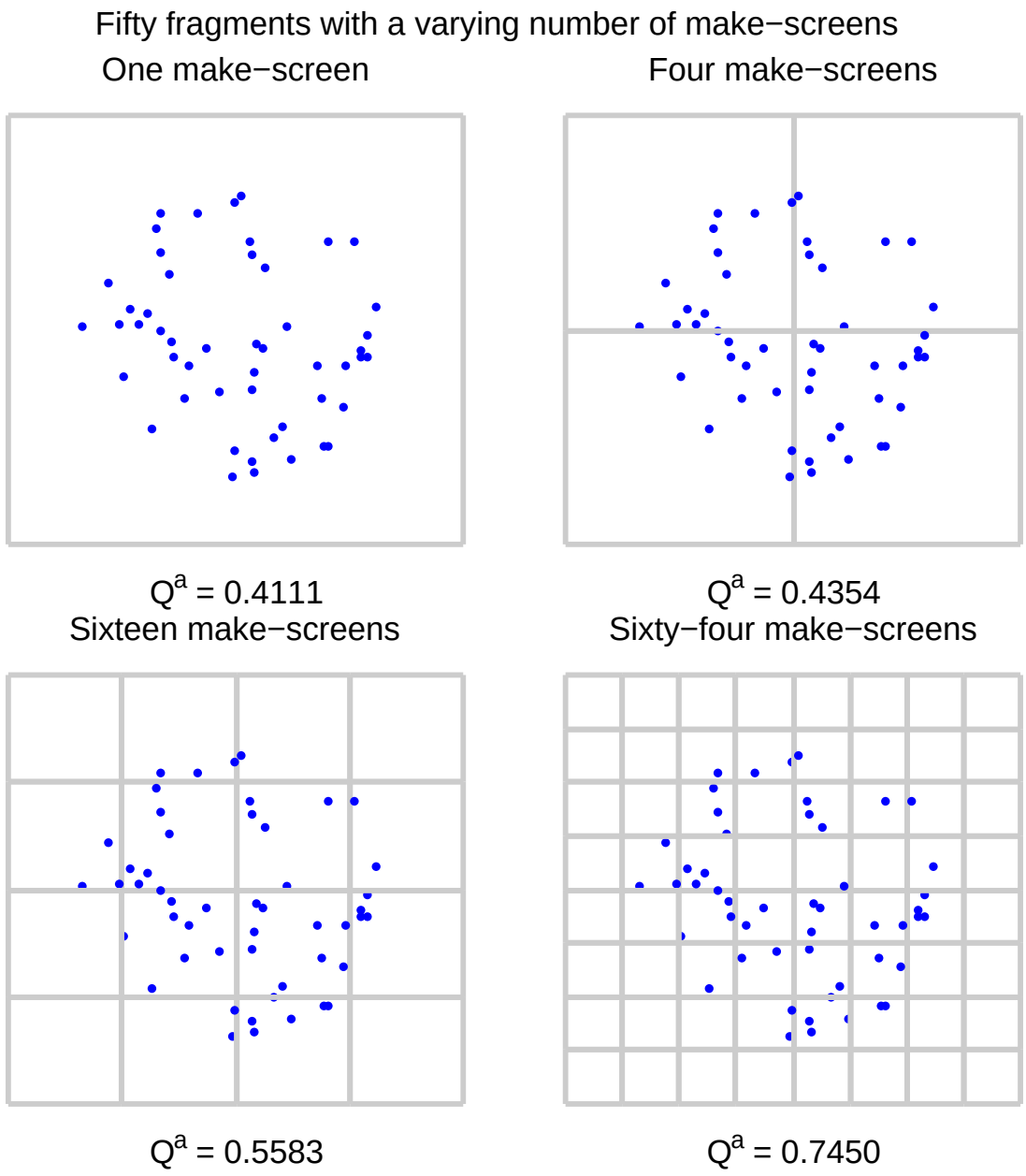


Figure 4.1: Fifty fragments on a wall with a varying number of make-screens. Square, evenly-tiled make-screens are used for convenience. Several screens for the sixteen and sixty-four make-screen configuration capture only one fragment, so they report the true velocity for that fragment.

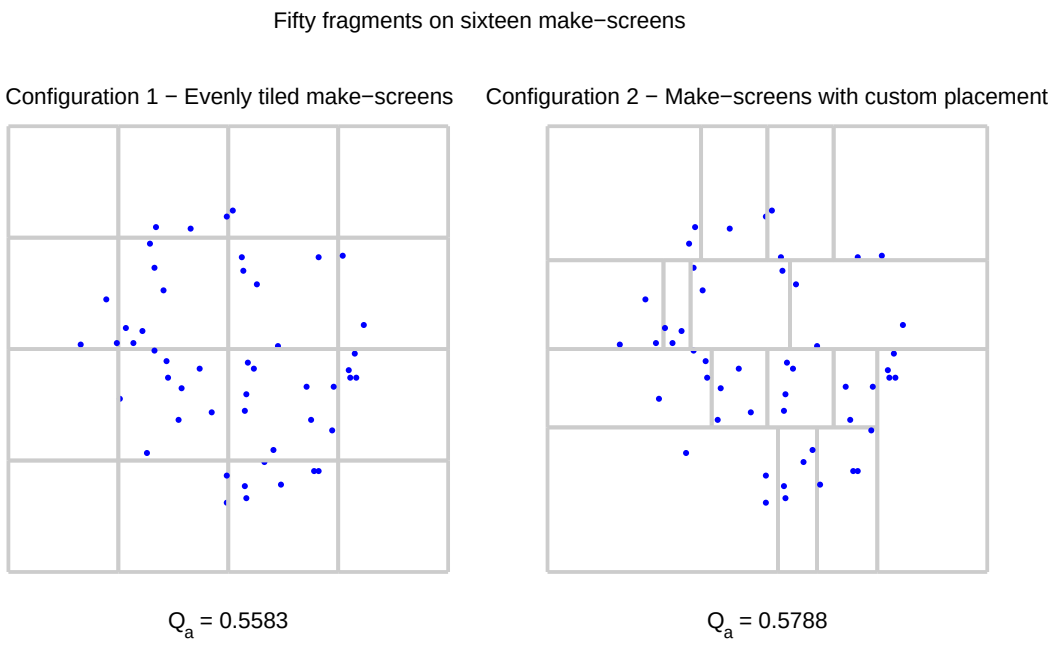


Figure 4.2: Fifty fragments with sixteen make-screens in two configurations. Configuration 1 has evenly tiled screens as seen in Figure 4.1. Configuration 2 has make-screens adjusted so each screen has between two and four fragments. The arena data quality scores show a two percent difference between the two configurations.

4.2 How can different arena configurations be compared?

The overall quality of an arena must be assessed with respect to both the quality of data it can capture and the cost/complexity of the physical arena. Complex arenas require more make-screens and a corresponding increase in instrumentation for those screens. The cost/complexity of a design must be traded off against the velocity estimate accuracy and quantity of data obtained. Therefore, arena cost must be quantified, to include monetary, resource, and time costs, plus measures of complexity related to the arena's design and construction.

Make-screens can be fabricated inexpensively with paperboard and conductive foil, so material costs are considered negligible compared to test elements such as the weapon itself. Each make-screen uses one data channel, regardless of screen size. Finally,

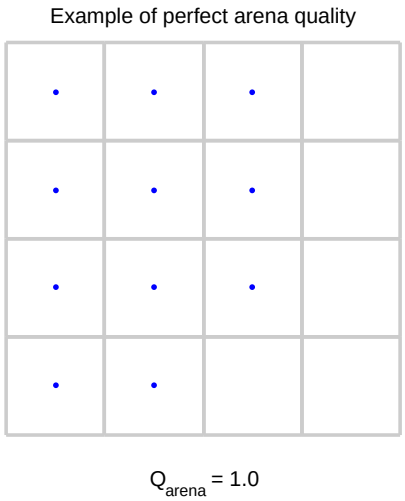


Figure 4.3: Example of configuration with $Q_{arena} = 1$. Fragments are synthetically placed so that each make-screen has no more than one fragment. A number of screens have no fragments, which demonstrates that inactive screens do not count against the quality score.

technicians spend time mounting each screen on the arena walls and making the physical connection to the data channel. Define the combined costs of constructing and installing a single make-screen as one cost unit. (Since material and labor costs will vary with each test, no attempt to convert cost units to dollars is made in this study.)

While make-screens of different sizes are treated as having the same cost, using a variety of screen sizes adds complexity to the arena configuration. Technicians must construct the proper number of screens for each dimension and ensure the screens are installed according to the configuration. These actions require time and expend logistical overhead. The cost relationships are nonlinear, however – using a single screen size has no additional cost; working with two sizes is only slightly more challenging; including ten sizes in the design adds significant work to the process. Let the complexity cost $c_{complex}$ be defined as

$$c_{complex} = \{\exp[t(S)] - \exp(1)\} \left(\binom{t(S)}{|S|} \right)^{1/4}, \tag{4.4}$$

where $t(S)$ is the number of screen sizes in S, and S is the set of all screens as before. The complexity cost is zero when $t(S) = 1$. The term $\left(\!\binom{t(S)}{|S|}\!\right)$ is the k-combination with repetition for $k = |S|$, which is expressed as

$$\left(\!\binom{t(S)}{|S|}\!\right) = \binom{t(S) + |S| - 1}{|S|} = \frac{(t(S) + |S| - 1)!}{|S|!\,(t(S) - 1)!}\,. \tag{4.5}$$

The combination term scales $c_{complex}$ in proportion with the number of make-screens used. Rotations of a screen are treated as different screen types, since they need to be specified differently in a configuration document and require extra attention from technicians.

Each arena can capture a fraction of all fragments produced in a test based on the sizes and positions of the arena walls. Taller arena walls can capture more fragments but require more make-screens. Arena walls placed closer to the weapon will capture more fragments but are more likely to be critically damaged in the blast. The fragment capture fraction affects the quantity of available data, not the quality, and does not directly affect the arena cost. In general, test designers should determine the minimum acceptable capture fraction based on M&S needs and build the arena walls accordingly. The problem of optimizing arena wall placement is left for future research.

The per-screen costs and the configuration complexity cost contribute to the overall arena cost function

$$c^a = |S| + c_{complex}\,. \tag{4.6}$$

c^a has a lower bound of unity, occurring for the single make-screen configuration, and no upper bound. However, the number of available data channels `maxChannels` at the test facility creates an upper limit on the number of make-screens allowed in a configuration. The configuration is infeasible if $|S|$ exceeds the available data channel capacity, and c^a is maximized when $|S| =$ `maxChannels`.

Six sample make-screen configurations are used to investigate this cost function and are presented in Figure 4.4. Each configuration covers a square wall measuring eight units on a side and is constructed from the following pallet of make-screen sizes:

- 1 x 1,
- 1 x 2,
- 2 x 1,
- 2 x 2, and
- 4 x 2.

Each item above is screen height by screen width and measured in generic units.

Comparisons among arenas involve data quality (Equation 4.3) and arena cost (Equation 4.6). Calculating the former requires use of a fragment pattern. A "shotgun blast" fragment pattern similar to Figure 4.6, Distribution 2, is applied to the six arena configurations to allow calculation of Q^a for each. Summary values, arena cost, and data quality are calculated for all six configurations (Table 4.2). Since Configurations A - C have equal number and types of make-screens, their costs are equal; their data quality scores are similar for the fragment distribution used. The $t(S)$ value for a configuration appears to be the main cost driver in the make-screen configurations studied.

The cost and data quality values from Table 4.2 are presented in a scatter plot in Figure 4.5. Low arena costs and high data quality scores are desirable. Therefore, Configurations C, D, and E form a pareto-optimal boundary for this data; Configurations A and B are clearly dominated; F lies slightly below the line. Note that Configuration D likely does not provide sufficient data quality to justify using in a test.

4.3 How can we improve arena configurations based on expected fragment distribution data?

The previous section shows a method for comparing arenas when a particular fragment pattern is expected. However, no single make-screen configuration is likely best suited for every fragment impact distribution. What is needed is a configuration that works

Table 4.2: Summary of make-screen configurations in Figure 4.4 with their calculated costs and data quality scores for a uniform spherical fragment distribution.

Configuration	$\|S\|$	$t(S)$	c^a	Q^a
A	36	4	543.3	0.6965
B	36	4	543.3	0.5583
C	36	4	543.3	0.7450
D	8	1	8.0	0.4955
E	36	2	47.5	0.7263
F	48	3	150.7	0.7278

reasonably well across a range of likely distributions. Hill and McIntyre use the concept of "robust solutions" for selecting military force structures that work well against a variety of possible threat scenarios (Hill and McIntyre, 2000). For this study, a **robust arena configuration** is a make-screen layout with the best overall performance against a set of expected fragment impact patterns. Such a configuration has the highest mean data quality score Q^a for some set of possible fragment patterns, weighted by the probability of realizing each fragment pattern. Using this metric, arena configurations can be improved by sampling several configurations or by using a heuristic to generate improving solutions. The former is conducted below, while the latter is discussed later.

The six make-screen configuration used in the previous section (Figure 4.4) are treated with several fragment patterns (Figure 4.6). Distribution 1 is a simple uniform spherical distribution. Distribution 2 is a "shotgun blast" pattern, with a set number of fragments confined to a certain angular region. Distribution 3 starts as a uniform spherical distribution, but each point is shifted toward the horizontal plane, creating a band of high fragment density. Distribution 4 is similar to Distribution 2, but the fragments are largely confined to an annulus about the center of the wall. These four distributions are not representative of

expected results from live-fire testing but do provide enough diversity to explore the topic of robustness. For this analysis, each distribution has an equal probability of occurrence.

Data quality scores for each configuration-distribution pair are given in Table 4.3; the arena costs are the same as in the previous section. Of the four fragment distributions used, Configuration F has the highest Q^a for the first three distributions, while Configuration A is best for the last distribution. Overall, Configuration F has the highest average data quality, making it the robust configuration choice of the six considered. Configuration A is close in mean quality (0.7232 versus 0.7367) but has an arena cost that is more than triple that of Configuration F. It is interesting to note that Distribution 1, the uniform spherical fragment distribution, has the best mean Q^a across the six configurations. This is reasonable since none of the configurations were designed with the specific biases of the other distributions in mind, so they perform well for a uniform distribution. Also, each configuration sees its worst performance with Distribution 2, except for Configuration D which sees its best performance.

Building a variety of make-screen configurations and fragment patterns and evaluating each pair is one method of searching for a robust configuration. A more powerful method is to employ a heuristic to search the space of possible configurations to improve mean data quality. The solution space is sparse, especially if a small set of make-screen sizes is used. Some heuristics work with sparse spaces more effectively than others; this is an important consideration in heuristic selection and design (e.g. genetic algorithms tend to handle sparse spaces poorly). The forward problem of building a feasible configuration from a screen size pallet is difficult; the inverse problem of checking configuration feasibility is simple. If fixed screen sizes are not required, then the algorithm could start with a random configuration and adjust the boundaries of screens while measuring the response of the mean data quality score. This could give better solutions but would have much greater computational complexity, and the result could greatly increase the complexity cost

$c_{complex}$ of the arena due to the increased number of screen sizes. Exploring simulation-based optimization approaches to the robustness problem is a valuable endeavor but is set aside for future research efforts.

Table 4.3: Arena data quality scores for various pairings of arena configurations and fragment distributions. Make-screen configurations are shown in Figure 4.4. Fragment patterns are shown in Figure 4.6 and are assumed to be equally likely. The highest score for each distribution is in boldface. Configuration F has the highest mean score and is therefore the robust configuration.

		Fragment Distribution				
	Q^a	1	2	3	4	Mean
Arena configuration	A	0.7511	0.6788	0.6979	**0.7652**	0.7232
	B	0.7628	0.6745	0.6946	0.6879	0.7050
	C	0.7608	0.6923	0.7099	0.7110	0.7185
	D	0.5828	0.6006	0.4798	0.5511	0.5536
	E	0.7095	0.6757	0.6833	0.6930	0.6904
	F	**0.7825**	**0.7101**	**0.7249**	0.7292	**0.7367**
	Mean	0.7249	0.6720	0.6650	0.6896	

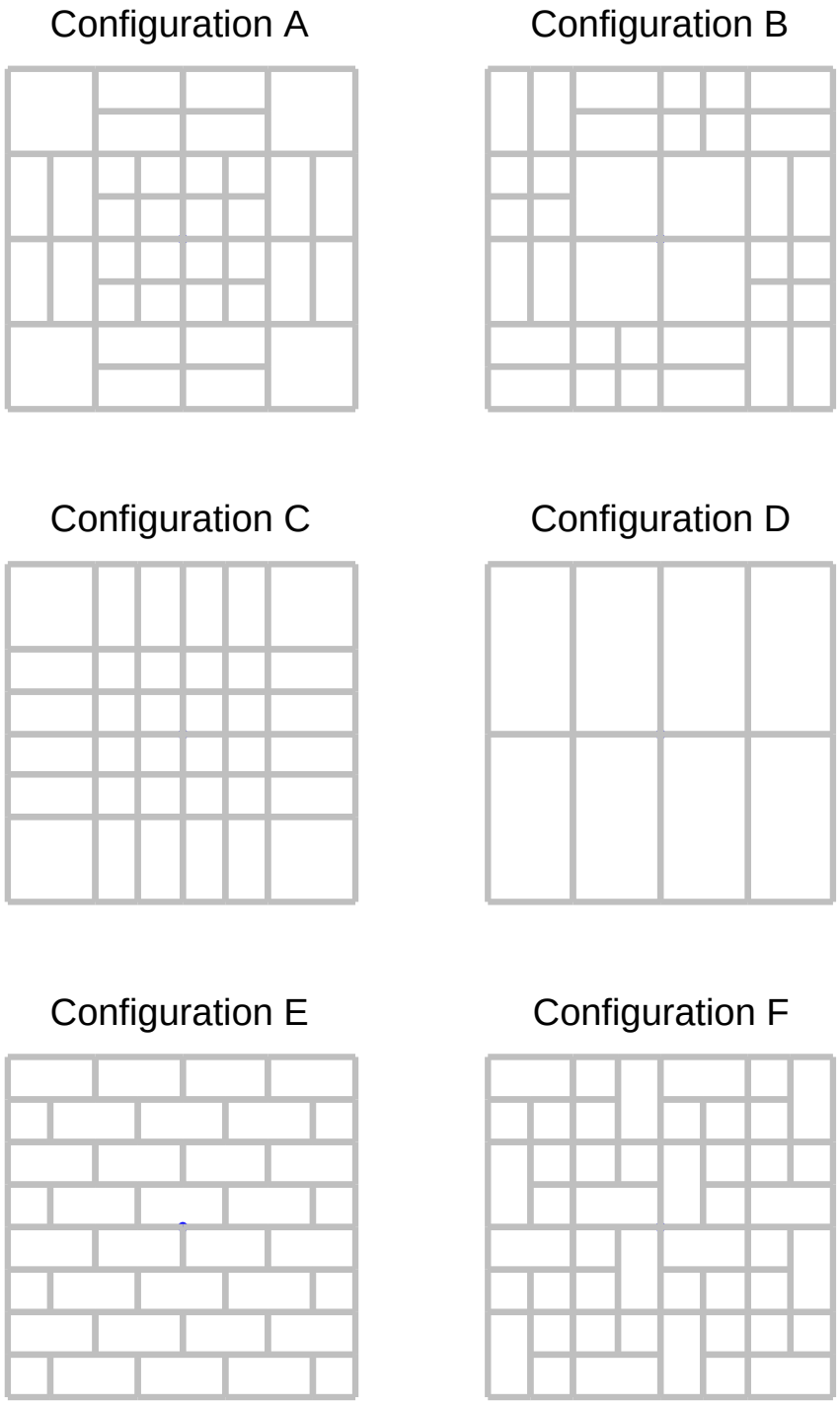


Figure 4.4: Six make-screen configurations for an 8 x 8 unit wall. Each configuration is arbitrarily created from a limited pallet of make-screen sizes. The smallest make-screen is a one-unit square.

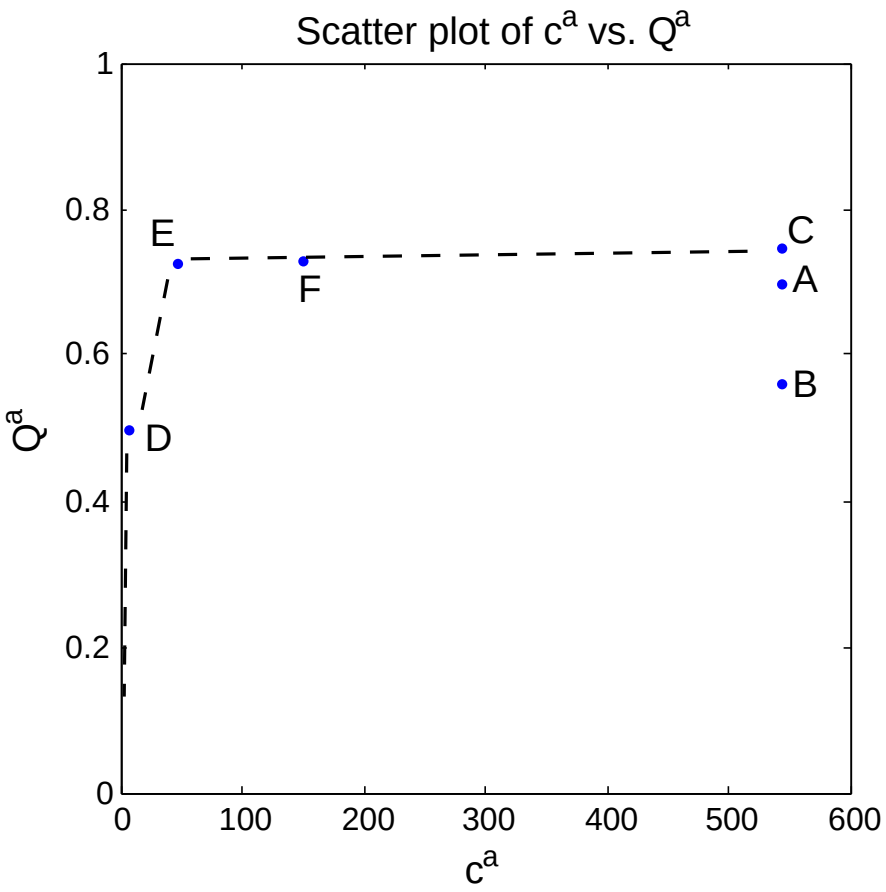


Figure 4.5: Scatter plot of c^a and Q^a data in Table 4.2. Low cost and high data quality are ideal, so Configurations C - D form a pareto-optimal boundary (dashed line), dominating Configurations A and B; F lies slightly below the line.

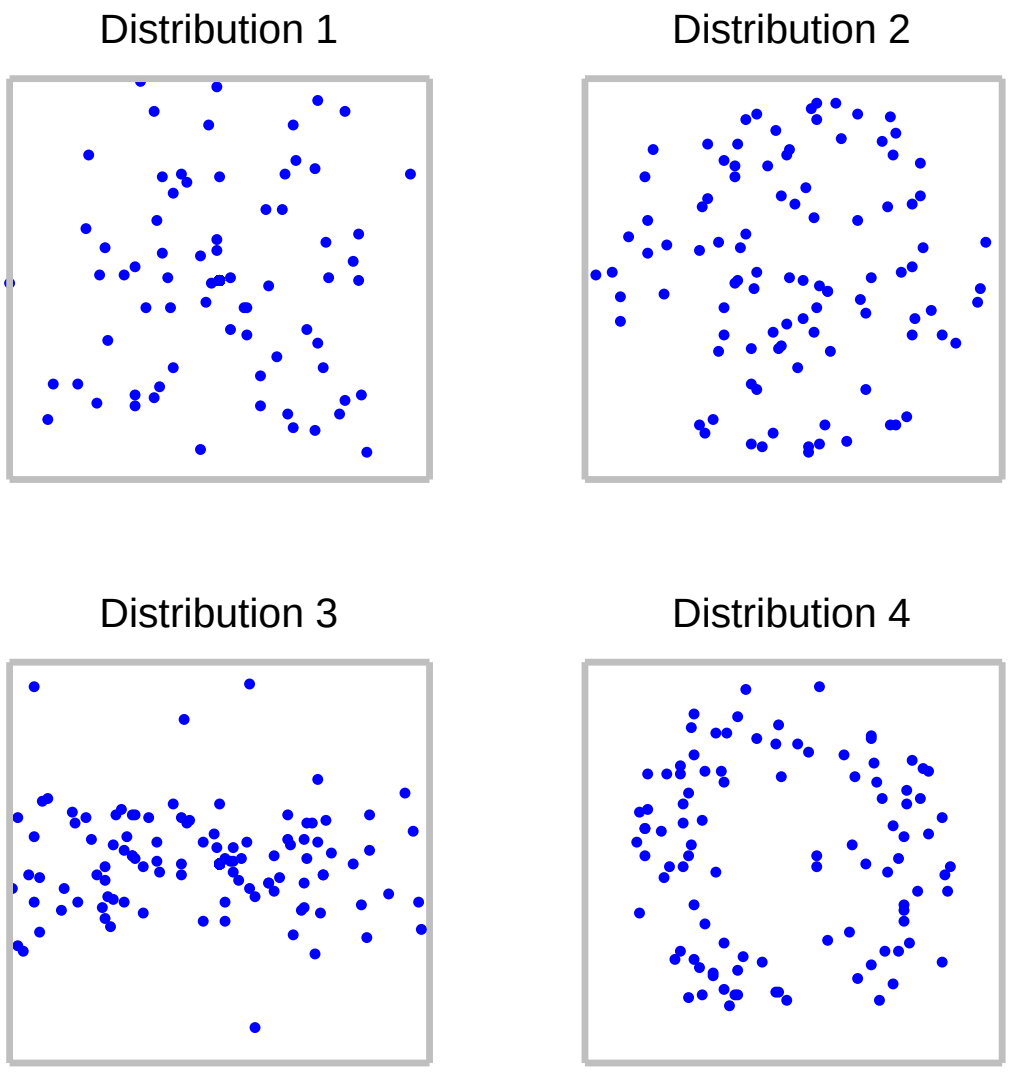


Figure 4.6: Fragment distribution patterns used for make-screen configuration analysis in Section 4.3.

V. Conclusion

This research examines the viability of a MANPADS arena simulation to assess the quality of potential arena configurations. The simulation is designed and a prototype developed. To assess arena configuration quality, a measure of configuration quality is developed. Arena configurations and representative MANPAD fragment patterns are used to demonstrate the utility of the simulation and the configuration quality metric.

5.1 Conclusions

This study explores the problem of fragment capture arena development for physical MANPADS testing. The nature of explosive munitions testing is reviewed. Past work in the field of aircraft survivability analysis and explosive modeling is surveyed, supporting the design of a simulation framework for arena testing. The simulation model encompasses velocity-measuring make-screen layout, weapon characteristics, and fragment capture analysis. A number of physics and engineering equations drive the simulation from initial setup through weapon detonation to the impact of fragments on the arena walls. Several research questions are explored: measuring potential data quality, comparing arena designs, and improving arena configurations using the data quality and arena cost metrics.

Data quality is defined using the error between true and estimated fragment velocities. Each make-screen has a quality score; the mean of all make-screen scores is the overall arena quality score. The data quality metric is shown to be influenced by the number of make-screens and the density of fragment impacts on make-screens.

Comparisons between arena configurations are made using the data quality metric and a cost function. The cost function captures the material, manpower, and logistical costs associated with a particular arena design. It includes a complexity term that is driven by the number of sizes of make-screens. More complex screen patterns are possible with a

larger selection of make-screen sizes, and more complex patterns carry greater logistical overhead.

The idea of robustness is used for selecting improved arena configurations. The robust choice from a set of make-screen layouts is one with the best mean data quality, given a set of expected fragment patterns. Candidate layouts can be made manually, as done in this work, or by using a heuristic, as discussed in the next section.

Certain aspects of MANPADS arena testing receive limited or no treatment in this study. Weapon blast characterization (heat and pressure) is not incorporated into this model. Blast tests are conducted separately from fragment capture tests, so omitting blast characterization in this model causes no loss of relevance. Fragment mass is included in the model but is not the focus of any analysis. Compared to velocity, fragment masses are easy to measure in practice with high precision. Also, M&S analysts put greater emphasis on gathering velocity data from tests (Czarnecki et al., 2011a), so velocity assessment is the primary focus. Arenas with multiple walls are not analyzed in this thesis to allow for better graphical presentation of results. However, the simulation includes support for any number of walls; the analysis methods may require some modification.

Physical test designers can potentially leverage this arena model to lower costs and/or increase achievable data quality. Multiple make-screen configurations can be evaluated in software before any are actually assembled. Designers can also make informed trade-offs between arena cost and data quality. Likewise, this model can be improved by comparing its predictions to physical test reality, especially with respect to improving the various distributions used by the model. Arena design, model development, and test execution form a feedback loop that improves the results of both physical and simulated arena tests (Figure 5.1).

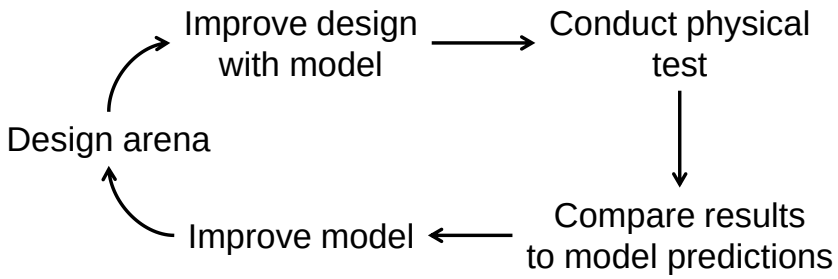


Figure 5.1: Feedback loop between physical arena tests and model development.

5.2 Recommendations for future work

The scenario under study provides a wealth of additional avenues of research that are not explored in this thesis. Several candidate topics are discussed below.

Although the framework supports arenas with any number of walls, single-walled arenas are studied in this document for the ease of displaying output. In practice, arenas have multiple walls that form a half or full enclosure of the weapon. The model can be used to assess the effectiveness of different arena sizes and number of walls, which is related to the problem of determining arena wall placement based on desired fragment capture fraction (mentioned in Section 4.2). Arena shapes other than square can be evaluated. For example, a square enclosure could be replaced with an octagonal arena, which has a smaller perimeter (and surface area) while still fully surrounding the weapon. Smaller surface area leads to fewer make-screens and lower arena cost, but the logistics of constructing and collecting data from a non-square arena may add expense not accounted for by the arena cost equation (Equation 4.6).

A potential cost reduction in arena configurations is the use of partial arenas. The analysis of such partial arenas requires strong assumptions regarding the symmetries of fragment patterns. Actual impact patterns may be used to assess whether such partial configurations would provide sufficient data fidelity.

Better fragment mass, velocity, and trajectory distributions can be generated via statistical analyses of physical test data. Computational engineering models may exist that can model fragment dynamics directly; output from such systems can be used to directly assign fragment characteristics in this model. More realistic fragment data would enhance the arena design recommendations from this model.

The limiting factor in velocity measurement is the error in make-screen velocity measurement. Large numbers of fragments on a screen diminishes the utility of the velocity estimates. If it is not practical to reconfigure make-screens in areas of extremely high expected fragment density, then make-screens can be excluded from such regions. Survivability experts may be able to predict that fragment densities above a certain threshold are certainly lethal to an airframe. In such cases, test resources may be better employed in different regions of the arena.

Finally, using a heuristic for selecting robust make-screen configurations is much more powerful than the manual approach detailed in Section 4.3. The main challenge in the robust arena configuration problem is the efficient generation of feasible make-screen configurations. The secondary challenge is producing the set of potential fragment impact distributions. With those two problems addressed, wrapping a heuristic around the simulation should be a straightforward task.

Appendix A: Input Specifications

The user typically customizes the arena configuration in four places: the simulation configuration file, the arena wall configuration file, the make-screen layout file, and the distribution functions in `Weapon.m`.

In the simulation configuration file, `Globals.m`, the user specifies the file paths for the wall and make-screen layout files, seeds the random number generator, sets the maximum velocity reduction factor, and configures the weapon (Figure A.1).

The arena wall configuration file is a comma-separated value (CSV) file where each line gives the upper-left corner, lower-right corner, and name of a wall ("Front," "Left," etc.). An example of this file is shown in Figure A.2. The full line specification is

`Ax`, `Ay`, `Az`, `Bx`, `By`, `Bz`, `name`

where `Ax` is the x-coordinate of the upper-left corner and `Bz` is the z-coordinate of the lower-right corner, both measured in the arena coordinate system.

The format for the make-screen layout file is nearly identical to the wall configuration file. Each line identifies a single make-screen using the format

`Ax`, `Ay`, `Az`, `Bx`, `By`, `Bz`, `wallNumber`

where `wallNumber` is the line number in the walls configuration file corresponding to the wall on which the make-screen is placed. Using Figure A.2 as an example, `wallNumber` = 1 for the front wall and `wallNumber` = 2 for the left wall. Figure A.3 show a make-screen layout file for a simple arena.

Trajectory distributions should accept a single argument `n`, the number of trajectories to return, which is typically the number of fragments produced by the `Weapon`. The return value should be the array `[Theta, Phi]`, where `Theta` and `Phi` are each vectors of length

`n` and give the travel direction of the fragment in spherical coordinates. Figure A.4 shows an example of a trajectory distribution function.

Mass distribution functions take two parameters: the total fragmentary mass of the weapon, and the total number of fragments. A single vector of fragment masses is returned. Mott's equation (Equation 3.10) is used for fragment masses. The code is designed around modularity, so the user can use any function with the proper signature; only the pointer in `Globals.m` needs to be changed.

User-defined distributions for weapon fragments are located in `Weapon.m`, but can actually be placed anywhere – the distribution function pointers in `Globals.m` are the only direct references to the functions themselves.

```
classdef Globals < handle
    % Container class for simulation configuration. The user
    % sets all custom values in here by changing the appropriate values.

    properties
        % == Simulation configuration ==

        % Configuration files; all paths relative to program root.
        wallsConfigFile   = 'walls.csv';
        screensConfigFile = 'screenConfigs/screens4.csv';

        rngSeed           = 12345; % Random number generator (RNG) seed

        % == Environment configuration ==

        % Reduction factor in [0, 1] for computing fragment final velocity.
        maxReduxFactor    = 0.90;

        % == Weapon configuration ==

        fragMass          = 10;          % kilograms
        chargeMass        = 4;           % kilograms
        expConst          = 0.00267;     % meters / sec
        numFrags          = 100;

        % Pointers to the distribution functions used by Weapon.
        % They must be preceeded by the @ symbol.
        trajDistro        = @Weapon.uniformSphericalDistro;
        massDistro        = @Weapon.mottMassDistro;
    end
end
```

Figure A.1: Example of the Globals.m file.

```
-10,10,10,10,10,-10,Front
-10,-10,10,-10,10,-10,Left
```

Figure A.2: Sample walls configuration file for a two-wall arena.

```
-10,10,10,0,10,0,1
-10,10,0,0,10,-10,1
0,10,10,10,10,0,1
0,10,0,10,10,-10,1
-10,-10,10,-10,10,-10,2
```

Figure A.3: Sample make-screen layout file, using the arena walls defined in Figure A.2. This file defines four make-screens on the front wall and a single screen on the left wall.

```
function [Theta, Phi] = uniformSphericalDistro(n)
    xyz = zeros(n, 3); % Initialize matrix
    for i = 1 : n
        v = randn(1, 3); x = v(1); y = v(2); z = v(3);
        xyz(i,:) = 1 / sqrt(x^2 + y^2 + z^2) * [x y z];  % Normalize
    end

    X = xyz(:,1); Y = xyz(:,2); Z = xyz(:,3);
    [Theta, Phi] = cart2sph(X, Y, Z);
end
```

Figure A.4: Example of trajectory distribution function used in model.

Appendix B: Software Overview

This appendix provides a summary of the simulation functions and samples from the code. A full, electronic version of the code is available upon request from AFIT/ENS.

The driver for the model (`myDriver.m`) carries out the following actions (Listing B.1):

1. configures the `Arena` object based on values set in `Globals.m` (as described in Appendix A),
2. detonates the `Weapon,`
3. generates graphical output, and
4. produces a text summary.

Calculation of the metrics used in Chapter 4 is not part of the normal program flow, but they can be found using the methods `Arena.getDataQuality()` and `Arena.getArenaCost()`.

Detonation is handled by `Arena.detonate()` and is listed in Figure B.2. The detonation process involves computing fragment trajectories and masses based on user-defined distributions, calculating fragment impact positions, and setting final fragment velocities.

Graphical output is composed of lines connecting the four corners of each make-screen and dots showing each fragment impact; the `Arena.plot()` method handles this. The user can choose to have the arena shown using a spherical projection by calling `Arena.projectToSphere()`. The arena is plotted in three dimensions for both of these methods, so the user can use the mouse or the `view()` command to change the camera angle. If the user wishes to focus on a single arena wall, they must adjust the camera angle as desired, then change the plot axes to cull the rest of the arena. This must be done manually.

The program can produce two text reports. The fragment report, created by `Arena.showFragReport()`, is a listing of the trajectories and impact information (if applicable) for each fragment produced by the weapon. A truncated example of this report is shown in Figure B.3. The method `Arena.showArenaSummary()` summarizes some of the model inputs and outputs from the simulation run (Figure B.4). Both reporting functions take a single boolean parameter. If true, then the report data is written to a file. Otherwise, it is displayed on-screen. Writing the data to a file would be useful for running a batch of simulations and analyzing the results later. If the user requires data not produced by these built-in reporting methods, they can customize the model or interrogate the `Arena` object directly.

```
% Driver code for running simulation

% -- Configure arena --
clear all;
global g; g = Globals(); % Need to use an instance

arena = Arena();

% Add walls to arena; a wall is a Screen object that
% is big enough to hold all its screens
arena.importWalls(g.wallsConfigFile);

% Add screens to arena; these should fit on the above walls
arena.importScreens(g.screensConfigFile);

% -- Detonate weapon --
arena.detonate();

% -- Produce graphical output --
arena.plot()
% arena.projectToSphere();

view(0,0);
rotate3d % Set Rotate 3d mode in plot viewer

% -- Produce text reports/summaries --
arena.showFragReport(false);
arena.showArenaSummary(false);
```

Figure B.1: Listing for myDriver.m.

```
function detonate(self)
    % Simulate detonation of weapon by generating fragments,
    % assessing the impact screen for each fragment, and
    % calculating fragment impact points and velocities.

    % Set trajectories for frags
    [fragTheta, fragPhi] = self.weapon.trajectoryDistro(self.weapon.numFrags);

    for i = 1:self.weapon.numFrags
        f = Frag();
        f.spherical = Geometry.SphericalCoordinate(fragTheta(i), fragPhi(i), 1);
        f.projPos = Geometry.Vector(f.spherical);

        self.weapon.frags(i) = f;
    end

    % Set masses for fragments
    self.weapon.setFragMasses();

    % Determine which screen each frag strikes and where.
    self.setImpactsForFrags();

    % Compute frag final velocities
    self.weapon.setFragFinalVelocities();
end
```

Figure B.2: Detonation code within the `Arena.m` file.

ID	mass	Theta	Phi	Screen	X	Y	Z	V_impact
1	0.3875	-0.1457	-0.1611	**NaN**	0	0	0	0
2	1.1800	2.9792	0.3873	**NaN**	0	0	0	0
3	0.5333	0.5321	0.3920	**NaN**	0	0	0	0
4	0.0583	1.3489	0.3278	3	2.2557	10	3.4868	0.0011
5	0.0723	-2.3919	-0.5066	**NaN**	0	0	0	0
6	0.0546	1.2338	0.6297	3	3.5032	10	7.7217	0.0007

Figure B.3: Example fragment report. Screen values of "NaN" indicate that fragment did not hit a make-screen; its impact position and velocity is are therefore zero.

```
Arena Summary for Simulation Run Ending 140302-114116
=====================================================

Inputs:
- Fragmentary mass (kg) = 10.0000
- Charge mass (kg) = 4.0000
- Explosive constant = 0.0027
- Number of fragments = 50
- maxReduxFactor = 0.90
- Trajectory distribution = Weapon.uniformSphericalDistro
- Mass distribution = Weapon.mottMassDistro
- RNG seed = 12345

Outputs:
- Captured fragments = 13 (26.00 %)
- Elapsed time is 1.16 seconds.
```

Figure B.4: Example arena summary report.

Fragment Capture Simulation for MANPADS Test Arena Optimization

1st Lt Michael J. Garee Department of Operational Sciences (ENS), Air Force Institute of Technology

Objective

To develop a model, analysis toolkit, and simulation framework for MANPADS fragment capture arena testing.

MANPADS Overview

Man-portable air defense systems (MANPADS) are shoulder-fired, guided anti-aircraft missiles. An estimated 500K - 750K remain stockpiled worldwide.

The U.S. State Department views curbing the spread of MANPADS as a top priority for national security. In 2005, the RAND Corporation concluded:

> "... one anti-aircraft missile purchased for as little as a few thousand dollars on the black market could kill hundreds of people and cause economic damage exceeding $16 billion."

Fragment Capture Arenas

These tests collect fragment masses, impact positions, and impact speeds. Walls of catch bundles partially surround the weapon. These bundles capture fragments for later extraction. However, the partial enclosure will not capture all of a weapon's fragments.

The walls are covered in make-screens that estimate fragment impact velocities. Make-screens are typically hit with multiple fragments and are unable to match velocity estimates to particular impacts. So, the average velocity is assigned to each of a screen's fragments, resulting in measurement error.

Contact Information

Adviser: Dr. Raymond R. Hill

Reader: Dr. Daryl Ahner

Sponsors: OSD DOT&E and 96th TG

Model Development

Model Components (MATLAB)

`Arena` Container for a test weapon, walls, make-screens, and methods for constructing the arena.

`Screen` An individual make-screen model.

`Weapon` Model for MANPADS missile being tested.

`Frag` An individual weapon fragment.

Fragment Modeling

- Mott's equation

$$m = \mu \left(\ln \left[\frac{N}{M_0 \mu^{-1}} \right] \right)^2$$

- Gurney's equation

$$\frac{V_0}{\sqrt{2E}} = \left(\frac{M}{C} + \frac{1}{2} \right)^{-1/2}$$

- Velocity reduction factor

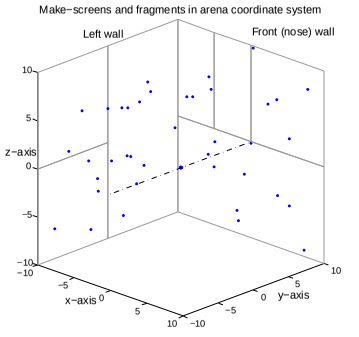


Figure 4: Model output of simple arena with make-screens and fragments.

Research Question 1

"How is data quality defined?"

- Mean Absolute Velocity Error

$$MAVE^s = \frac{1}{n^s} \sum_{f=1}^{n^s} |v_f - v_s|$$

- Arena Quality Score

$$Q^a = \frac{1}{|S|} \sum_{s \in S_{active}} \left(1 - \frac{MAVE^s}{MAVE^*} \right)$$

($MAVE^*$ is worst-case $MAVE^s$ for a scenario.)

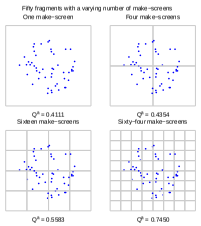


Figure 1: One fragment pattern applied to four screen layouts. Screens with few fragment contribute most to Q^a.

Research Question 2

"How can arena configurations be compared?"

Two factors: quality Q^a and arena cost c^a

Cost includes:

- Money
- Resources
- Time/labor
- Design complexity

Complexity cost $c_{complex}$ is a function of total screen count $|S|$ and number of screen sizes.

$$c^a = |S| + c_{complex}$$

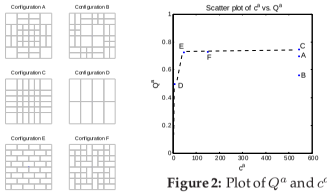


Figure 2: Plot of Q^a and c^a.

Research Question 3

"How can configurations be improved using fragment predictions?"

Robust arena configuration: a make-screen layout with highest mean Q^a for a set of expected fragment patterns.

Manual search: create sets of screen layouts & fragment patterns by hand, then find Q^a for each pair. The layout with highest mean quality is the **robust choice**.

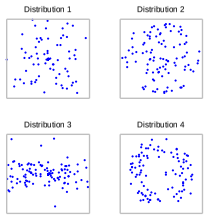
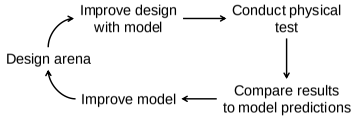


Figure 3: Four fragment patterns used in study.

Conclusions

- Developed simulation framework
- Assessed model utility via research questions
- Recommend test designers use model in future development efforts



Future Research

The best topic for future work is using a heuristic to search for robust arena configurations. Key challenges are the efficient generation of make-screen layouts and encoding the search space in a way that works with heuristics.

Bibliography

Bestard, Jamie and Greg Czarnecki. Survivability in the low altitude regime – MANPADS miss distance assessment. *Aircraft Survivability*, Summer:6–7, 2009.

Bureau of Political-Military Affairs. MANPADS: Combating the threat to global aviation from Man-Portable Air Defense Systems, July 2011.

Calapodas, Nicholas J. MANPADS ballistic test of a helicopter composite generic tailboom. *Aircraft Survivability*, Winter:27–29, 2001.

Cooper, Paul W. *Explosives Engineering*. Wiley-VCH, 1996.

Czarnecki, Greg. Private communications. Information from meetings, 2013.

Czarnecki, Greg, Kristina Langer, and Jeff Wuich. Improving aircraft survivability to shoulder-fired missiles. *Aircraft Survivability*, Spring:32–33, 2003.

Czarnecki, Greg, Gautam Shah, and John Haas. Control surface vulnerablilty to MANPADS. *Aircraft Survivability*, Summer:21–22, 2008.

Czarnecki, Greg, Stan Loughmiller, John Valek, Tim Grose, and John Haas. MANPADS threat model development. *Aircraft Survivability*, Fall:6–8, 2011a.

Czarnecki, Greg, Joe Manchor, Gautam Shah, and John Haas. Large engine vulnerablilty to MANPADS. *Aircraft Survivability*, Fall:9–11, 2011b.

Gold, Vladimir M. Engineering model for design of explosive fragmentation munitions. Technical report, U.S. Army Armament Research, Development, and Engineering Center, 2007.

Grady, Dennis. The Statistical Fragmentation Theory of N. F. Mott. In *American Institute of Physics Conference Series*, volume 706 of *American Institute of Physics Conference Series*, pages 455–460, July 2004. doi: 10.1063/1.1780276.

Hill, Raymond and Greg McIntyre. A methodology for robust, multi-scenario optimizaton. *Phalanx*, pages 27–31, September 2000.

MathWorks. Transform cartesian coordinates to spherical, September 2013. URL http://www.mathworks.com/help/matlab/ref/cart2sph.html.

MATLAB. *version 8.0.0.783 (R2012b)*. The MathWorks Inc., 2012.

Schmitt, Alexandra. MANPADS at a glance, March 2013. URL http://www.armscontrol.org/factsheets/manpads.

Schroeder, Matt. Countering the MANPADS threat: Strategies for success, September 2007. URL http://www.armscontrol.org/act/2007_09/CoverStory.

SURVIAC. Models distributed by SURVIAC. *SURVIAC Bulletin*, 1:13, 2013.

Under Secretary of Defense (AT&L). DoD Modeling and Simulation (M&S) Management (DoDD 5000.59), 2007.

Weisstein, Eric W. Line-plane intersection, September 2013. URL http://mathworld.wolfram.com/Line-PlaneIntersection.html.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2014	Master's Thesis	Oct 2012–Mar 2014

4. TITLE AND SUBTITLE

Fragment capture simulation for MANPADS test arena optimization

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Garee, Michael J., 1st Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-14-M-09

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of the Secretary of Defense
ATTN: Vincent Lillard
1700 Defense Pentagon
Washington D.C., 20301
(703) 697-7247
Vincent.Lillard@osd.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

OSD DOT&E

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The assessment of aircraft survivability against explosive munitions is an expensive undertaking. Test articles for both aircraft and weapon are scarce due to their high costs, leading to a limited supply of test data. The development of newer, hopefully more effective weaponry and protection measures continues despite the short supply of test data. Therefore, test organizations need to explore methods for increasing the quality of test results while looking for ways to decrease the associated costs. This research focuses on the Man-Portable Air-Defense System (MANPADS) as the weapon of choice and live-fire arena testing as the experimental data source. A simulation infrastructure is built and used to examine how to optimize the arena configuration to maximize the test information obtained. Several research questions are explored: measuring potential data quality, comparing arena designs, and improving arena configurations based on fragment pattern predictions.

15. SUBJECT TERMS

MANPADS, survivability, projectile fragments, blast simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Raymond R. Hill (ENS)
U	U	U	UU	74	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 ext. 7469