INFO FLASH
Finabel
The European Land Force Commanders Organisation

FINABEL - THE EUROPEAN LAND FORCE COMMANDERS ORGANISATION

AUGUST 2025

WRITTEN BY
PIETER DE NIJS

EDITED BY
JONAS HEINS

SUPERVISED BY
STEPHEN CROWLEY

Introduction

The proliferation of advanced precision-guided munitions (PGMs) has constituted a significant operational risk for fixed military installations and soft targets, particularly airbases and grounded aircraft. In contemporary peer-level high-intensity conflict, the ability of an adversary to conduct large-scale, pre-emptive missile strikes against these static assets presents a critical challenge to allied force projection and airpower generation (Priebe et al., 2019, pp. viii-ix, 1-3). While active defence systems are critical to the defensive matrix, their potential to be overwhelmed or exhausted in a saturation attack (Wilkening, 2000, pp. 191, 203) necessitates a robust addition of passive defence measures, including hardening, dispersal, and deception to enhance airbase survivability and operational resilience (Karako, 2019, p. 3). This is particularly salient for European and NATO forces, whose forward posture on the Eastern flank concentrates high-value air assets at a limited number of main operating bases. This report aims to develop and employ a simulation-based framework to quantitatively assess the protective value of hardening techniques, specifically the use of Hardened Aircraft Shelters (HAS). By modelling various attack scenarios against different defensive postures of a representative NATO airbase, this study evaluates how different shelter configurations contribute to asset survivability. The analysis utilises probabilistic methods to account for the stochastic nature of missile combat, offering data-driven insights into the marginal utility of infrastructure hardening in mitigating the effects of a first strike.

1. The Doctrinal Foundations of Airpower and Its Inherent Vulnerabilities

Air superiority, or "the degree of dominance of one force over another which permits the conduct of operations by the former and its related land, sea, and air forces at a given time and place without prohibitive interference by the opposing force" (US DOD, 2015, p.10), has been a central objective in military strategy since the conceptualisation and integration of airpower during World War I. Over the past century, aerial supremacy has evolved from a novel concept into a critical enabler of both land and maritime operations (NATO, 2018, para. 22). Airpower's ability to conduct a broad array of tasks, from tactical ground support, logistical interdiction, to the strategic degradation of rear-echelon assets (UK MOD, 2021), has made it a force multiplier on modern battlefields and a prerequisite to military operations (Piatkowski et al., 2024, para. 1). Consequently, the development and

preservation of air capabilities are a central priority for advanced militaries, a trend reflected in the sustained global investment in aerospace research and procurement (ASD Europe, 2024, para. 2; Department of the Air Force, 2025, paras. 4-6).

These investments in air power have produced a paradox. Offensively, modern air forces employ platforms that are significantly faster, more networked, and more lethal to air- and ground forces than in previous generations (Ellis, 2016). On the defensive side, advancements integrated radar networks, electronic warfare (EW) capabilities, and low observability (LO) have made aircraft significantly more survivable in contested airspace (Kostis, 2024. p. 133). As a result, the destruction of aircraft in flight has become a difficult and resource-intensive task. The enhanced in-flight survivability of modern aircraft inadvertently shifts offensive focus toward a more fundamental vulnerability, the aircraft on the ground. Regardless of technological sophistication, an aircraft is dependent on fixed infrastructure for re-arming, refuelling, maintenance, and sortie generation. On the ground, it is a static, predictable target, exposed to pre-emptive strikes, thus exposing even the most advanced fleets to asymmetric risks. This vulnerability has been exacerbated by two concurrent trends, the maturation of persistent intelligence, surveillance, and reconnaissance (ISR) capabilities and the proliferation of PGMs. Adversaries can now detect, fix, and engage targets at fixed installations with unprecedented speed and accuracy (Van Hooft & Boswinkel, 2019). The strategic fragility of concentrated airpower has become visible by Russian losses during its invasion of Ukraine, where Ukrainian missile strikes on Russian airbases, such as the attack on Saky Air Base in Crimea (Schwirtz, 2022), demonstrated that a technologically sophisticated force can suffer significant attrition on the ground.

This strategic challenge is particularly acute for European and NATO forces, especially along the Eastern flank. Alliance air policing and forward-presence missions necessitate the concentration of high-value assets (NATO, 2025), such as fifth-generation F-35s, as well as tanker and AWACS aircraft (NATO Allied Air Command Public Affairs Office, 2025a, paras 3, 6.) at a limited number of Main Operating Bases (MOBs) (NATO Allied Air Command Public Affairs Office, 2025b, paras. 2, 18). Furthermore, several member states are modernising their air forces (Jepma, 2025; Lunday & Groeneveld, 2025; Pilgrim, 2025) without a proportional expansion of basing infrastructure (Zeigler et al., 2025), leading to a potential overconcentration of assets that creates potentially decisive targets in the opening phases of a peer-level conflict. In response to the threats posed to military infrastructure, defensive

planners have traditionally relied on a matrix of active and passive measures to deter or mitigate offensive operations.

Active Defences

Active defences, in the context of missile defence, rely on advanced kinetic interceptor systems that physically destroy incoming threats before impact (EBESCO, 2022, para. 1). Examples include the PATRIOT missile system, Terminal High Altitude Area Defence (THAAD), and the Iron Dome, each tailored to counter specific ranges and types of projectiles. While these systems are highly effective and play a vital role in layered defence strategies, they are finite in number. Massed, multi-vector, or sustained attacks can potentially saturate and overwhelm advanced active defence networks, a problem compounded by the unfavourable cost-exchange ratio of using expensive interceptors against relatively cheap drones or cruise missiles (Karako & Williams, 2017).

Passive Defences

Passive defences are designed to reduce damage and maintain operational continuity, should active defences become penetrated or rendered ineffective. These measures encompass a range of strategies aimed at complicating enemy targeting and protecting critical assets. Dispersal involves spreading aircraft across multiple locations, including austere or improvised airfields, to make adversary strikes less effective, a key principle of the Agile Combat Employment (ACE) concept (Richardson, 2025). Deception employs decoys, camouflage, and concealment to disrupt or mislead enemy targeting systems (Global Security, n.d.; Morgan, 2021; Pettyjohn, 2022). Hardening focuses on physically reinforcing infrastructure, most notably through the use of Hardened Aircraft Shelters (HAS), which shield aircraft and essential equipment from the effects of kinetic attacks (Pettyjohn, 2022). The argument for enhancing passive defences raises critical questions about resource allocation and operational benefit. While concepts like ACE address dispersal, the physical protection of assets at MOBs remains essential, particularly for ensuring continuity of operations under fire.

This study aims to provide a quantitative assessment of the protective value of HAS under conditions of a saturation missile attack. The central research objective is to determine the

extent to which HAS contribute to the survivability of air assets and the preservation of sortie-generation capacity during the initial stages of a high-intensity conflict. To achieve this, the report develops and employs a simulation-based assessment framework to model a NATO airbase under various attack profiles. The analysis systematically varies key parameters, including shelter density, the use of decoys, and the scale and precision of missile strikes to evaluate asset vulnerability. Incorporating probabilistic Monte Carlo methods and dynamic targeting logic, the model captures the stochastic nature of missile engagements and defensive responses. This approach moves beyond simple binary outcomes (hit or miss), by incorporating layered damage modelling, weapon proximity effects, and intercept probabilities to provide a nuanced understanding of how infrastructure and aircraft fare under realistic combat conditions. The findings are intended to offer policymakers and military planners data-driven insights into the marginal protective value of hardening infrastructure as a key component of airbase resilience.

2. Research Design

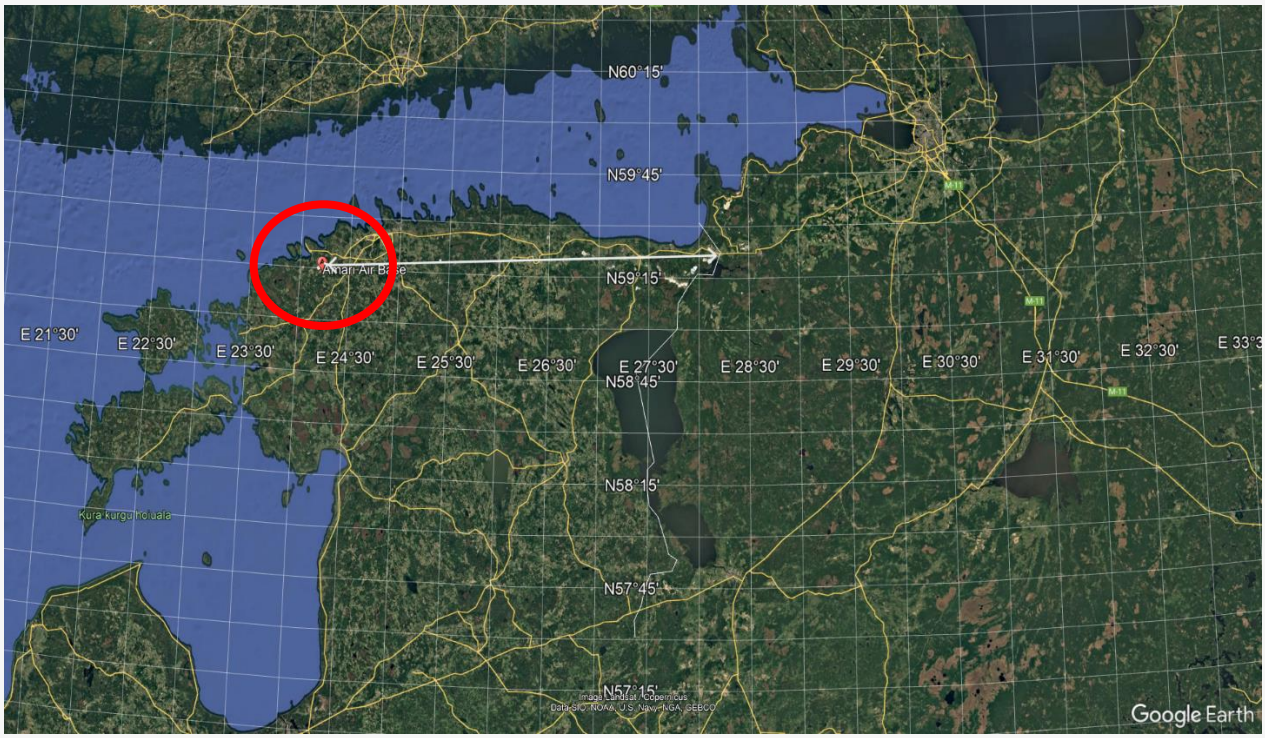
This study adopts a quantitative, simulation-based approach to evaluate the operational effectiveness of HAS in mitigating aircraft losses during saturation missile strikes in a peer-level conflict scenario. The analysis is motivated by the recognition that air assets remain critically dependent on fixed, and therefore vulnerable, infrastructure during pre-sortie operations, and the understanding that a prioritisation of active defence measures might introduce potential shortcomings in base defence. The study aims to reinforce the importance of passive measures in airbases' defence matrices to relevant parties. Consistent with earlier work in this field (Bracken, 1986; Holliday, 1990; Liu et al., 2022; Trzun & Vrdoljak, 2020), the simulation proposed in this design uses a probabilistic Monte Carlo framework. The simulation, developed in Python coding language (*see Appendix A.5*) incorporates dynamic missile targeting, active defence interception mechanics, and structural damage modelling over multiple attack waves and airbase configurations, allowing for comparative analyses of HAS- and non-HAS-protected aircraft under variable threat environments.

A central component of the simulation design is the selection of an appropriate reference airbase. This study has chosen to utilise Ämari airbase, Estonia, as the testbed for analysis, based on several operational factors that make it emblematic of the challenges facing NATO's forward-deployed air infrastructure in Eastern Europe. Geographically, the

airbase's proximity to the Russian border (~200km) places it within the engagement envelope of a broad spectrum of Russian PGMs, including short- to medium-range ballistic missiles and cruise missiles, allowing for the realistic modelling of compressed threat-warning timelines and the testing of HAS efficacy under broader saturation conditions.

Image 1

Geographical location of Ämari Airbase



In addition, Ämari serves as one of the primary operating locations for the Baltic Air Policing (BAP) missions (NATO Allied Air Command Public Affairs Office, 2025b, paras. 2, 1) and therefore regularly hosts rotational deployments of 4^{th-} and 5th-generation fighter aircraft from allied states (NATO Allied Air Command Public Affairs Office, 2025a, paras 3, 6.), making it a presumptive critical target in pre-emptive strike scenario's, it serves well as a representative case for broader NATO air infrastructure across the Eastern flank. As such, insights derived from simulations involving Ämari are not merely site-specific but may be broadly applicable to similarly configured NATO installations across Northern and Eastern Europe. This generalisability enhances the utility of the simulation results in informing alliance-wide infrastructure resilience and defence planning.

The basic layout of Ämari has been extracted from Google Earth using the manual creation of polygons and converted into a. shv-file for use in Python, creating a near 1:1

representation (see figure XX). On this two-dimensional model, entities (e.g. HAS structures, exposed aircraft) are assigned spatial coordinates based on the locations of aircraft aprons (3 in total, 25 parking spaces) and HAS (3 clusters, 21 total). Structural parameters, such as hardness thresholds and splash-damage vulnerability radii, are included as well. The core of the analysis is built on discrete-event simulation, wherein each missile-entity is processed sequentially during an attack-wave. Monte Carlo iterations are then used to allow for the random sampling of targeting choices, impact location, and interception outcomes variables, allowing the simulation to reflect the stochastic nature of airbase conflict and accommodate for the significant uncertainty inherent to strike accuracy and damage effects.

Image 2

Visual Representation of Ämari Air Base

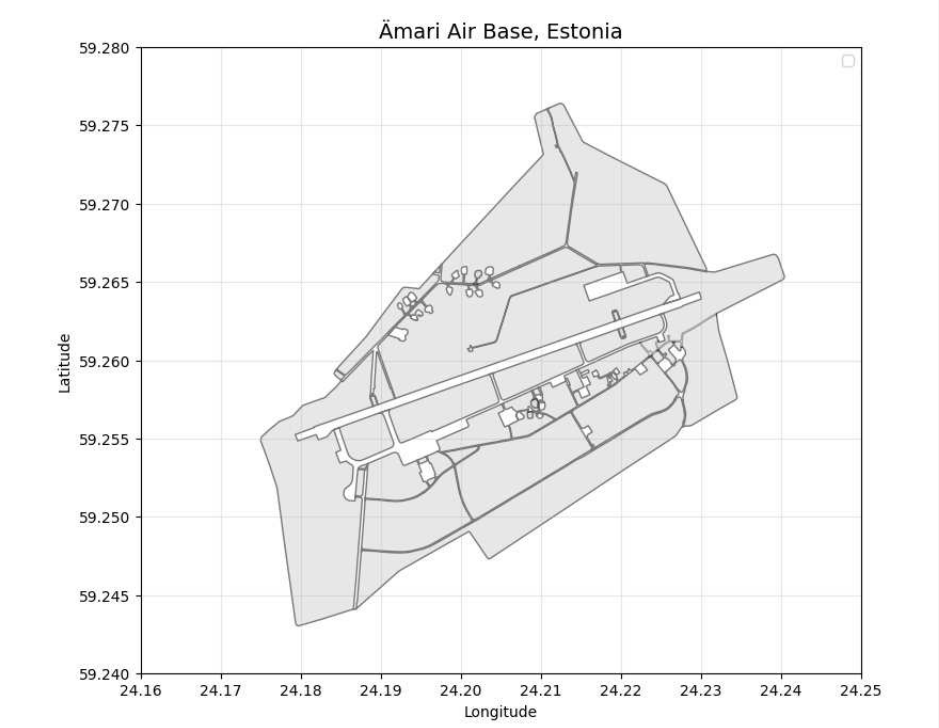
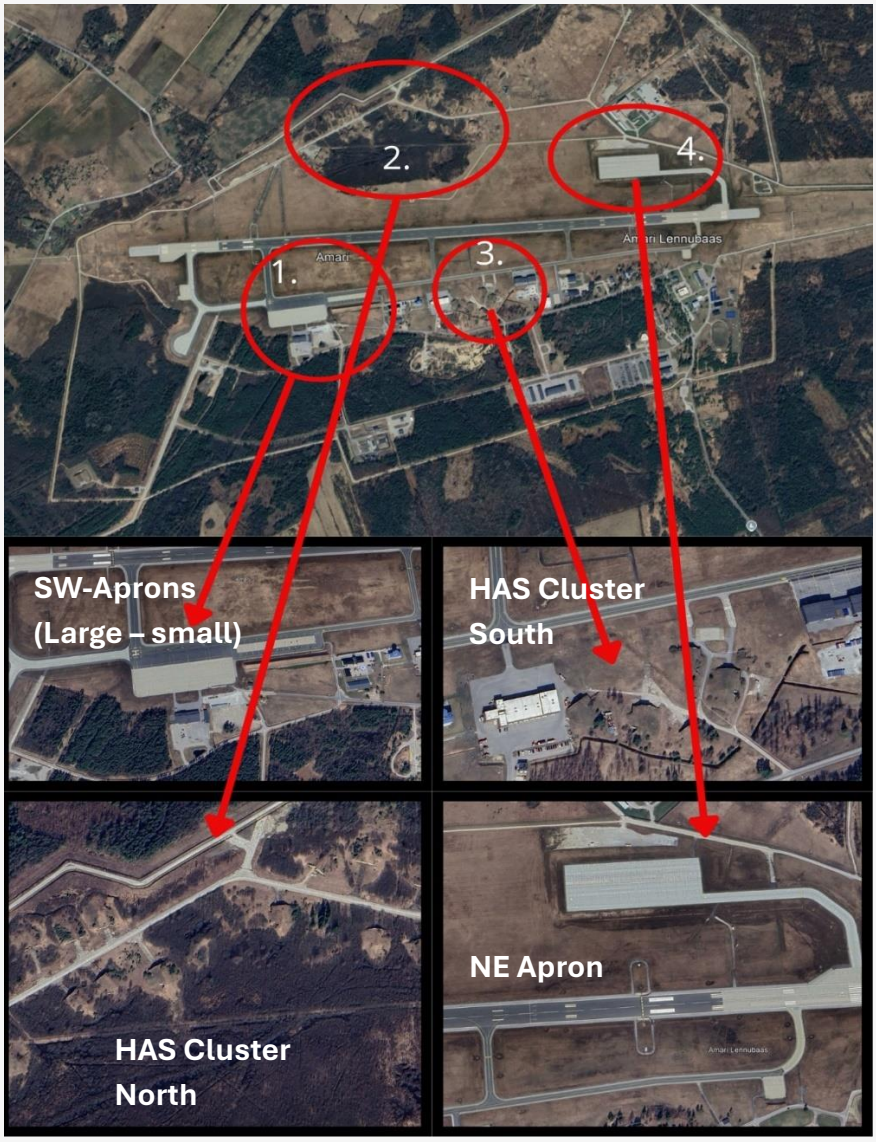


Image 3

Ämari Air Base Facilities



3. Targeting & Impact Probability Modelling

The simulation includes four distinct assets that are targeted by missiles:

Table 1
Target Dimensions and Weights

Target Type	Target Radius (m)	Target Weight	Weight Rationale
Exposed fighters	5.35; based on F-35 (wingspan / 2)	2	Higher priority than protected fighters due to ease of destruction and value.
HAS-protected fighters	30; based on Ämari HAS dimension (HAS-length / 2)	1	Lowest priority due to hardened protection, reducing the likelihood of kill.
Tanker aircraft	30; based on Airbus A330 MRTT (wingspan / 2)	3	High operational value as force multipliers.
AWACS	22; based on Boeing E-3 Sentry (wingspan / 2)	6	Highest strategic value; key enabler of air superiority; limited availability.

The individual missiles modelled by the simulation are assigned to targets through a weighted random function. The probability of selecting a given target is determined by target value weight (V_{tj}) and missile-type specific priorities (λ_m). For each missile (m), the probability of being assigned to target (t_j) is computed as:

$$P_{target}(t_j) = \frac{V_{tj} * \lambda_m}{\sum_{i=1}^{n} V_{t_k} * \lambda_{m,i}}$$

In this case, λ_m is a scalar reflecting missile specific preference. Missile preferences are based on the type of missile modelled by the simulation.

Table 2
Missile Types and Specifications

Missile type	CEP (m)	Warhead (kg)	Value weight	Weight Rationale
Iskander-M (1)	50	700	5	High warhead mass allows reliable HAS destruction; good for

				all target types including hardened.
Iskander-K (1)	50	480	3	Insufficient warhead to guarantee HAS kill; moderate accuracy and payload reduce effectiveness against hardened targets.
P-800 (2)	1.5	300	4	Excellent accuracy makes it suitable for high-value soft targets (AWACS, tankers); not ideal for HAS.
Kalibr (3)	12	400	3	Accurate but lacks HAS-kill warhead; suited for exposed or semi-hardened targets.
Kh-101 (4)	15	450	3	Precise, but sub-500kg warhead limits utility against HAS. Better for exposed fighters and support aircraft.
OTR-21 (5)	150	482	2	Poor accuracy and insufficient warhead for HAS; mostly useful against area targets or in saturation roles.

1. Center for Strategic and International Studies (2024a)
2. Vermylen (2017)
3. Missile Defense Advocacy Alliance (2017)
4. Center for Strategic and International Studies (2024b)
5. Center for Strategic and International Studies (2024c)

Once a target is assigned, the probability of a successful hit within the radius of the target is modelled using a Gaussian function based on the missile's Circular Error Probable (CEP)[1]. The missiles' impact error is sampled from a two-dimensional normal distribution, centred

[1] A missile's CEP is the radius within which 50% of munitions are expected to land, centered on the aimpoint (essentially a 50% confidence interval for targeting accuracy). For example, a CEP of 100 m means half of the munitions will land within 100 m of the target point.

on the target, with the standard deviation derived from missile system CEP specifications. The cumulative probability of hitting a target within the effective radius (r_t) is approximated as

$$P_{hit} = 1 - e^{(-\left(\frac{r_t}{k*CEP}\right)^2 * \ln(2))}$$

Where:

- r_t: target radius
- ln (2) = 0.6931
- k = 1.774, (conversion factor derived from Rayleigh's cumulative distribution function ($F_{(r)} = 1 - e^{-x^2/2\sigma^2}$), so that missiles striking chance within the CEP = always 0.5*m_n)

4. Kill Logic

The damage assessment after a missile strike follows a dual-mode function that distinguishes between exposed aircraft and HAS-protected aircraft. Exposed aircraft are considered to be destroyed when struck directly (i.e. within r_2) or when subjected to sufficient overpressure/fragmentation damage from nearby impacts (see R_{blast}). In contrast, HAS shelters are engineered to withstand specific warhead masses and/or a defined number of repeated impacts. Analysis by Swisdak et al. (1994) reports resistance to warheads of approximately 430 kg, albeit with structural damage indicative of imminent failure. Given the classified nature of NATO STANAG specifications governing HAS construction, this simulation adopts a nominal destruction threshold of 500 kg. Under this assumption, HAS structures are modelled as capable of withstanding munitions below 500 kg, though such impacts still inflict cumulative structural degradation. This degradation is recorded via a damage counter, accounting for both blast effects from nearby detonations (see "R_{blast}") or direct hits

$$D_t \leftarrow D_t + \alpha * W_m$$

Where:

- Dt is the accumulated damage counter,
- *Wm* is the missile's warhead mass;
- α is a scaler which models the impact energy weakening due to HAS's reinforced nature.

Once the accumulated damage exceeds the hardness threshold (H_t/500kg), a probabilistic destruction check is performed

$$P_{kill} = \begin{cases} p, if\ D_t \geq H_t \\ 0\ otherwise \end{cases}$$

Where:

- P_{kill} is the probability of kill;
- H_t is the hardness threshold (500kg);
- p is the kill probability when the shelter has been breached (0.8 in this simulation.

When a missile fails to hit its target, the simulation employs a physics-based blast damage model grounded in cube-root scaling, derived from explosive energy dispersion physics per the Taylor-Sedov solution. Unlike simplified linear or square-root models, this approach aims to capture the three-dimensional nature of blast propagation. The blast radius is calculated as

$$R_{blast} = \beta * \sqrt[3]{W_m}$$

Where:

- W_m is TNT-equivalent of missile warhead,
- β is an empirical scaling factor, denotating a three-zone damage model:
    - Lethal zone ($3 * \sqrt[3]{W_m}$): (High destruction probability due to overpressure and thermal effects.
    - Damage zone ($5 * \sqrt[3]{W_m}$): Moderate structural damage with decreasing probability by distance.
    - Fragment zone ($8 * \sqrt[3]{W_m}$): Outer region affected by fragments and light damage.

Damage probability is modelled with continuous, distance-based functions, not binary thresholds, and incorporates target-specific vulnerability factors. For instance, exposed aircraft have a vulnerability of 1.0, while hardened aircraft shelters (HAS) reduce vulnerability to 0.25 due to structural protection. Warhead types are differentiated by TNT equivalency factors,e.g., Fuel-Air Explosive warheads (FAE) = 1.3, Fragmentation warheads (FRAG) = 0.7, reflecting varying overpressure, fragment spread, or penetration behaviour. Compared to linear models, the inclusion of cube-root scaling results in significantly

realistic and thus larger affected areas. For example, a 700 kg Iskander-M warhead will yield a lethal damage radius of >26m, damage radius of >44m, and a fragment zone of >71m.

5. Interception modelling

The simulation includes a basic air defence component, representing SAM batteries, in this case X Patriot batteries, each with X launchers with a capacity for X missiles. Ämari is assigned a fixed number of interceptors per run, divided equally across missile waves. Each incoming missile is assigned an interception probability ($P_{intercept}$), which is modified by missile priority (i.e. heavier warheads are more likely to be prioritized). Interception success itself is modelled as a Bernoulli trial for each missile

$$I_m = Bernoulli(P_{intercept})$$

If I_m = 1, the missile is neutralized, and no further impact is processed. Once the pool of available interceptors is exhausted, subsequent missiles proceed unimpeded, allowing the simulation to not only capture saturation effects, but exhaustion effects as well.

6. Scenario Parameters

This study models and compares two airbase defence configurations at Ämari Air Base. In the baseline scenario the airbase maintains its current operational configuration. The aircraft inventory totals 34, dispersed between exposed apron areas and existing HAS infrastructure. Eighteen aircraft remain exposed, including seven fighters on the northeast apron, four large support aircraft, comprised of one Boeing E-3 AWACS and three Airbus A330 MRTTs on the south large apron, four fighters on the south small apron, and three fighters on the south-southeast apron. The remaining fifteen aircraft are housed within HAS, including five in the northwest cluster and ten distributed across other HAS locations on the base. Additionally, four HAS remain intentionally empty to serve as decoys, intended to mislead enemy targeting algorithms. The incoming missile threat in both scenarios comprises thirty-four precision-guided munitions, ten Iskander-Ms, ten Iskander-Ks, five OTR-21s, five Kh-101 cruise missiles, two Kalibr missiles, and two P-800s. This threat package maintains an approximate 1:1 missile-to-aircraft ratio, aligned with doctrinal principles for overwhelming airbase defences and ensuring strike redundancy. The

enhanced scenario modifies this setup by relocating four fighters from exposed apron positions into newly activated HAS, bringing the total number of sheltered aircraft to nineteen and reducing the exposed count to fourteen. The total number of aircraft and decoy shelters remains unchanged. This enhanced configuration models a more robust passive defence strategy without altering the base's active defence systems.

7. Defensive Parameters and Simulation Design

Both scenarios utilise identical active defences to isolate the impact of changes in HAS utilisation. The airbase is defended by a Patriot battery composed of six launchers with a total of twenty-four interceptors. Each interceptor engagement has a 41 percent success probability. The simulated attacks are structured into six waves, and heavier missiles (those with warheads 500 kilograms) are prioritised for interception. To analyse the outcomes, each scenario is subjected to 10,000 Monte Carlo iterations, allowing for comprehensive modelling of stochastic variation in missile targeting, blast effects, interception outcomes, and aircraft survival.

Table 3
Scenario Parameters

Parameter	Scenario 1	Scenario 2
Aircraft in HAS	15	19
Aircraft Exposed	19	15
Empty HAS structures	4	4
Total Interceptors	34	34
Defence-Interception Rate	41%	41%

8. Results and Analysis

The simulation reveals stark differences in aircraft survivability across the two scenarios. In the baseline case, scenario 1, an average of 11.94 aircraft are destroyed per attack, representing approximately 35.1 percent of the total fleet. Of these, around 7.75 losses occur among exposed aircraft, while 4.19 are attributed to hits on HAS-protected assets. This yields a loss ratio of approximately 1.85 to 1 between exposed and HAS-sheltered aircraft, confirming the protective value of hardened shelters even under intense

bombardment. By comparison, the enhanced scenario (scenario 2) achieves measurable improvement. Aircraft losses fall to an average of 11.39 per attack, or 34.5 percent of total aircraft. The exposed segment sees a reduced loss of 5.55 aircraft, while losses among HAS-protected aircraft drop to 3.61. This shifts the exposed-to-HAS loss ratio to 1.54 to 1, indicating that beyond simple numerical sheltering gains, increased HAS density provides a compounding benefit to overall survivability.

Further analysis of HAS performance in the baseline configuration shows considerable variability in protection outcomes depending on shelter location. Some HAS positions suffer destruction rates as high as 67 percent, while others remain as low as 21 percent. These differences are attributed to proximity to high-value targets and repeated targeting within missile salvos. On average, shelters sustaining heavy losses were hit 1.8 to 2.1 times per simulation run, approaching or surpassing structural damage thresholds modelled for cumulative impacts. In contrast, the enhanced scenario demonstrates improved survivability across nearly all HAS positions. The additional shelters dilute missile targeting probability, with most HAS experiencing destruction rates below 40 percent. However, this benefit exhibits diminishing returns. As shelter numbers increase, the marginal protective gain declines, suggesting that beyond a certain density, the strategic benefit of adding more HAS tapers off. Splash damage contributes to the destruction of 1.2 to 1.5 aircraft per scenario, primarily affecting exposed aircraft near direct impact zones. Larger warheads, particularly the 700kg Iskander-M, produce blast radii exceeding 100 meters, creating overlapping damage areas on crowded aprons. HAS-protected aircraft fare significantly better in this regard, showing vulnerability levels approximately one-quarter that of their exposed counterparts. In high-density apron areas, this differential survivability becomes even more pronounced, with single missile strikes often resulting in multiple aircraft losses.

Evaluating missile type effectiveness, the simulation confirms that heavy ballistic missiles like the Iskander-M are the most capable of defeating HAS protection, averaging 2.1 aircraft kills per missile when blast effects are included. Their payloads exceed the modelled HAS hardening threshold, making them the primary threat to sheltered assets. Precision cruise missiles such as the P-800, Kalibr, and Kh-101, while less effective against HAS, excel at targeting exposed high-value assets. These missiles demonstrate destruction probabilities above 80 percent when directed at unprotected AWACS or tanker aircraft, with targeting algorithms correctly prioritising such assets.

The PATRIOT system provides moderate but insufficient mitigation. On average, 14 to 15 missiles are intercepted per scenario, roughly 41 percent of incoming threats. While prioritisation of heavier missiles slightly improves HAS survival, the overall defence remains saturated. With 34 inbound munitions and only 24 interceptors, at least 10 missiles inevitably penetrate the system, reinforcing the limitations of active defence during saturation strikes. Empty HAS used for deception show meaningful, albeit limited, impact. Each decoy shelter attracts between 0.8 and 1.2 missile strikes per simulation, and destruction rates of 40 to 50 percent suggest that enemy targeting systems are partially deceived. However, the limited number of decoys constrains their overall contribution to survivability. The results imply that expanding the number of decoy shelters could serve as a cost-effective way to increase base resilience due to defence economics and fog of war dynamics.

9. Strategic Implications

The enhanced HAS configuration produces a modest yet meaningful improvement in survivability: a 4.6 percent increase overall, with 0.55 more aircraft surviving per attack, on average. Though seemingly minor in percentage terms, this gain carries significant operational weight, considering the high cost and limited availability of modern combat aircraft. More importantly, the simulation reveals that HAS effectiveness is not strictly linear. Adding shelters not only reduces the number of exposed aircraft but also produces beneficial distributed effects that improve the survivability of all assets on base. The improved loss ratios and reduced splash damage in the enhanced scenario suggest that greater HAS utilisation contributes to a more resilient defence posture across the board. These findings affirm the relevance of Cold War-era passive defence infrastructure in modern threat environments. They also highlight the necessity for adaptive strategies that integrate hardened shelters, realistic deception, and active defence systems. Under saturation strike conditions, reliance on a single line of defence is inadequate. Instead, survival depends on a layered approach that maximises protection at every level of the engagement envelope.

Table 4
Scenario Comparison

Metric	Scenario 1	Scenario 2	Change
Valid Simulation Runs	10.000	10.000	N/A
Avg. Aircraft Lost per Attack	11.94 (35.1%)	11.39 (34.5%)	↓ 0.55 aircraft (-4.6%)
Exposed Aircraft Losses (avg)	7.75	5.55	↓ 2.2
HAS Aircraft Losses (avg)	4.19	3.61	↓ 0.58
Exposed-to-HAS Loss Ratio	1.85:1	1.54:1	Improved survivability
Missiles Intercepted (avg)	14–15 (41% intercept rate)	14–15 (41% intercept rate)	No change
Minimum Missiles Penetrating Defences	≥10	≥10	No change
Avg. Splash Damage Aircraft Losses	1.35	1.20	↓ 0.15 due to reduced clustering
HAS Destruction Rate Range	21% – 67%	18% – 39%	↓ overall
Avg. Missile Hits on Vulnerable HAS Locations	1.95	1.50	↓ 0.45 due to shelter dilution
Avg. Decoy HAS Strikes per Simulation	1.00	1.00	No change
Avg. Effectiveness of Heavy Missiles (e.g., Iskander-M)	2.10 kills/missile	2.00	↓ 0.10
High-Value Exposed Target Kill Probability (AWACS/Tanker)	~ 85%	~ 78%	↓ ~ 7% due to reduced exposure

10. Limitations and Future Studies

While the simulation framework developed in this study provides valuable quantitative insights into the protective value of HAS under saturation missile attack conditions, several limitations constrain the scope and generalisability of the findings. First, the simulation relies on open-source or nominal data for HAS hardness thresholds, warhead effects, and missile system performance. Classified specifications for NATO-standard shelters and weapon systems could meaningfully alter modelled survivability outcomes. Similarly, the assumed 500 kg hardening threshold and cumulative damage model represent a plausible but simplified proxy for real-world structural performance, which may vary with

construction quality, warhead type, and impact geometry. Second, the model does not incorporate detailed logistics, maintenance, and operational sortie generation constraints that could affect post-attack airbase functionality. Survival of aircraft in the simulation equates to asset availability, but damaged runways, fuel depots, or maintenance facilities might render surviving aircraft temporarily unusable. The results therefore represent a best-case estimate of post-strike force availability. Third, the active defence modelling uses a static 41 percent interception probability and fixed missile prioritisation logic. Real-world performance would be influenced by sensor coverage, engagement doctrine, decoy discrimination, and adversary countermeasures, all of which could shift the balance between active and passive defence effectiveness. The enemy targeting algorithms in the simulation also employ fixed weights rather than adaptive or learning behaviours, which underrepresents the capacity of advanced ISR and strike systems to optimise targeting over successive salvos.

Future studies could address these limitations in several ways. Expanding the model to include runway cratering, fuel infrastructure targeting, and repair timelines would provide a more holistic measure of operational resilience. Enhancing adversary targeting logic with adaptive algorithms could better simulate an intelligent, learning opponent. Sensitivity analyses on interceptor allocation, decoy density, and dispersal strategies would illuminate cost-effective combinations of passive and active defences. Finally, considering the strategic developments of warfare, future simulations should include drones alongside PGMs, reflecting their growing role in reconnaissance, target designation, and direct attack. The integration of loitering munitions, swarming tactics, and unmanned strike platforms could significantly alter airbase vulnerability profiles, particularly by saturating defences, exploiting detection gaps, and targeting high-value assets with lower-cost systems.

Bibliography

ASD Europe. (2024). *Research & development*.

https://www.asd-europe.org/news-media/facts-figures/research-development/

Bracken, J. (1986). *Monte Carlo Layered Defense Model* (No. ADA175217). Defence Technical Information Center.

https://apps.dtic.mil/sti/citations/ADA175217

Center for Strategic and International Studies. (2024a). *9K720 Iskander (SS-26) | Missile Threat*. CSIS Missile Threat.

https://missilethreat.csis.org/missile/ss-26-2/

Center for Strategic and International Studies. (2024b). *KH-101 / KH-102 | Missile Threat*. Missile Threat.

https://missilethreat.csis.org/missile/kh-101-kh-102/

Center for Strategic and International Studies. (2024c). *OTR-21 Tochka (SS-21) | Missile threat*. Missile Threat.

https://missilethreat.csis.org/missile/ss-21/

Department of the Air Force. (2025). *Budget Request Fiscal Year 2026*.

https://www.saffm.hq.af.mil/FM-Resources/Budget/#:~:text=The%20FY%202026%20U.S.%20The,many%20of%20the%20core%20functions.

EBESCO. (2022). Antiballistic missile defense systems.

https://www.ebsco.com/research-starters/military-history-and-science/antiballistic-missile-defense-systems

Ellis, R. W. (2016). *Building the Future Air Force: Analysis of Platform versus Weapon Development* [Monograph]. United States Army Command and General Staff College.

https://apps.dtic.mil/sti/tr/pdf/AD1021962.pdf

Global Security. (n.d.). *Chapter 4: Passive Air Defense Measures*.

https://www.globalsecurity.org/military/library/policy/army/fm/44-8/ch4.htm

Gluck, M. (2024, April 23). *9K720 Iskander (SS-26) | Missile Threat*. Missile Threat.

https://missilethreat.csis.org/missile/ss-26-2/

Holliday, M. R. (1990). Methodology of an event-driven Monte Carlo missile simulation. *Mathematical and Computer Modelling*, *14*, 1123–1128.

https://doi.org/10.1016/0895-7177(90)90352-n

Jepma, L. (2025, July 2). *A new chapter for the Royal Netherlands Air Force: 112 years of history and a leap into the future*. Nederlands Lucht- En Ruimtevaartcentrum.

https://www.nlr.org/newsroom/nieuws/royal-netherlands-air-force-112-years-of-history/#:~:text=For%20NLR%2C%20this%20strong%20commitment,Keuning%2C%20Director%20Space%20at%20NLR.

Karako, T, & Williams, I. (2017). The Missile Defense Review: Insufficient for Complex and Integrated Attack. *Strategic Studies Quarterly*, *3*(2), 3–15.

https://www.jstor.org/stable/26639670?casa_token=i114U6PoOtkAAAAA%3ApbKt3U3HGDkMqsHmns5BESmkiue6w8Rsx-id8Ob71LCmbzsDvvsdr0jZSKwlGaIjtvmUXr_EW9krJHWyG5M9x9JBUzrwSP1gLqnhFySEZ9aKNDc6kX0&seq=1

Kostis, T. (2024). The Future of Stealth Military Doctrine. *JFQ*, *116*(1).

https://digitalcommons.ndu.edu/cgi/viewcontent.cgi?article=1199&context=joint-force-quarterly

Liu, Y., Xiong, Z., Wang, J., & Wang, D. (2022). Monte Carlo-Based analysis and experimental validation of the Interception-Damage probability of the new active interception Net. *Mathematical Problems in Engineering*, *2022*, 1–12.

https://doi.org/10.1155/2022/5438023

Lunday, C., & Groeneveld, J. (2025, July 11). Germany weighs buying more F-35 fighter jets from the US. *POLITICO*.

https://www.politico.eu/article/germany-plan-buy-f35-fighter-jet-united-states/

Missile Defence Advocacy Alliance. (n.d.). *P-800 ONIKS (SS-N-26 Strobile)*.

https://missiledefenseadvocacy.org/missile-threat-and-proliferation/todays-missile-threat/russia/16962-2/

Morgan, H. (2021, September 6). The primacy of passive air defense. *Modern War Institute*.

https://mwi.westpoint.edu/the-primacy-of-passive-air-defense/

NATO. (2018). *NATO's Joint Air Power Strategy*.

https://www.nato.int/cps/fr/natohq/official_texts_156374.htm?selectedLocale=en

NATO. (2025). *NATO Air Policing*.

https://www.nato.int/cps/en/natohq/topics_132685.htm

NATO Allied Air Command Public Affairs Office. (2025a). *Estonian and Dutch collaborate during Agile Combat Employment training at Åmari, Estonia*.

https://ac.nato.int/archive/2025-2/estonian-and-dutch-collaborate-during-agile-combat-employment-training-at-aamari--estonia

NATO Allied Air Command Public Affairs Office. (2025b). *NATO's Air Policing Mission: A Steadfast Commitment by the Alliance*.

https://ac.nato.int/archive/2025-2/natos-air-policing-mission-a-steadfast-commitment-by-the-alliance-#:~:text=A%20notable%20operational%20highlight%20occurred,Air%20and%20Missile%20Defence%20System.

Pettyjohn, S. L. (2022, January 10). *Spiking the Problem: Developing a Resilient Posture in the Indo-Pacific with Passive Defenses - War on the Rocks*. War on the Rocks.

https://warontherocks.com/2022/01/spiking-the-problem-developing-a-resilient-posture-in-the-indo-pacific-with-passive-defenses/#:~:text=Passive%20defenses%20minimize%20the%20damage,before%20it%20reaches%20its%20target.

Piatkowski, M., Piatkowski, M., & Goździewicz, W. (2024, September 6). Precautions and aerial superiority or supremacy. *Lieber Institute West Point*.

https://lieber.westpoint.edu/precautions-aerial-superiority-supremacy/

Pilgrim, L. (2025, July 4). *The UK's F-35 Procurement Strategy: A Balancing Act*. Wavell Room.

https://wavellroom.com/2025/07/04/the-uks-f-35-procurement-strategy-a-balancing-act/

Priebe, M., Vick, A. J., Heim, J. L., & Smith, M. L. (2019, July 17). *Distributed Operations in a Contested Environment: Implications for USAF Force presentation*. RAND.

https://www.rand.org/pubs/research_reports/RR2959.html

Richardson, I. D. (2025). Protecting ACE: Air Defence and Agile Combat Employment. *JFQ*, *117*(2).

https://digitalcommons.ndu.edu/cgi/viewcontent.cgi?article=1243&context=joint-force-quarterly

Schwirtz, M. (2022). Damage in Air Base Blasts Appears Worse Than Russia Claimed. *The New York Times*.

https://www.nytimes.com/live/2022/08/10/world/ukraine-russia-news-war

Swisdak, M. M., Jacobs, E. M., & Ward, J. M. (1994). *Hazard Ranges for Small Net Explosive Quantities in Hardened Aircraft Shelters*. DTIC.

https://apps.dtic.mil/sti/pdfs/ADA507453.pdf

Trzun, Z., & Vrdoljak, M. (2020). Monte Carlo Simulation of Missile Trajectories Dispersion due to Imperfectly Manufactured Warhead. In *Annals of DAAAM for . . . & proceedings of the . . . International DAAAM Symposium* (pp. 0574–0583).

https://doi.org/10.2507/31st.daaam.proceedings.079

UK MOD. (2021). *Joint Doctrine Publication 0-30*.

https://assets.publishing.service.gov.uk/media/636baad0d3bf7f1649c4e36d/UK_Air_Power_JDP_0_30.pdf

US DOD. (2015). *Joint Publication 1-02: Dictionary of Military and Associated Terms*.

https://irp.fas.org/doddir/dod/jp1_02.pdf

Van Hooft, P., & Boswinkel, L. (2019). *Surviving the Deadly Skies: Integrated Air and Missile Defence 2021-2035* (No. 9789492102898). The Hague Centre for Strategic Studies.

https://www.researchgate.net/publication/356981510_Surviving_the_Deadly_Skies_Integrated_Air_and_Missile_Defence_2021-2035

Vermylen, M. (2017). *P-800 OnIkS (SS-N-26 Strobile)*. Missile Defense Advocacy Alliance.

https://missiledefenseadvocacy.org/missile-threat-and-proliferation/todays-missile-threat/russia/16962-2/

Wilkening, D. A. (2000). A simple model for calculating ballistic missile defence effectiveness. *Science and Global Security*, *8*(2), 183–215.

https://doi.org/10.1080/08929880008426475

Zeigler, S. M., Phillips, D., Hoehn, J., Hagem, J., Edenfield, N., Hill, D., & Doll, A. (2025). *Assessing Progress on Air Base Defence* (No. RRA3142-1).

https://www.rand.org/content/dam/rand/pubs/research_reports/RRA3100/RRA3142-1/RAND_RRA3142-1.pdf

APPENDIX

A.1

Scenario 1 Output

Manual HAS Assignment 1534 (44.1%)
Manual EXPOSED Assignment 1934 (55.9%)
Empty HAS Structures 4
Total Aircraft 34
Total Structures 38
Total Missiles 34
Valid Simulation Runs 10000

DESTRUCTION SUMMARY
  Aircraft Destroyed 9.15
    - In HAS 0.09
    - Exposed 9.05
    - From Splash 7.06
  Empty HAS Destroyed 0.02

BY MISSILE TYPE
  Iskander-K 2.94 destroyed, 3.45 intercepted, 0.00 skipped
  Kalibr 1.54 destroyed, 0.91 intercepted, 0.00 skipped
  Iskander-M 0.08 destroyed, 4.49 intercepted, 0.00 skipped
  P-800 3.59 destroyed, 0.82 intercepted, 0.00 skipped
  OTR-21 0.14 destroyed, 1.72 intercepted, 0.00 skipped
  Kh-101 0.88 destroyed, 0.00 intercepted, 0.00 skipped

BY AIRCRAFT TYPE
  tanker 2.38
  awacs 0.81
  fighterEXP 5.87
  fighterHAS 0.09

HAS DAMAGE ANALYSIS
  HAS 20 (fighterHAS) 0.32 avg hits, 0.7% destruction rate
  HAS 21 (fighterHAS) 0.32 avg hits, 0.5% destruction rate
  HAS 22 (fighterHAS) 0.32 avg hits, 0.5% destruction rate
  HAS 23 (fighterHAS) 0.32 avg hits, 0.6% destruction rate
  HAS 24 (fighterHAS) 0.32 avg hits, 0.5% destruction rate
  HAS 25 (fighterHAS) 0.31 avg hits, 0.5% destruction rate
  HAS 26 (fighterHAS) 0.33 avg hits, 0.6% destruction rate
  HAS 27 (fighterHAS) 0.33 avg hits, 0.5% destruction rate

HAS 28 (fighterHAS) 0.33 avg hits, 0.5% destruction rate

HAS 29 (fighterHAS) 0.31 avg hits, 0.5% destruction rate
HAS 30 (fighterHAS) 0.32 avg hits, 0.5% destruction rate
HAS 31 (fighterHAS) 0.31 avg hits, 0.6% destruction rate
HAS 32 (fighterHAS) 0.33 avg hits, 0.9% destruction rate
HAS 33 (fighterHAS) 0.32 avg hits, 1.1% destruction rate
HAS 34 (fighterHAS) 0.33 avg hits, 0.5% destruction rate

EXPOSED AIRCRAFT ANALYSIS
EXPOSED 1 (fighterEXP) 0.08 avg hits [P-800 0.05, Kalibr 0.01, Kh-101 0.01, Iskander-K 0.01], 35.3% destruction rate, 7.8% direct hits, 27.5% splash kills
EXPOSED 2 (fighterEXP) 0.08 avg hits [Kalibr 0.01, Kh-101 0.01, P-800 0.05, Iskander-K 0.01, OTR-21 0.00], 38.1% destruction rate, 8.0% direct hits, 30.1% splash kills
EXPOSED 3 (fighterEXP) 0.07 avg hits [Iskander-K 0.00, P-800 0.05, Kalibr 0.01, Kh-101 0.01, OTR-21 0.00], 42.3% destruction rate, 7.4% direct hits, 34.9% splash kills
EXPOSED 4 (fighterEXP) 0.07 avg hits [Iskander-K 0.00, P-800 0.05, OTR-21 0.00, Kalibr 0.01, Kh-101 0.01], 46.9% destruction rate, 7.3% direct hits, 39.6% splash kills
EXPOSED 5 (fighterEXP) 0.07 avg hits [P-800 0.04, Kh-101 0.01, Kalibr 0.02, Iskander-K 0.01, OTR-21 0.00], 45.0% destruction rate, 7.1% direct hits, 38.0% splash kills
EXPOSED 6 (fighterEXP) 0.08 avg hits [P-800 0.05, Kalibr 0.01, Iskander-K 0.00, Kh-101 0.01, OTR-21 0.00], 36.1% destruction rate, 7.6% direct hits, 28.4% splash kills
EXPOSED 7 (fighterEXP) 0.08 avg hits [P-800 0.05, Iskander-K 0.00, Kalibr 0.01, Kh-101 0.01, OTR-21 0.00], 31.5% destruction rate, 8.1% direct hits, 23.4% splash kills
EXPOSED 8 (awacs) 0.30 avg hits [Kalibr 0.05, Iskander-K 0.15, P-800 0.05, Kh-101 0.03, OTR-21 0.01], 81.0% destruction rate, 29.9% direct hits, 51.1% splash kills
EXPOSED 9 (tanker) 0.18 avg hits [Iskander-K 0.12, Kalibr 0.03, Kh-101 0.01, P-800 0.02, OTR-21 0.00], 83.7% destruction rate, 18.4% direct hits, 65.3% splash kills
EXPOSED 10 (tanker) 0.19 avg hits [Iskander-K 0.12, Kalibr 0.03, P-800 0.02, Kh-101 0.02, OTR-21 0.01], 82.2% destruction rate, 19.0% direct hits, 63.3% splash kills
EXPOSED 11 (tanker) 0.22 avg hits [Iskander-K 0.13, P-800 0.03, Kalibr 0.03, Kh-101 0.03, OTR-21 0.01], 71.8% destruction rate, 22.4% direct hits, 49.4% splash kills
EXPOSED 12 (fighterEXP) 0.04 avg hits [P-800 0.03, Kalibr 0.01, Iskander-K 0.00, Kh-101 0.00, OTR-21 0.00], 67.8% destruction rate, 4.0% direct hits, 63.8% splash kills
EXPOSED 13 (fighterEXP) 0.05 avg hits [Kalibr 0.01, P-800 0.03, Iskander-K 0.00, Kh-101 0.01, OTR-21 0.00], 57.4% destruction rate, 5.0% direct hits, 52.4% splash kills
EXPOSED 14 (fighterEXP) 0.08 avg hits [P-800 0.05, Kh-101 0.01, Kalibr 0.01, Iskander-K 0.00, OTR-21 0.00], 35.9% destruction rate, 7.6% direct hits, 28.2% splash kills
EXPOSED 15 (fighterEXP) 0.09 avg hits [Iskander-K 0.00, P-800 0.06, Kalibr 0.01, Kh-101 0.01, OTR-21 0.00], 20.7% destruction rate, 8.7% direct hits, 12.0% splash kills
EXPOSED 16 (fighterEXP) 0.08 avg hits [P-800 0.05, Kalibr 0.01, OTR-21 0.00, Iskander-K 0.01, Kh-101 0.01], 32.4% destruction rate, 8.3% direct hits, 24.1% splash kills
EXPOSED 17 (fighterEXP) 0.08 avg hits [Kh-101 0.01, P-800 0.05, Kalibr 0.01, Iskander-K 0.00, OTR-21 0.00], 32.4% destruction rate, 7.9% direct hits, 24.6% splash kills
EXPOSED 18 (fighterEXP) 0.08 avg hits [P-800 0.05, Kalibr 0.01, Kh-101 0.01, Iskander-K 0.01, OTR-21 0.00], 32.4% destruction rate, 8.5% direct hits, 23.9% splash kills

EXPOSED 19 (fighterEXP) 0.08 avg hits [Kalibr 0.01, P-800 0.05, Kh-101 0.01, Iskander-K 0.00, OTR-21 0.00], 32.4% destruction rate, 7.8% direct hits, 24.6% splash kills

EXPOSED SUMMARY
Average destruction rate 47.7%
Average hits per exposed aircraft 0.11
Total direct hit kills 2.01
Total splash kills 7.05

EMPTY HAS ANALYSIS
EMPTY HAS 1 0.14 avg hits [Iskander-K 0.05, Kh-101 0.03, Kalibr 0.03, P-800 0.03, OTR-21 0.00], 0.5% destruction rate
EMPTY HAS 2 0.15 avg hits [P-800 0.03, Kh-101 0.04, Iskander-K 0.05, Kalibr 0.03, OTR-21 0.00], 0.6% destruction rate
EMPTY HAS 3 0.15 avg hits [Iskander-K 0.05, P-800 0.03, Kh-101 0.03, Kalibr 0.03, OTR-21 0.00], 0.4% destruction rate
EMPTY HAS 4 0.14 avg hits [Kh-101 0.03, P-800 0.03, Iskander-K 0.05, Kalibr 0.03, OTR-21 0.00], 0.4% destruction rate

EMPTY HAS SUMMARY
Average destruction rate 0.5%
Average hits per empty HAS 0.14
Total empty HAS destroyed 0.02
Deception effectiveness 99.5% survival rate

A.2

Scenario 2 Output

Manual HAS Assignment: 19/34 (55.9%)
Manual EXPOSED Assignment: 15/34 (44.1%)
Empty HAS Structures: 4
Total Aircraft: 34
Total Structures: 38
Total Missiles: 34
Valid Simulation Runs: 10000

DESTRUCTION SUMMARY:
Aircraft Destroyed: 7.19
- In HAS: 0.08
- Exposed: 7.11
- From Splash: 5.21
Empty HAS Destroyed: 0.02

BY MISSILE TYPE:

Iskander-K: 2.84 destroyed, 3.43 intercepted, 0.00 skipped
Kalibr: 1.19 destroyed, 0.90 intercepted, 0.00 skipped
P-800: 2.37 destroyed, 0.82 intercepted, 0.00 skipped
Kh-101: 0.63 destroyed, 0.00 intercepted, 0.00 skipped
Iskander-M: 0.06 destroyed, 4.48 intercepted, 0.00 skipped
OTR-21: 0.13 destroyed, 1.72 intercepted, 0.00 skipped

BY AIRCRAFT TYPE:
tanker: 2.49
awacs: 0.84
fighterEXP: 3.78
fighterHAS: 0.08

HAS DAMAGE ANALYSIS:
HAS 16 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 17 (fighterHAS): 0.28 avg hits, 0.4% destruction rate
HAS 18 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 19 (fighterHAS): 0.28 avg hits, 0.4% destruction rate
HAS 20 (fighterHAS): 0.28 avg hits, 0.3% destruction rate
HAS 21 (fighterHAS): 0.28 avg hits, 0.4% destruction rate
HAS 22 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 23 (fighterHAS): 0.29 avg hits, 0.4% destruction rate
HAS 24 (fighterHAS): 0.27 avg hits, 0.3% destruction rate
HAS 25 (fighterHAS): 0.28 avg hits, 0.4% destruction rate
HAS 26 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 27 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 28 (fighterHAS): 0.29 avg hits, 0.7% destruction rate
HAS 29 (fighterHAS): 0.26 avg hits, 1.0% destruction rate
HAS 30 (fighterHAS): 0.28 avg hits, 0.5% destruction rate
HAS 31 (fighterHAS): 0.27 avg hits, 0.3% destruction rate
HAS 32 (fighterHAS): 0.27 avg hits, 0.4% destruction rate
HAS 33 (fighterHAS): 0.28 avg hits, 0.3% destruction rate
HAS 34 (fighterHAS): 0.28 avg hits, 0.4% destruction rate

EXPOSED AIRCRAFT ANALYSIS:
EXPOSED 1 (fighterEXP): 0.09 avg hits [P-800: 0.06, Kh-101: 0.01, Kalibr: 0.02, Iskander-K: 0.01, OTR-21: 0.00], 31.6% destruction rate, 9.2% direct hits, 22.5% splash kills
EXPOSED 2 (fighterEXP): 0.09 avg hits [P-800: 0.06, Kalibr: 0.01, Kh-101: 0.01, Iskander-K: 0.01, OTR-21: 0.00], 33.6% destruction rate, 9.0% direct hits, 24.5% splash kills
EXPOSED 3 (fighterEXP): 0.08 avg hits [P-800: 0.05, Kalibr: 0.02, Iskander-K: 0.01, Kh-101: 0.01, OTR-21: 0.00], 40.0% destruction rate, 8.4% direct hits, 31.6% splash kills
EXPOSED 4 (fighterEXP): 0.09 avg hits [Kalibr: 0.02, P-800: 0.06, Iskander-K: 0.01, Kh-101: 0.01, OTR-21: 0.00], 37.9% destruction rate, 8.9% direct hits, 29.1% splash kills
EXPOSED 5 (fighterEXP): 0.10 avg hits [Kalibr: 0.02, P-800: 0.06, Kh-101: 0.01, Iskander-K: 0.01, OTR-21: 0.00], 26.1% destruction rate, 9.7% direct hits, 16.4% splash kills

EXPOSED 6 (awacs): 0.32 avg hits [Iskander-K: 0.17, Kalibr: 0.06, P-800: 0.05, Kh-101: 0.03, OTR-21: 0.01], 84.0% destruction rate, 31.5% direct hits, 52.4% splash kills
EXPOSED 7 (tanker): 0.20 avg hits [Iskander-K: 0.13, Kalibr: 0.03, P-800: 0.02, Kh-101: 0.01, OTR-21: 0.01], 86.8% destruction rate, 19.5% direct hits, 67.2% splash kills
EXPOSED 8 (tanker): 0.20 avg hits [Iskander-K: 0.14, Kh-101: 0.01, Kalibr: 0.03, P-800: 0.02, OTR-21: 0.01], 86.0% destruction rate, 20.1% direct hits, 65.9% splash kills
EXPOSED 9 (tanker): 0.23 avg hits [Iskander-K: 0.14, Kh-101: 0.02, OTR-21: 0.01, Kalibr: 0.03, P-800: 0.03], 76.0% destruction rate, 23.3% direct hits, 52.7% splash kills
EXPOSED 10 (fighterEXP): 0.05 avg hits [Kh-101: 0.00, P-800: 0.03, Iskander-K: 0.00, Kalibr: 0.01, OTR-21: 0.00], 68.5% destruction rate, 4.6% direct hits, 63.9% splash kills
EXPOSED 11 (fighterEXP): 0.09 avg hits [Kh-101: 0.01, P-800: 0.06, Iskander-K: 0.01, Kalibr: 0.01, OTR-21: 0.00], 35.0% destruction rate, 8.9% direct hits, 26.1% splash kills
EXPOSED 12 (fighterEXP): 0.10 avg hits [Kalibr: 0.02, P-800: 0.06, Iskander-K: 0.01, Kh-101: 0.01, OTR-21: 0.00], 20.3% destruction rate, 9.8% direct hits, 10.5% splash kills
EXPOSED 13 (fighterEXP): 0.09 avg hits [P-800: 0.06, Iskander-K: 0.01, Kh-101: 0.01, Kalibr: 0.02, OTR-21: 0.00], 28.5% destruction rate, 9.4% direct hits, 19.1% splash kills
EXPOSED 14 (fighterEXP): 0.10 avg hits [P-800: 0.06, Kh-101: 0.01, Kalibr: 0.02, Iskander-K: 0.01, OTR-21: 0.00], 28.5% destruction rate, 9.6% direct hits, 18.9% splash kills
EXPOSED 15 (fighterEXP): 0.09 avg hits [P-800: 0.06, Iskander-K: 0.00, Kalibr: 0.02, Kh-101: 0.01, OTR-21: 0.00], 28.5% destruction rate, 9.5% direct hits, 19.0% splash kills

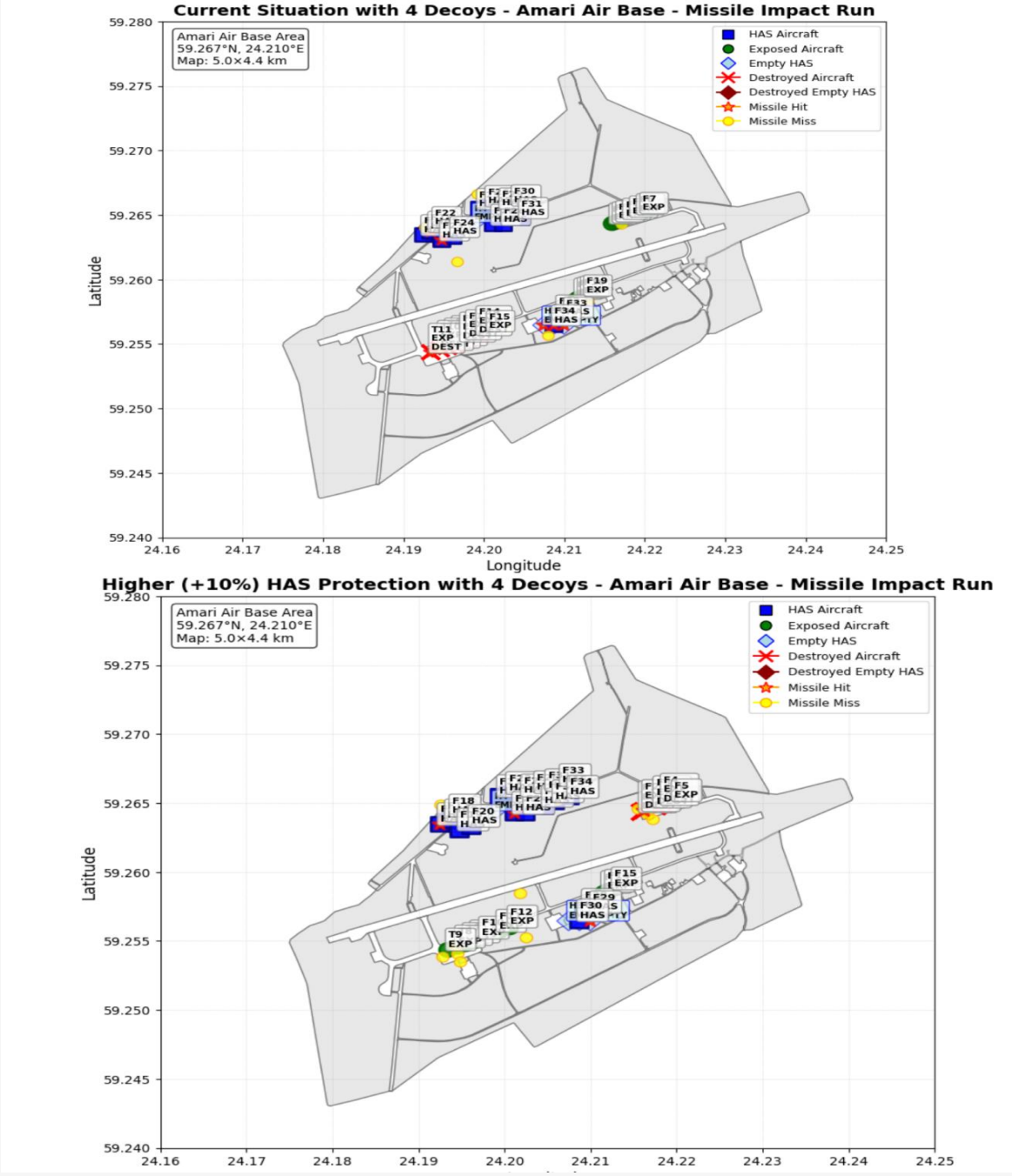
EXPOSED SUMMARY:
Average destruction rate: 47.4%
Average hits per exposed aircraft: 0.13
Total direct hit kills: 1.91
Total splash kills: 5.20

EMPTY HAS ANALYSIS:
EMPTY HAS 1: 0.16 avg hits [Kh-101: 0.04, Kalibr: 0.03, P-800: 0.04, Iskander-K: 0.05, OTR-21: 0.00], 0.5% destruction rate
EMPTY HAS 2: 0.16 avg hits [Kalibr: 0.03, P-800: 0.04, Iskander-K: 0.05, Kh-101: 0.03, OTR-21: 0.00], 0.7% destruction rate
EMPTY HAS 3: 0.16 avg hits [P-800: 0.04, Kh-101: 0.04, Kalibr: 0.03, Iskander-K: 0.06, OTR-21: 0.00], 0.6% destruction rate
EMPTY HAS 4: 0.16 avg hits [P-800: 0.04, Kh-101: 0.03, Iskander-K: 0.06, Kalibr: 0.03, OTR-21: 0.00], 0.6% destruction rate

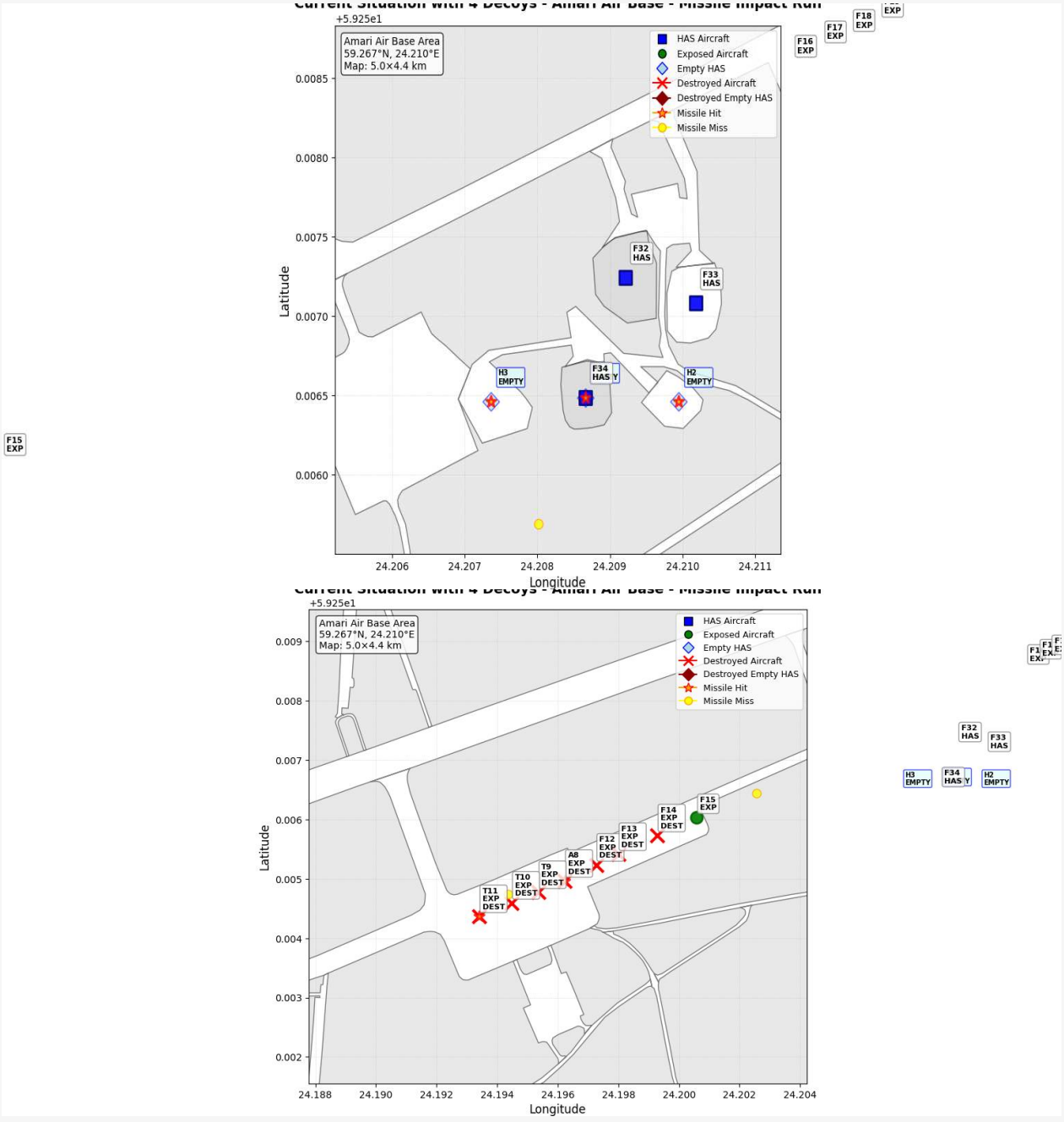
EMPTY HAS SUMMARY:
Average destruction rate: 0.6%
Average hits per empty HAS: 0.16
Total empty HAS destroyed: 0.02
Deception effectiveness: 99.4% survival rate

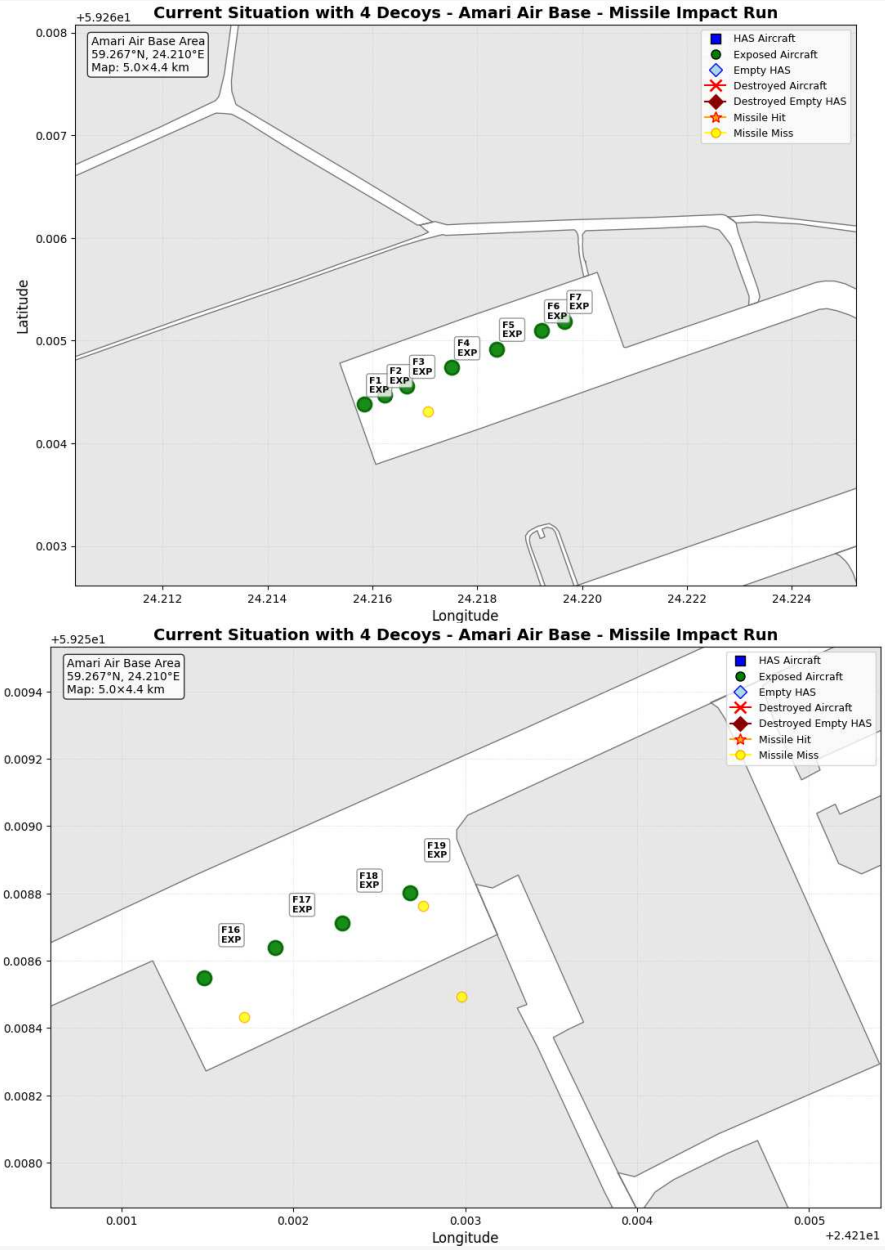
A.3

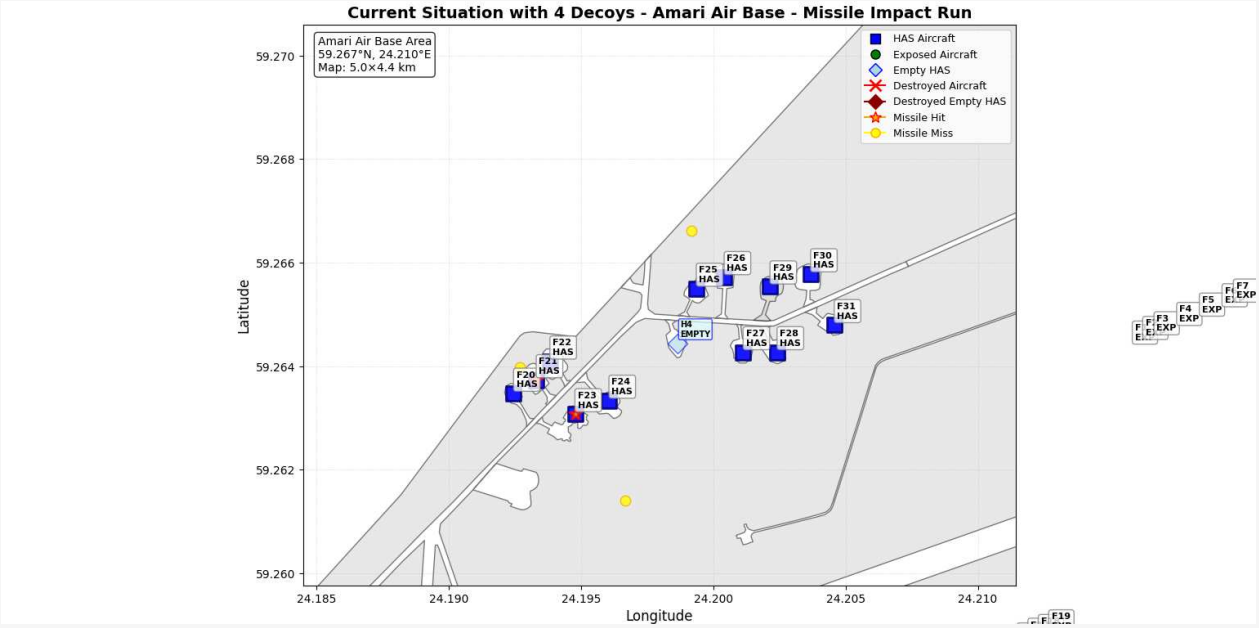
Outcome Visualization Comparisons Scenarios 1



Scenario 1 Visualizations

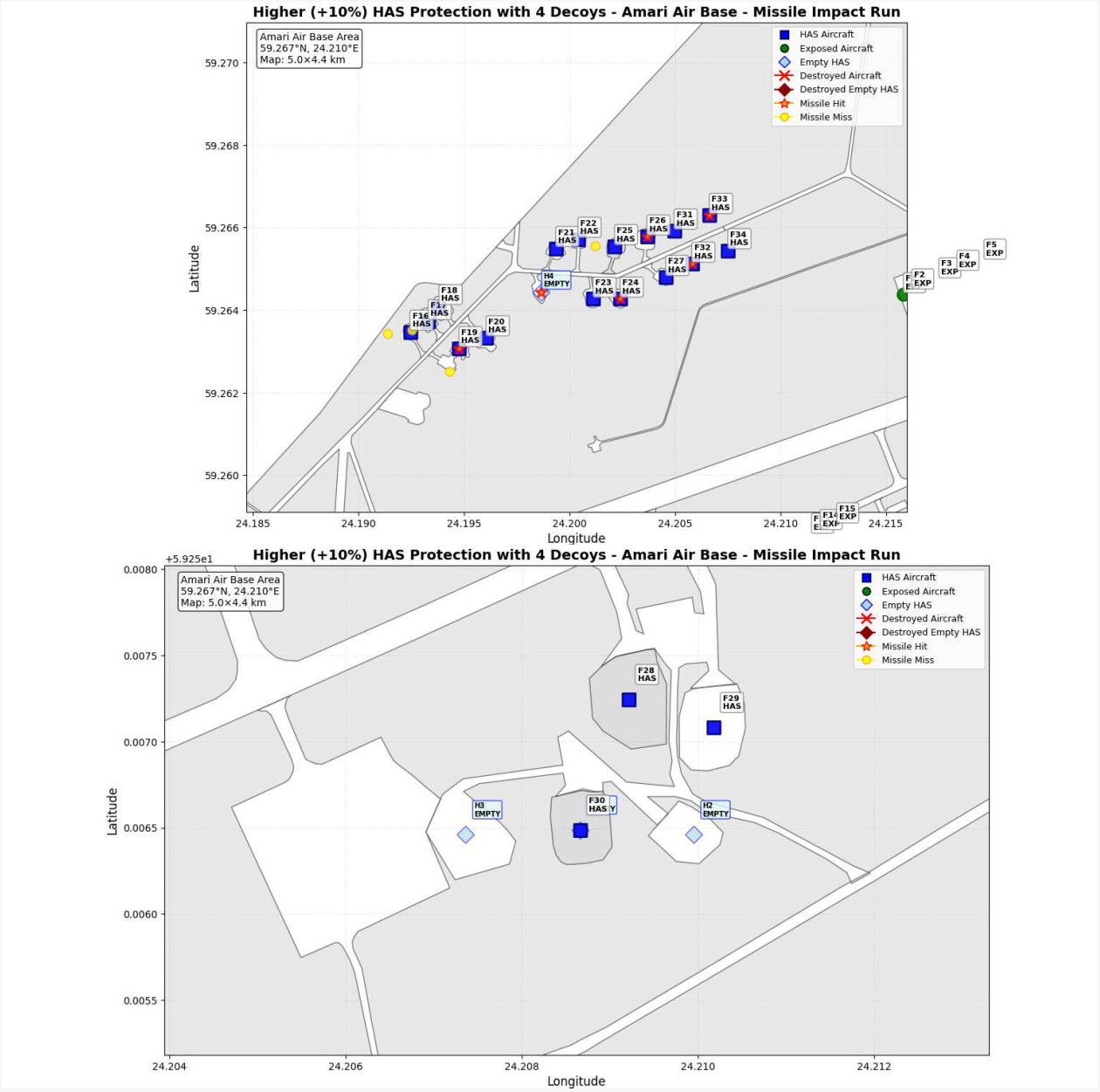


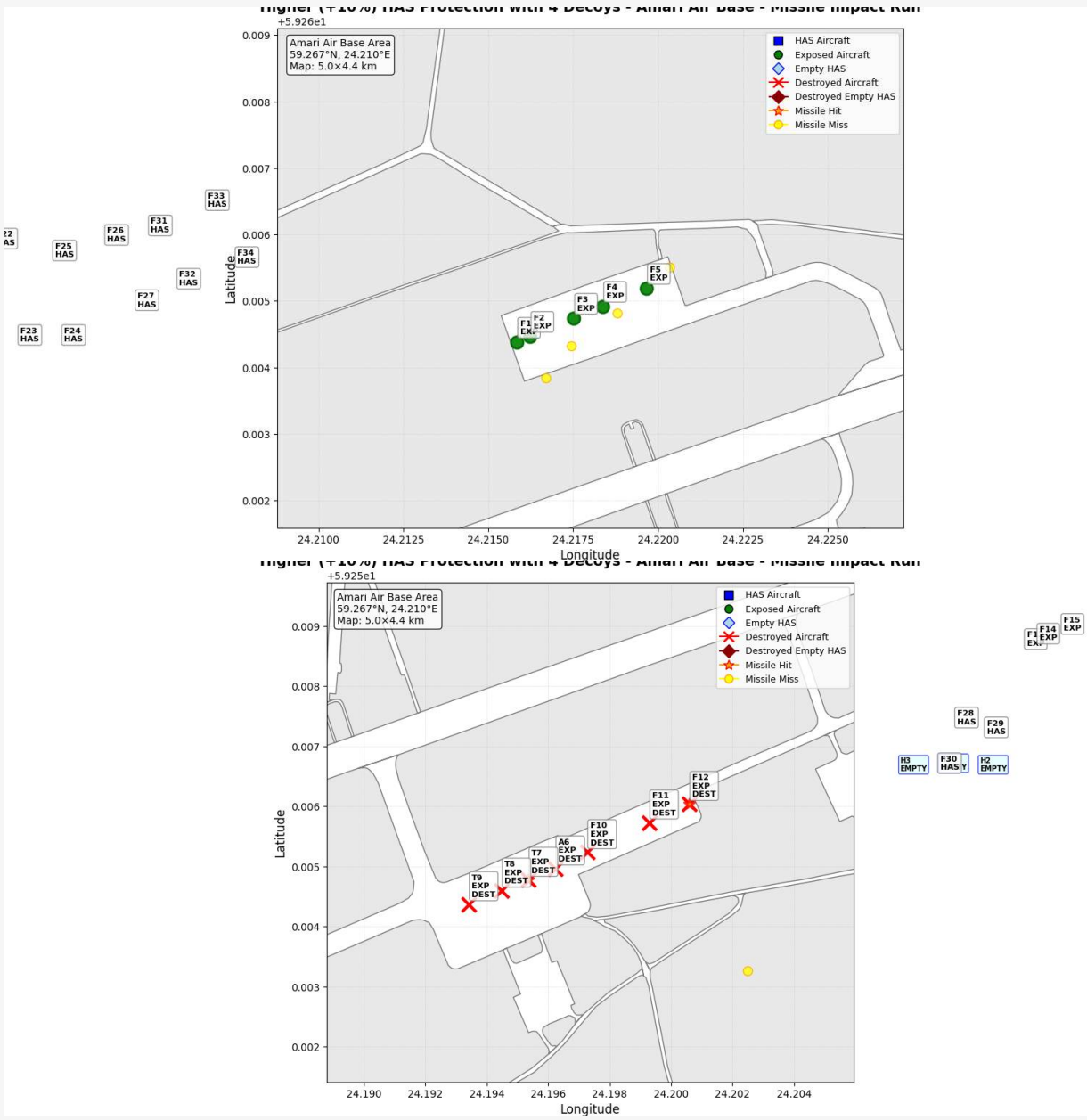

Current Situation with 4 Decoys - Amari Air Base - Missile Impact Run
+5.926e1
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
F1 EXP
F2 EXP
F3 EXP
F4 EXP
F5 EXP
F6 EXP
F7 EXP
Latitude
Longitude
0.008
0.007
0.006
0.005
0.004
0.003
24.212
24.214
24.216
24.218
24.220
24.222
24.224
Current Situation with 4 Decoys - Amari Air Base - Missile Impact Run
+5.925e1
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
F16 EXP
F17 EXP
F18 EXP
F19 EXP
Latitude
Longitude
0.0094
0.0092
0.0090
0.0088
0.0086
0.0084
0.0082
0.0080
0.001
0.002
0.003
0.004
0.005
+2.421e1


Current Situation with 4 Decoys - Amari Air Base - Missile Impact Run
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
Latitude
Longitude
59.270
59.268
59.266
59.264
59.262
59.260
24.185
24.190
24.195
24.200
24.205
24.210
F20 HAS
F21 HAS
F22 HAS
F23 HAS
F24 HAS
F25 HAS
F26 HAS
F27 HAS
F28 HAS
F29 HAS
F30 HAS
F31 HAS
H6 EMPTY
F3 EXP
F4 EXP
F5 EXP
F7 EXP

A.4

Scenario 2 Outcome Visualizations




Higher (+10%) HAS Protection with 4 Decoys - Amari Air Base - Missile Impact Run
+5.926e1
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
F1 EXP
F2 EXP
F3 EXP
F4 EXP
F5 EXP
F22 HAS
F23 HAS
F24 HAS
F25 HAS
F26 HAS
F27 HAS
F31 HAS
F32 HAS
F33 HAS
F34 HAS
Latitude
Longitude
Higher (+10%) HAS Protection with 4 Decoys - Amari Air Base - Missile Impact Run
+5.925e1
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
F10 EXP DEST
F11 EXP DEST
F12 EXP DEST
A6 EXP DEST
T7 EXP DEST
T8 EXP DEST
T9 EXP DEST
F13 EXP
F14 EXP
F15 EXP
F28 HAS
F29 HAS
F30 HAS
H2 EMPTY
H3 EMPTY
Latitude
Longitude


Higher (+10%) HAS Protection with 4 Decoys - Amari Air Base - Missile Impact Run
+5.925e1
Amari Air Base Area
59.267°N, 24.210°E
Map: 5.0×4.4 km
HAS Aircraft
Exposed Aircraft
Empty HAS
Destroyed Aircraft
Destroyed Empty HAS
Missile Hit
Missile Miss
F15 EXP
F14 EXP
F13 EXP
Latitude
0.00950
0.00925
0.00900
0.00875
0.00850
0.00825
0.00800
0.00775
0.000
0.001
0.002
0.003
0.004
0.005
Longitude
+2.421e1

A.5

Python Code

```
import random
import math
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import geopandas as gpd

from collections import defaultdict
from matplotlib.lines import Line2D
from shapely.geometry import Point
from typing import Dict, List, Any, Optional, Tuple

# --- Configuration Parameters ---
CONFIG = {
    'targeting': {
        'exposed_preference_probability': 0.7,
        'has_priority_for_heavy_missiles': True
    },
    'has_damage': {
        'hits_for_destruction_chance': 3,
        'destruction_probability_after_hits': 0.6,
        'adjacent_has_damage_chance': 0.3
    },
    'splash_damage': {
        'use_cube_root_scaling': True,
        'lethal_radius_factor': 25,      # Multiplier for cube root (high damage zone)
        'damage_radius_factor': 25,      # Multiplier for cube root (moderate damage zone)
        'fragment_radius_factor': 40,    # Multiplier for cube root (fragment zone)

        # Legacy parameters
        'exposed_destruction_chance': 0.5,
        'blast_radius_coefficient': 1.0,
        'adjacent_has_range_m': 50,

        # Target vulnerability factors
        'vulnerability_factors': {
            'EXPOSED': 2.0,    # Full vulnerability
            'HAS': 0.25,       # Reduced vulnerability in hardened shelters
            'EMPTY_HAS': 0.25
        },

        # Distance-based damage probability
        'damage_decay_model': 'exponential',  # 'linear' or 'exponential'
        'decay_constant': 2.0
    },
    'interception': {
        'heavy_missile_priority': True,
        'random_intercept_chance': 0.41
    },
    'visualization': {
        'figure_size': (12, 10)}
}

# --- Missile and Aircraft Data Definitions ---
missile_types = {
    'Iskander-M': {'cep_m': 50, 'warhead_kg': 700, 'value_weight': 3, 'warhead_type': 'HE'},
    'Iskander-K': {'cep_m': 50, 'warhead_kg': 480, 'value_weight': 2, 'warhead_type': 'HE'},
```

```
    'P-800': {'cep_m': 1.5, 'warhead_kg': 300, 'value_weight': 1, 'warhead_type': 'HE'},
    'Kalibr': {'cep_m': 12, 'warhead_kg': 400, 'value_weight': 2, 'warhead_type': 'HE'},
    'OTR-21': {'cep_m': 150, 'warhead_kg': 482, 'value_weight': 2, 'warhead_type': 'HE'},
    'Kh-101': {'cep_m': 15, 'warhead_kg': 450, 'value_weight': 2, 'warhead_type': 'HE'},
}

aircraft_types = {
    'fighterEXP': {'radius_m': 5.35, 'has_resistance_kg': 0, 'target_value': 2},
    'fighterHAS': {'radius_m': 30, 'has_resistance_kg': 500, 'target_value': 1},
    'tanker': {'radius_m': 30, 'has_resistance_kg': 500, 'target_value': 3},
    'awacs': {'radius_m': 22, 'has_resistance_kg': 500, 'target_value': 6}
}

def calculate_enhanced_blast_radius(warhead_kg, warhead_type='HE'):
    """Calculate blast radius using cube root scaling for realistic physics."""

    # TNT equivalent factors for different warhead types
    tnt_factors = {
        'HE': 1.0,      # High Explosive (standard)
        'HEAT': 0.8,    # High Explosive Anti-Tank
        'FAE': 1.3,     # Fuel-Air Explosive (more blast effect)
        'FRAG': 0.7     # Fragmentation (less blast, more fragments)
    }

    tnt_equivalent = warhead_kg * tnt_factors.get(warhead_type, 1.0)

    if CONFIG['splash_damage']['use_cube_root_scaling']:
        # Cube root scaling (realistic physics)
        cube_root_weight = tnt_equivalent ** (1/3)

        lethal_radius = CONFIG['splash_damage']['lethal_radius_factor'] * cube_root_weight
        damage_radius = CONFIG['splash_damage']['damage_radius_factor'] * cube_root_weight
        fragment_radius = CONFIG['splash_damage']['fragment_radius_factor'] * cube_root_weight
    else:
        # Legacy square root scaling
        sqrt_weight = (tnt_equivalent ** 0.5)
        lethal_radius = sqrt_weight * CONFIG['splash_damage']['blast_radius_coefficient']
        damage_radius = lethal_radius * 1.5
        fragment_radius = lethal_radius * 2.0

    return {
        'lethal': lethal_radius,
        'damage': damage_radius,
        'fragment': fragment_radius
    }



# --- Input Validation ---
def validate_inputs(missiles: List[Dict], aircraft: List[Dict], empty_has: List[Dict] = None) ->
bool:
    """Validate input parameters for the simulation."""
    try:
        # Validate missiles
        for missile in missiles:
            if 'type' not in missile or missile['type'] not in missile_types:
                raise ValueError(f"Invalid missile type: {missile.get('type', 'Unknown')}")

        # Validate aircraft
        for i, aircraft_obj in enumerate(aircraft):
            if 'type' not in aircraft_obj or aircraft_obj['type'] not in aircraft_types:
                raise ValueError(f"Invalid aircraft type: {aircraft_obj.get('type', 'Unknown')}")
```

```
            # Validate coordinates if provided
            if 'x' in aircraft_obj and not isinstance(aircraft_obj['x'], (int, float)):
                raise ValueError(f"Aircraft {i} x coordinate must be numeric, got:
{aircraft_obj['x']}")
            if 'y' in aircraft_obj and not isinstance(aircraft_obj['y'], (int, float)):
                raise ValueError(f"Aircraft {i} y coordinate must be numeric, got:
{aircraft_obj['y']}")

            # Validate shelter assignment
            if 'shelter' not in aircraft_obj:
                raise ValueError(f"Aircraft {i} must have 'shelter' field specified ('HAS' or
'EXPOSED')")
            if aircraft_obj['shelter'] not in ['HAS', 'EXPOSED']:
                raise ValueError(f"Aircraft {i} shelter must be 'HAS' or 'EXPOSED', got:
{aircraft_obj['shelter']}")

        # Validate empty HAS
        if empty_has:
            for i, has_obj in enumerate(empty_has):
                if 'x' not in has_obj or 'y' not in has_obj:
                    raise ValueError(f"Empty HAS {i} must have 'x' and 'y' coordinates")
                if not isinstance(has_obj['x'], (int, float)) or not isinstance(has_obj['y'],
(int, float)):
                    raise ValueError(f"Empty HAS {i} coordinates must be numeric")

        return True
    except Exception as e:
        print(f"Input validation error: {e}")
        return False

def calculate_damage_probability(distance, blast_radii, target_shelter, target_type='aircraft'):
    """Calculate damage probability based on distance from blast center."""

    # Get vulnerability factor based on shelter type
    vuln_key = target_shelter if target_type == 'aircraft' else 'EMPTY_HAS'
    vulnerability = CONFIG['splash_damage']['vulnerability_factors'].get(vuln_key, 1.0)

    # Calculate base probability based on distance and blast zones
    if distance <= blast_radii['lethal']:
        base_prob = 0.95  # Very high probability in lethal zone
    elif distance <= blast_radii['damage']:
        # Linear decay in damage zone
        zone_progress = (distance - blast_radii['lethal']) / (blast_radii['damage'] -
blast_radii['lethal'])
        base_prob = 0.95 - (0.65 * zone_progress)  # 0.95 to 0.30
    elif distance <= blast_radii['fragment']:
        # Linear decay in fragment zone
        zone_progress = (distance - blast_radii['damage']) / (blast_radii['fragment'] -
blast_radii['damage'])
        base_prob = 0.30 - (0.25 * zone_progress)  # 0.30 to 0.05
    else:
        base_prob = 0.0  # No damage beyond fragment radius

    # Apply decay model
    if CONFIG['splash_damage']['damage_decay_model'] == 'exponential' and distance > 0:
        decay_factor = math.exp(-CONFIG['splash_damage']['decay_constant'] * distance /
blast_radii['fragment'])
        base_prob *= decay_factor

    # Apply vulnerability factor
    final_prob = base_prob * vulnerability

    return max(0.0, min(1.0, final_prob))
```

```
# --- Plot Function ---
def plot_simulation(aircraft_status: List[Dict], missile_impacts: List[Dict],
                    empty_has_status: List[Dict] = None,
                    title: str = "Missile Impact Visualization",
                    shapefile_path: Optional[str] = None) -> None:
    """Plot the simulation results with aircraft positions, empty HAS, and missile impacts."""
    fig, ax = plt.subplots(figsize=CONFIG['visualization']['figure_size'])

    # Amari Air Base coordinates
    base_lon, base_lat = 24.21, 59.267
    map_bounds = {
        'lon_min': 24.16, 'lon_max': 24.25,
        'lat_min': 59.240, 'lat_max': 59.280
    }
    use_geographic = False

    # Load shapefile
    if shapefile_path is not None:
        try:
            gdf = gpd.read_file(shapefile_path)
            print(f"Original CRS: {gdf.crs}")
            gdf = gdf.to_crs(epsg=4326)  # Convert to WGS84 (lat/lon)
            gdf.plot(ax=ax, color='lightgray', edgecolor='black', alpha=0.5, zorder=-1)
            print(f"Shapefile loaded - CRS: {gdf.crs}, Bounds: {gdf.total_bounds}")
            use_geographic = True

            # Set plot bounds to the full map area
            ax.set_xlim(map_bounds['lon_min'], map_bounds['lon_max'])
            ax.set_ylim(map_bounds['lat_min'], map_bounds['lat_max'])

        except Exception as e:
            print(f"Could not load shapefile: {e}")

    if not use_geographic:
        pass

    # Plot empty HAS
    if empty_has_status:
        for i, has in enumerate(empty_has_status):
            if use_geographic:
                # Convert meter offsets to degree offsets within map bounds
                x = base_lon + (has['x'] / 56000)
                y = base_lat + (has['y'] / 111000)

                # Ensure coordinates stay within map bounds
                x = max(map_bounds['lon_min'], min(map_bounds['lon_max'], x))
                y = max(map_bounds['lat_min'], min(map_bounds['lat_max'], y))
            else:
                x, y = has['x'], has['y']

            if has['destroyed']:
                color = 'darkred'
                marker = 'D'  # Diamond for destroyed empty HAS
                size = 180
                alpha = 0.8
                edge_color = 'black'
                edge_width = 2
            else:
                color = 'lightblue'
                marker = 'D'  # Diamond for empty HAS
                size = 150
```

```
                alpha = 0.6
                edge_color = 'blue'
                edge_width = 1

            # Plot empty HAS
            ax.scatter(x, y, c=color, marker=marker, s=size,
                       edgecolors=edge_color, linewidth=edge_width, zorder=1, alpha=alpha)

            # Add empty HAS labels
            has_label = f"H{i+1}"
            destroyed_status = "DEST" if has['destroyed'] else "EMPTY"
            full_label = f"{has_label}\n{destroyed_status}"

            # Position label slightly offset from HAS
            label_offset_x = 0.0001 if use_geographic else 30
            label_offset_y = 0.0001 if use_geographic else 30

            ax.text(x + label_offset_x, y + label_offset_y, full_label,
                    fontsize=7, fontweight='bold', ha='left', va='bottom',
                    bbox=dict(boxstyle='round,pad=0.2', facecolor='lightcyan', alpha=0.7,
edgecolor='blue'),
                    zorder=4)

    # Plot aircraft with labels
    for i, ac in enumerate(aircraft_status):
        if use_geographic:
            # Convert meter offsets to degree offsets within map bounds
            x = base_lon + (ac['x'] / 56000)
            y = base_lat + (ac['y'] / 111000)

            # Ensure coordinates stay within map bounds
            x = max(map_bounds['lon_min'], min(map_bounds['lon_max'], x))
            y = max(map_bounds['lat_min'], min(map_bounds['lat_max'], y))
        else:
            x, y = ac['x'], ac['y']

        if ac['destroyed']:
            color = 'red'
            marker = 'x'
            size = 200
            alpha = 1.0
            edge_color = 'darkred'
            edge_width = 3
        elif ac['shelter'] == 'HAS':
            color = 'blue'
            marker = 's'  # Square for HAS
            size = 150
            alpha = 0.9
            edge_color = 'navy'
            edge_width = 2
        else:
            color = 'green'
            marker = 'o'
            size = 150
            alpha = 0.9
            edge_color = 'darkgreen'
            edge_width = 2

        # Plot aircraft
        ax.scatter(x, y, c=color, marker=marker, s=size,
                   edgecolors=edge_color, linewidth=edge_width, zorder=2, alpha=alpha)

        # Add aircraft labels showing type and ID
```

```
        aircraft_label = f"{ac['type'][0].upper()}{i+1}"  # F1, T1, A1, etc.
        shelter_status = "HAS" if ac['shelter'] == 'HAS' else "EXP"
        destroyed_status = "DEST" if ac['destroyed'] else ""

        # Create label with aircraft info
        full_label = f"{aircraft_label}\n{shelter_status}"
        if destroyed_status:
            full_label += f"\n{destroyed_status}"

        # Position label slightly offset from aircraft
        label_offset_x = 0.0001 if use_geographic else 50
        label_offset_y = 0.0001 if use_geographic else 50

        ax.text(x + label_offset_x, y + label_offset_y, full_label,
                fontsize=8, fontweight='bold', ha='left', va='bottom',
                bbox=dict(boxstyle='round,pad=0.3', facecolor='white', alpha=0.8,
edgecolor='gray'),
                zorder=4)

    # Plot missile impacts
    for impact in missile_impacts:
        if use_geographic:
            # Convert meter offsets to degree offsets within map bounds
            x = base_lon + (impact['x'] / 56000)
            y = base_lat + (impact['y'] / 111000)

            # Ensure coordinates stay within map bounds
            x = max(map_bounds['lon_min'], min(map_bounds['lon_max'], x))
            y = max(map_bounds['lat_min'], min(map_bounds['lat_max'], y))
        else:
            x, y = impact['x'], impact['y']

        hit = impact['hit']
        if hit:
            color = 'orange'
            marker = '*'
            size = 120
            edge_color = 'red'
            edge_width = 2
        else:
            color = 'yellow'
            marker = 'o'  # Changed from '.' to 'o' for better visibility
            size = 80     # Increased size for misses
            edge_color = 'orange'
            edge_width = 1

        ax.scatter(x, y, c=color, marker=marker, s=size,
                   edgecolors=edge_color, linewidth=edge_width, zorder=3, alpha=0.8)

    # Create legend
    legend_elements = [
        Line2D([0], [0], marker='s', color='w', label='HAS Aircraft',
               markerfacecolor='blue', markeredgecolor='black', markersize=8),
        Line2D([0], [0], marker='o', color='w', label='Exposed Aircraft',
               markerfacecolor='green', markeredgecolor='black', markersize=8),
        Line2D([0], [0], marker='D', color='w', label='Empty HAS',
               markerfacecolor='lightblue', markeredgecolor='blue', markersize=8),
        Line2D([0], [0], marker='x', color='red', label='Destroyed Aircraft',
               markersize=10, markeredgewidth=2),
        Line2D([0], [0], marker='D', color='darkred', label='Destroyed Empty HAS',
               markersize=8, markeredgewidth=2),
        Line2D([0], [0], marker='*', color='orange', label='Missile Hit',
               markerfacecolor='orange', markeredgecolor='red', markersize=10),
```

```
        Line2D([0], [0], marker='o', color='yellow', label='Missile Miss',
               markerfacecolor='yellow', markeredgecolor='orange', markersize=8),
    ]
    ax.legend(handles=legend_elements, loc='upper right', fontsize=9)
    ax.set_title(title, fontsize=14, fontweight='bold')

    if use_geographic:
        ax.set_xlabel("Longitude", fontsize=12)
        ax.set_ylabel("Latitude", fontsize=12)
        ax.grid(True, linestyle=':', linewidth=0.5, alpha=0.7)
        # Add coordinates and map info
        map_width_km = (map_bounds['lon_max'] - map_bounds['lon_min']) * 56  # Approximate km
        map_height_km = (map_bounds['lat_max'] - map_bounds['lat_min']) * 111  # Approximate km
        ax.text(0.02, 0.98, f"Amari Air Base Area\n{base_lat:.3f}°N, {base_lon:.3f}°E\nMap:
{map_width_km:.1f}×{map_height_km:.1f} km",
                transform=ax.transAxes, fontsize=10, verticalalignment='top',
                bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))
    else:
        ax.set_xlabel("X Coordinate (meters)", fontsize=12)
        ax.set_ylabel("Y Coordinate (meters)", fontsize=12)
        ax.set_xlim(-5000, 5000)
        ax.set_ylim(-3000, 3000)
        ax.axhline(0, color='gray', linestyle='--', linewidth=0.5, alpha=0.7)
        ax.axvline(0, color='gray', linestyle='--', linewidth=0.5, alpha=0.7)
        ax.grid(True, linestyle=':', linewidth=0.5, alpha=0.7)
        ax.set_aspect('equal')

    plt.tight_layout()
    plt.show()

# --- Target Selection ---
def compute_hit_probability(target_radius_m: float, cep_m: float, k: float = 1.774) -> float:
    """Calculate hit probability based on target radius and missile CEP."""
    if cep_m <= 0 or target_radius_m <= 0:
        return 0.0
    return 1 - math.exp(-k * (target_radius_m ** 2) / (cep_m ** 2))

def choose_target(exposed_targets: List[int], has_targets: List[int], empty_has_targets:
List[int],
                  aircraft_status: List[Dict], empty_has_status: List[Dict], missile_type: str) ->
Tuple[Optional[int], str]:
    """Choose target based on missile type and targeting doctrine. Returns (target_index,
target_type)."""
    missile = missile_types[missile_type]

    if (CONFIG['targeting']['has_priority_for_heavy_missiles'] and
        missile['warhead_kg'] > 500):
        if has_targets:
            return random.choice(has_targets), 'aircraft'
        elif empty_has_targets:
            return random.choice(empty_has_targets), 'empty_has'

    # Weight targets by value and availability
    all_targets = []

    # Add exposed targets with preference
    for idx in exposed_targets:
        ac_value = aircraft_types[aircraft_status[idx]['type']]['target_value']
        weight = ac_value * CONFIG['targeting']['exposed_preference_probability']
        all_targets.extend([('aircraft', idx)] * int(weight * 10))

    # Add occupied HAS targets
    for idx in has_targets:
```

```
        ac_value = aircraft_types[aircraft_status[idx]['type']]['target_value']
        weight = ac_value * (1 - CONFIG['targeting']['exposed_preference_probability'])
        all_targets.extend([('aircraft', idx)] * int(weight * 10))

    # Add empty HAS targets with lower weight (deception value)
    for idx in empty_has_targets:
        weight = 0.7  # Lower value for empty HAS
        all_targets.extend([('empty_has', idx)] * int(weight * 10))

    if all_targets:
        target_type, target_idx = random.choice(all_targets)
        return target_idx, target_type
    else:
        return None, 'none'

def apply_splash_damage(target_index, aircraft_status, missile_type, destroyed_summary,
                        empty_has_status=None, target_type='aircraft'):
    """Apply enhanced splash damage using cube root scaling and distance-based probability."""

    missile = missile_types[missile_type]

    # Get target coordinates based on type
    if target_type == 'aircraft':
        target_obj = aircraft_status[target_index]
    else:  # empty_has
        target_obj = empty_has_status[target_index] if empty_has_status else None
        if not target_obj:
            return

    # Calculate enhanced blast radii
    blast_radii = calculate_enhanced_blast_radius(
        missile['warhead_kg'],
        missile.get('warhead_type', 'HE')
    )

    #print(f"DEBUG: {missile_type} blast radii - Lethal: {blast_radii['lethal']:.1f}m, "
          #f"Damage: {blast_radii['damage']:.1f}m, Fragment: {blast_radii['fragment']:.1f}m")

    # Apply splash damage to aircraft
    for i, other_ac in enumerate(aircraft_status):
        if i == target_index and target_type == 'aircraft':
            continue  # Skip the primary target
        if other_ac['destroyed']:
            continue  # Skip already destroyed aircraft

        # Calculate distance
        dx_m = other_ac['x'] - target_obj['x']
        dy_m = other_ac['y'] - target_obj['y']
        distance = math.hypot(dx_m, dy_m)

        # Skip if beyond maximum blast radius
        if distance > blast_radii['fragment']:
            continue

        # Calculate damage probability
        damage_prob = calculate_damage_probability(
            distance, blast_radii, other_ac['shelter'], 'aircraft'
        )

        if damage_prob > 0 and random.random() < damage_prob:
            other_ac['destroyed'] = True
            other_ac['splash_kill'] = True
            destroyed_summary[other_ac['shelter']] += 1
```

```
            destroyed_summary['by_missile_type'][missile_type] += 1
            destroyed_summary['by_aircraft_type'][other_ac['type']] += 1
            destroyed_summary['splash_kills'] += 1

            #print(f"DEBUG: Splash kill - Aircraft {i} ({other_ac['type']}) at {distance:.1f}m "
                  #f"with {damage_prob:.2%} probability")

    # Apply splash damage to empty HAS if provided
    if empty_has_status:
        for i, other_has in enumerate(empty_has_status):
            if i == target_index and target_type == 'empty_has':
                continue  # Skip the primary target
            if other_has.get('destroyed', False):
                continue  # Skip already destroyed HAS

            # Calculate distance
            dx_m = other_has['x'] - target_obj['x']
            dy_m = other_has['y'] - target_obj['y']
            distance = math.hypot(dx_m, dy_m)

            # Skip if beyond maximum blast radius
            if distance > blast_radii['fragment']:
                continue

            # Calculate damage probability for empty HAS
            damage_prob = calculate_damage_probability(
                distance, blast_radii, 'EMPTY_HAS', 'empty_has'
            )

            if damage_prob > 0 and random.random() < damage_prob:
                other_has['destroyed'] = True
                destroyed_summary['EMPTY_HAS'] += 1
                destroyed_summary['by_missile_type'][missile_type] += 1

                #print(f"DEBUG: Splash damage - Empty HAS {i} at {distance:.1f}m "
                      #f"with {damage_prob:.2%} probability")

# --- Core Simulation ---
def run_simulation(missiles: List[Dict], aircraft: List[Dict], empty_has: List[Dict] = None,
                   patriot_launchers: int = 6, missiles_per_launcher: int = 4,
                   intercept_rate: float = 0.41, waves: int = 6) -> Dict:
    """Run a single simulation of missile strikes against aircraft and empty HAS."""

    if not validate_inputs(missiles, aircraft, empty_has):
        raise ValueError("Invalid input parameters")

    impact_log = []
    aircraft_status = []
    empty_has_status = []

    # Initialize aircraft status
    for i, ac in enumerate(aircraft):
        shelter = ac['shelter']
        x = float(ac.get('x', (i % 5) * 800 - 1600))
        y = float(ac.get('y', (i // 5) * 800 - 1600))

        aircraft_status.append({
            'type': ac['type'],
            'shelter': shelter,
            'destroyed': False,
            'x': x,
            'y': y,
            'hit_count': 0,
```

```
            'hit_log': defaultdict(int),
            'splash_kill': False
        })

    # Initialize empty HAS status
    if empty_has:
        for i, has in enumerate(empty_has):
            empty_has_status.append({
                'destroyed': False,
                'x': float(has['x']),
                'y': float(has['y']),
                'hit_count': 0,
                'hit_log': defaultdict(int)
            })

    # Categorize targets
    has_targets = [i for i, ac in enumerate(aircraft_status) if ac['shelter'] == 'HAS']
    exposed_targets = [i for i, ac in enumerate(aircraft_status) if ac['shelter'] == 'EXPOSED']
    empty_has_targets = list(range(len(empty_has_status)))

    # Initialize results tracking
    destroyed_summary = {
        'HAS': 0,
        'EXPOSED': 0,
        'EMPTY_HAS': 0,
        'by_missile_type': defaultdict(int),
        'by_aircraft_type': defaultdict(int),
        'splash_kills': 0,
        'intercepted_by_type': defaultdict(int),
        'skipped_due_to_no_targets': defaultdict(int),
        'aircraft_status': aircraft_status,
        'empty_has_status': empty_has_status,
        'impact_log': []
    }

    # Process waves
    total_interceptors = patriot_launchers * missiles_per_launcher
    interceptors_per_wave = max(1, total_interceptors // waves)
    wave_size = max(1, len(missiles) // waves)

    for w in range(waves):
        start_idx = w * wave_size
        end_idx = min((w + 1) * wave_size, len(missiles))
        wave_missiles = missiles[start_idx:end_idx]
        remaining_interceptors = interceptors_per_wave

        # Improved interception: First pass to identify high-priority targets
        heavy_missiles = [m for m in wave_missiles if missile_types[m['type']]['warhead_kg'] >
500]
        other_missiles = [m for m in wave_missiles if missile_types[m['type']]['warhead_kg'] <=
499]

        # Process heavy missiles first with higher interception priority
        all_wave_missiles = []
        for missile_obj in heavy_missiles:
            all_wave_missiles.append((missile_obj, True))  # True = high priority
        for missile_obj in other_missiles:
            all_wave_missiles.append((missile_obj, False))  # False = normal priority

        for missile_obj, is_high_priority in all_wave_missiles:
            missile = missile_types[missile_obj['type']]

            # Skip if no valid targets
```

```
            available_has = [i for i in has_targets if not aircraft_status[i]['destroyed']]
            available_exposed = [i for i in exposed_targets if not
aircraft_status[i]['destroyed']]
            available_empty_has = [i for i in empty_has_targets if not
empty_has_status[i]['destroyed']]

            if not available_has and not available_exposed and not available_empty_has:
                destroyed_summary['skipped_due_to_no_targets'][missile_obj['type']] += 1
                continue

            # Interception logic
            intercepted = False
            if remaining_interceptors > 0:
                # Calculate interception probability based on priority and remaining interceptors
                base_intercept_prob = intercept_rate

                # Boost interception chance for high-priority targets
                if is_high_priority:
                    actual_intercept_prob = min(0.8, base_intercept_prob * 1.3)
                else:
                    actual_intercept_prob = base_intercept_prob

                # Further boost if Amari have plenty of interceptors remaining
                interceptor_availability_ratio = remaining_interceptors / max(1,
interceptors_per_wave)
                if interceptor_availability_ratio > 0.5:
                    actual_intercept_prob = min(0.9, actual_intercept_prob * 1.1)

                # Always attempt interception
                remaining_interceptors -= 1
                if random.random() < actual_intercept_prob:
                    destroyed_summary['intercepted_by_type'][missile_obj['type']] += 1
                    intercepted = True

            if intercepted:
                continue

            # Choose target
            target_index, target_type = choose_target(available_exposed, available_has,
available_empty_has,
                                                      aircraft_status, empty_has_status,
missile_obj['type'])
            if target_index is None:
                continue

            # Calculate hit probability and determine hit
            if target_type == 'aircraft':
                target_obj = aircraft_status[target_index]
                ac_type = aircraft_types[target_obj['type']]
                p_hit = compute_hit_probability(ac_type['radius_m'], missile['cep_m'])
            else:  # empty_has
                target_obj = empty_has_status[target_index]
                # Empty HAS have similar hit probability to small aircraft
                p_hit = compute_hit_probability(25.0, missile['cep_m'])  # 25m radius for HAS
structure

            hit = random.random() <= p_hit

            # Log impact
            impact_log.append({
                'missile_type': missile_obj['type'],
                'target_index': target_index,
                'target_type': target_type,
```

```
                'x': target_obj['x'],
                'y': target_obj['y'],
                'hit': hit
            })

            # Handle misses
            if not hit:
                miss_x = target_obj['x'] + random.gauss(0, missile['cep_m'])
                miss_y = target_obj['y'] + random.gauss(0, missile['cep_m'])
                impact_log[-1]['x'] = miss_x
                impact_log[-1]['y'] = miss_y
                continue

            # Damage processing for aircraft
            if target_type == 'aircraft':
                target_ac = aircraft_status[target_index]
                ac_type = aircraft_types[target_ac['type']]

                if target_ac['shelter'] == 'HAS' and not target_ac['destroyed']:
                    target_ac['hit_log'][missile_obj['type']] += 1

                    # HAS damage model
                    if missile['warhead_kg'] >= 800:
                        target_ac['destroyed'] = True
                    else:
                        # Progressive damage system
                        if missile['warhead_kg'] >= 600:
                            damage_points = 1.5  # Heavy damage
                        elif missile['warhead_kg'] >= 400:
                            damage_points = 1.0  # Medium damage
                        else:
                            damage_points = 0.5  # Light damage

                        target_ac['hit_count'] += damage_points

                        # Calculate destruction probability based on cumulative damage
                        if target_ac['hit_count'] >=
CONFIG['has_damage']['hits_for_destruction_chance']:
                            # Higher threshold for destruction chance
                            destruction_prob =
CONFIG['has_damage']['destruction_probability_after_hits']
                            # Reduce probability for well-protected HAS
                            if target_ac['hit_count'] < 4:  # Need significant damage
accumulation
                                destruction_prob *= 0.7

                            if random.random() < destruction_prob:
                                target_ac['destroyed'] = True

                    if target_ac['destroyed']:
                        destroyed_summary['HAS'] += 1
                        destroyed_summary['by_missile_type'][missile_obj['type']] += 1
                        destroyed_summary['by_aircraft_type'][target_ac['type']] += 1

                elif target_ac['shelter'] == 'EXPOSED' and not target_ac['destroyed']:
                    # FIRST: Track the direct hit before destroying the aircraft
                    target_ac['hit_log'][missile_obj['type']] += 1
                    target_ac['hit_count'] += 1

                    # THEN: Exposed aircraft destroyed immediately by direct hit
                    target_ac['destroyed'] = True
                    target_ac['splash_kill'] = False  # This is a direct hit, not splash
                    destroyed_summary['EXPOSED'] += 1
```

```
                    destroyed_summary['by_missile_type'][missile_obj['type']] += 1
                    destroyed_summary['by_aircraft_type'][target_ac['type']] += 1

                    # Apply splash damage to nearby aircraft
                    apply_splash_damage(target_index, aircraft_status, missile_obj['type'],
destroyed_summary, empty_has_status, 'aircraft')


            else:  # empty_has damage model
                target_has = empty_has_status[target_index]
                if not target_has['destroyed']:
                    target_has['hit_log'][missile_obj['type']] += 1
                    target_has['hit_count'] += 1

                    if missile['warhead_kg'] >= 600:
                        target_has['destroyed'] = True
                    elif target_has['hit_count'] >= 2:  # Require at least 2 hits for lighter
missiles
                        # Probability-based destruction for accumulated damage
                        destruction_prob = 0.8 if target_has['hit_count'] >= 3 else 0.5
                        if random.random() < destruction_prob:
                            target_has['destroyed'] = True

                    if target_has['destroyed']:
                        destroyed_summary['EMPTY_HAS'] += 1
                        destroyed_summary['by_missile_type'][missile_obj['type']] += 1

    destroyed_summary['impact_log'] = impact_log
    return destroyed_summary


# --- Results Display ---
def print_detailed_results(scenario_num: int, avg_results: Dict, scenario: Dict) -> None:
    """Print detailed results for a scenario including empty HAS."""
    print(f"\n{'='*50}")
    print(f"SCENARIO {scenario_num} RESULTS")
    print(f"{'='*50}")

    # Calculate ratios from manual assignments
    has_count = sum(1 for ac in scenario['aircraft'] if ac['shelter'] == 'HAS')
    exposed_count = sum(1 for ac in scenario['aircraft'] if ac['shelter'] == 'EXPOSED')
    empty_has_count = len(scenario.get('empty_has', []))
    total_aircraft = len(scenario['aircraft'])
    total_structures = total_aircraft + empty_has_count

    has_ratio = has_count / total_aircraft if total_aircraft > 0 else 0
    exposed_ratio = exposed_count / total_aircraft if total_aircraft > 0 else 0

    print(f"Manual HAS Assignment: {has_count}/{total_aircraft} ({has_ratio:.1%})")
    print(f"Manual EXPOSED Assignment: {exposed_count}/{total_aircraft} ({exposed_ratio:.1%})")
    print(f"Empty HAS Structures: {empty_has_count}")
    print(f"Total Aircraft: {total_aircraft}")
    print(f"Total Structures: {total_structures}")
    print(f"Total Missiles: {len(scenario['missiles'])}")
    print(f"Valid Simulation Runs: {avg_results['valid_runs']}")

    print(f"\nDESTRUCTION SUMMARY:")
    total_destroyed = avg_results.get('HAS', 0) + avg_results.get('EXPOSED', 0)
    print(f"  Aircraft Destroyed: {total_destroyed:.2f}")
    print(f"    - In HAS: {avg_results.get('HAS', 0):.2f}")
    print(f"    - Exposed: {avg_results.get('EXPOSED', 0):.2f}")
    print(f"    - From Splash: {avg_results.get('splash_kills', 0):.2f}")
    print(f"  Empty HAS Destroyed: {avg_results.get('EMPTY_HAS', 0):.2f}")
```

```
    print(f"\nBY MISSILE TYPE:")
    for mtype, avg in avg_results.get('by_missile_type', {}).items():
        intercepted = avg_results.get('intercepted_by_type', {}).get(mtype, 0)
        skipped = avg_results.get('skipped_due_to_no_targets', {}).get(mtype, 0)
        print(f"  {mtype}: {avg:.2f} destroyed, {intercepted:.2f} intercepted, {skipped:.2f}
skipped")

    print(f"\nBY AIRCRAFT TYPE:")
    for atype, avg in avg_results.get('by_aircraft_type', {}).items():
        print(f"  {atype}: {avg:.2f}")

    # HAS damage analysis
    if avg_results.get('HAS_details') and has_count > 0:
        print(f"\nHAS DAMAGE ANALYSIS:")

        has_indices = [i for i, ac in enumerate(scenario['aircraft']) if ac['shelter'] == 'HAS']
        hits_accumulator = [defaultdict(int) for _ in range(len(has_indices))]
        hit_counts = [0 for _ in range(len(has_indices))]
        destroyed_counts = [0 for _ in range(len(has_indices))]

        for sim_status in avg_results['HAS_details']:
            for rel_idx, abs_idx in enumerate(has_indices):
                if abs_idx < len(sim_status) and sim_status[abs_idx]['shelter'] == 'HAS':
                    ac = sim_status[abs_idx]
                    hit_counts[rel_idx] += ac['hit_count']
                    for mtype, count in ac['hit_log'].items():
                        hits_accumulator[rel_idx][mtype] += count
                    if ac['destroyed']:
                        destroyed_counts[rel_idx] += 1

        for rel_idx, abs_idx in enumerate(has_indices):
            destruction_rate = destroyed_counts[rel_idx] / avg_results['valid_runs']
            avg_hits = hit_counts[rel_idx] / avg_results['valid_runs']
            ac_type = scenario['aircraft'][abs_idx]['type']
            print(f"  HAS {abs_idx + 1} ({ac_type}): {avg_hits:.2f} avg hits, "
                  f"{destruction_rate:.1%} destruction rate")

    # EXPOSED aircraft analysis
    if avg_results.get('HAS_details') and exposed_count > 0:
        print(f"\nEXPOSED AIRCRAFT ANALYSIS:")

    exposed_indices = [i for i, ac in enumerate(scenario['aircraft']) if ac['shelter'] ==
'EXPOSED']
    exposed_hits_accumulator = [defaultdict(int) for _ in range(len(exposed_indices))]
    exposed_hit_counts = [0 for _ in range(len(exposed_indices))]
    exposed_destroyed_counts = [0 for _ in range(len(exposed_indices))]
    splash_destroyed_counts = [0 for _ in range(len(exposed_indices))]
    direct_hit_destroyed_counts = [0 for _ in range(len(exposed_indices))]  # NEW

    for sim_status in avg_results['HAS_details']:
        for rel_idx, abs_idx in enumerate(exposed_indices):
            if abs_idx < len(sim_status) and sim_status[abs_idx]['shelter'] == 'EXPOSED':
                ac = sim_status[abs_idx]
                exposed_hit_counts[rel_idx] += ac['hit_count']
                for mtype, count in ac['hit_log'].items():
                    exposed_hits_accumulator[rel_idx][mtype] += count
                if ac['destroyed']:
                    exposed_destroyed_counts[rel_idx] += 1

                    if ac.get('splash_kill', False):
                        splash_destroyed_counts[rel_idx] += 1
                    else:
```

```
                        direct_hit_destroyed_counts[rel_idx] += 1

    for rel_idx, abs_idx in enumerate(exposed_indices):
        destruction_rate = exposed_destroyed_counts[rel_idx] / avg_results['valid_runs']
        avg_hits = exposed_hit_counts[rel_idx] / avg_results['valid_runs']
        splash_rate = splash_destroyed_counts[rel_idx] / avg_results['valid_runs']
        direct_hit_rate = direct_hit_destroyed_counts[rel_idx] / avg_results['valid_runs']
        ac_type = scenario['aircraft'][abs_idx]['type']

        missile_breakdown = []
        for mtype, hits in exposed_hits_accumulator[rel_idx].items():
            avg_missile_hits = hits / avg_results['valid_runs']
            if avg_missile_hits > 0:
                missile_breakdown.append(f"{mtype}: {avg_missile_hits:.2f}")

        missile_detail = f" [{', '.join(missile_breakdown)}]" if missile_breakdown else ""
        splash_detail = f", {splash_rate:.1%} splash kills" if splash_rate > 0 else ""
        direct_hit_detail = f", {direct_hit_rate:.1%} direct hits" if direct_hit_rate > 0 else ""

        print(f"  EXPOSED {abs_idx + 1} ({ac_type}): {avg_hits:.2f} avg hits{missile_detail}, "
              f"{destruction_rate:.1%} destruction rate{direct_hit_detail}{splash_detail}")

    # Summary statistics for exposed aircraft
    total_exposed_destruction_rate = sum(exposed_destroyed_counts) / (len(exposed_indices) *
avg_results['valid_runs']) if exposed_indices else 0
    total_exposed_hits = sum(exposed_hit_counts) / (len(exposed_indices) *
avg_results['valid_runs']) if exposed_indices else 0
    total_splash_kills = sum(splash_destroyed_counts) / avg_results['valid_runs'] if
avg_results['valid_runs'] > 0 else 0
    total_direct_hits = sum(direct_hit_destroyed_counts) / avg_results['valid_runs'] if
avg_results['valid_runs'] > 0 else 0

    print(f"\n  EXPOSED SUMMARY:")
    print(f"    Average destruction rate: {total_exposed_destruction_rate:.1%}")
    print(f"    Average hits per exposed aircraft: {total_exposed_hits:.2f}")
    print(f"    Total direct hit kills: {total_direct_hits:.2f}")
    print(f"    Total splash kills: {total_splash_kills:.2f}")

    # EMPTY HAS analysis
    if avg_results.get('HAS_details') and empty_has_count > 0:
        print(f"\nEMPTY HAS ANALYSIS:")

        empty_has_hits_accumulator = [defaultdict(int) for _ in range(empty_has_count)]
        empty_has_hit_counts = [0 for _ in range(empty_has_count)]
        empty_has_destroyed_counts = [0 for _ in range(empty_has_count)]

        # Extract empty HAS data from simulation runs
        for run_data in avg_results.get('full_aircraft_runs', []):
            if 'empty_has_status' in run_data:
                empty_has_data = run_data['empty_has_status']
                for i, has in enumerate(empty_has_data):
                    if i < len(empty_has_hit_counts):
                        empty_has_hit_counts[i] += has.get('hit_count', 0)
                        for mtype, count in has.get('hit_log', {}).items():
                            empty_has_hits_accumulator[i][mtype] += count
                        if has.get('destroyed', False):
                            empty_has_destroyed_counts[i] += 1

        for i in range(empty_has_count):
            destruction_rate = empty_has_destroyed_counts[i] / avg_results['valid_runs'] if
avg_results['valid_runs'] > 0 else 0
            avg_hits = empty_has_hit_counts[i] / avg_results['valid_runs'] if
avg_results['valid_runs'] > 0 else 0
```

```
            # Show missile type breakdown for this empty HAS
            missile_breakdown = []
            for mtype, hits in empty_has_hits_accumulator[i].items():
                avg_missile_hits = hits / avg_results['valid_runs']
                if avg_missile_hits > 0:
                    missile_breakdown.append(f"{mtype}: {avg_missile_hits:.2f}")

            missile_detail = f" [{', '.join(missile_breakdown)}]" if missile_breakdown else ""

            print(f"  EMPTY HAS {i + 1}: {avg_hits:.2f} avg hits{missile_detail}, "
                  f"{destruction_rate:.1%} destruction rate")

        # Summary statistics for empty HAS
        total_empty_has_destruction_rate = sum(empty_has_destroyed_counts) / (empty_has_count *
avg_results['valid_runs']) if empty_has_count > 0 and avg_results['valid_runs'] > 0 else 0
        total_empty_has_hits = sum(empty_has_hit_counts) / (empty_has_count *
avg_results['valid_runs']) if empty_has_count > 0 and avg_results['valid_runs'] > 0 else 0

        print(f"\n  EMPTY HAS SUMMARY:")
        print(f"    Average destruction rate: {total_empty_has_destruction_rate:.1%}")
        print(f"    Average hits per empty HAS: {total_empty_has_hits:.2f}")
        print(f"    Total empty HAS destroyed: {avg_results.get('EMPTY_HAS', 0):.2f}")
        print(f"    Deception effectiveness: {(1 - total_empty_has_destruction_rate):.1%}
survival rate")

# --- Monte Carlo Execution ---
def monte_carlo(n_runs: int, missiles: List[Dict], aircraft: List[Dict], empty_has: List[Dict] =
None) -> Dict:
    """Run Monte Carlo simulation with multiple iterations including empty HAS."""
    cumulative_results = {
        'HAS': 0,
        'EXPOSED': 0,
        'EMPTY_HAS': 0,
        'by_missile_type': defaultdict(int),
        'by_aircraft_type': defaultdict(int),
        'splash_kills': 0,
        'intercepted_by_type': defaultdict(int),
        'skipped_due_to_no_targets': defaultdict(int),
        'HAS_details': [],
        'impact_logs': [],
        'full_aircraft_runs': []
    }

    print(f"Running {n_runs} Monte Carlo simulations...")

    for run in range(n_runs):
        if n_runs > 10 and run % max(1, n_runs // 10) == 0:
            print(f"Progress: {run}/{n_runs} ({100*run//n_runs}%)")

        try:
            result = run_simulation(
                missiles=missiles.copy(),
                aircraft=aircraft.copy(),
                empty_has=empty_has.copy() if empty_has else None
            )

            # Aggregate results
            cumulative_results['HAS'] += result['HAS']
            cumulative_results['EXPOSED'] += result['EXPOSED']
            cumulative_results['EMPTY_HAS'] += result.get('EMPTY_HAS', 0)
            cumulative_results['splash_kills'] += result['splash_kills']
```

```
            for key in ['by_missile_type', 'intercepted_by_type', 'skipped_due_to_no_targets',
'by_aircraft_type']:
                for mtype, count in result[key].items():
                    cumulative_results[key][mtype] += count

            cumulative_results['HAS_details'].append(result['aircraft_status'])
            cumulative_results['impact_logs'].append(result['impact_log'])

            # Store complete run data including empty HAS
            run_data = {
                'aircraft_status': result['aircraft_status'],
                'empty_has_status': result.get('empty_has_status', []),
                'impact_log': result['impact_log']
            }
            cumulative_results['full_aircraft_runs'].append(run_data)

        except Exception as e:
            print(f"Error in simulation run {run}: {e}")
            continue

    # Calculate averages
    valid_runs = len(cumulative_results['full_aircraft_runs'])
    if valid_runs == 0:
        raise ValueError("No successful simulation runs completed")

    avg_results = {
        'HAS': cumulative_results['HAS'] / valid_runs,
        'EXPOSED': cumulative_results['EXPOSED'] / valid_runs,
        'EMPTY_HAS': cumulative_results['EMPTY_HAS'] / valid_runs,
        'by_missile_type': {k: v / valid_runs for k, v in
cumulative_results['by_missile_type'].items()},
        'splash_kills': cumulative_results['splash_kills'] / valid_runs,
        'by_aircraft_type': {k: v / valid_runs for k, v in
cumulative_results['by_aircraft_type'].items()},
        'intercepted_by_type': {k: v / valid_runs for k, v in
cumulative_results['intercepted_by_type'].items()},
        'skipped_due_to_no_targets': {k: v / valid_runs for k, v in
cumulative_results['skipped_due_to_no_targets'].items()},
        'HAS_details': cumulative_results['HAS_details'],
        'impact_logs': cumulative_results['impact_logs'],
        'full_aircraft_runs': cumulative_results['full_aircraft_runs'],
        'TOTAL': (cumulative_results['HAS'] + cumulative_results['EXPOSED']) / valid_runs,
        'valid_runs': valid_runs
    }
    return avg_results


if __name__ == "__main__":
    print("Advanced Missile Strike Simulation - Manual HAS Attribution with Empty HAS")
    print("=============================================================================")


    scenarios = [
        {
            'name': 'High HAS Protection with Decoys - Amari Air Base',
            'missiles': [{'type': 'Iskander-M'}] * 50 +
            [{'type': 'Iskander-K'}] * 10 +
            [{'type': 'OTR-21'}]* 5 +
            [{'type': 'Kalibr'}] * 5 +
            [{'type': 'P-800'}] * 2 +
            [{'type': 'Kh-101'}] * 20,
            'aircraft': [
                # APRON NORTH-EAST
```

```
            {'type': 'fighterEXP', 'x': 327, 'y': -291, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 349, 'y': -281, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 373, 'y': -271, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 397, 'y': -261, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 421, 'y': -251, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 445, 'y': -241, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 469, 'y': -231, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 493, 'y': -221, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 517, 'y': -211, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 541, 'y': -201, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 565, 'y': -191, 'shelter': 'EXPOSED'},

            #APRON SOUTH LARGE
            {'type': 'awacs', 'x': -771, 'y': -1336, 'shelter': 'EXPOSED'},
            {'type': 'tanker', 'x': -820, 'y': -1357, 'shelter': 'EXPOSED'},
            {'type': 'tanker', 'x': -869, 'y': -1377, 'shelter': 'EXPOSED'},
            {'type': 'tanker', 'x': -929, 'y': -1402, 'shelter': 'EXPOSED'},

            #APRON SOUTH SMALL
            {'type': 'fighterEXP', 'x': -713, 'y': -1306, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': -672, 'y': -1286, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': -636, 'y': -1268, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': -600, 'y': -1251, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': -562, 'y': -1233, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': -527, 'y': -1217, 'shelter': 'EXPOSED'},

            #APRON SOUTH-SOUTH-EAST
            {'type': 'fighterEXP', 'x': 83, 'y': -938, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 106, 'y': -928, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 128, 'y': -920, 'shelter': 'EXPOSED'},
            {'type': 'fighterEXP', 'x': 150, 'y': -910, 'shelter': 'EXPOSED'},

            # HAS NORTH-WEST (occupied)
            {'type': 'fighterHAS', 'x': -979, 'y': -619, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -961, 'y': -577, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -983, 'y': -392, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -935, 'y': -363, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -906, 'y': -323, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -886, 'y': -470, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -853, 'y': -436, 'shelter': 'HAS'},
            {'type': 'fighterHAS', 'x': -781, 'y': -407, 'shelter': 'HAS'},
        ],
        'empty_has': [
            # Empty HAS
            {'x': -596, 'y': -167},
            {'x': -537, 'y': -142},
            {'x': -497, 'y': -303},
            {'x': -425, 'y': -303},
            {'x': -440, 'y': -162},
            {'x': -354, 'y': -136},
            {'x': -304, 'y': -245},
            {'x': -44, 'y': -1083},
            {'x': 10, 'y': -1101},
            {'x': -75, 'y': -1167},
            {'x': -3, 'y': -1170},
            {'x': -148, 'y': -1170},
            {'x': -636, 'y': -284},
        ]
    },
    {
        'name': 'Low HAS Protection with Decoys - Amari Air Base',
        'missiles': [{'type': 'Iskander-M'}] * 10 +
        [{'type': 'Iskander-K'}] * 10 +
```

```
            [{'type': 'OTR-21'}]* 5 +
            [{'type': 'Kalibr'}] * 2 +
            [{'type': 'P-800'}] * 2 +
            [{'type': 'Kh-101'}] *5,
            'aircraft': [
                # APRON NORTH-EAST
                {'type': 'fighterEXP', 'x': 327, 'y': -291, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 349, 'y': -281, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 373, 'y': -271, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 397, 'y': -261, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 421, 'y': -251, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 445, 'y': -241, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 469, 'y': -231, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 493, 'y': -221, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 517, 'y': -211, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 541, 'y': -201, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': 565, 'y': -191, 'shelter': 'EXPOSED'},

                #APRON SOUTH LARGE
                {'type': 'awacs', 'x': -771, 'y': -1336, 'shelter': 'EXPOSED'},
                {'type': 'tanker', 'x': -820, 'y': -1357, 'shelter': 'EXPOSED'},
                {'type': 'tanker', 'x': -869, 'y': -1377, 'shelter': 'EXPOSED'},
                {'type': 'tanker', 'x': -929, 'y': -1402, 'shelter': 'EXPOSED'},

                #APRON SOUTH SMALL
                {'type': 'fighterEXP', 'x': -713, 'y': -1306, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': -672, 'y': -1286, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': -636, 'y': -1268, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': -600, 'y': -1251, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': -562, 'y': -1233, 'shelter': 'EXPOSED'},
                {'type': 'fighterEXP', 'x': -527, 'y': -1217, 'shelter': 'EXPOSED'},

                # HAS NORTH-WEST (limited occupied)
                {'type': 'fighterHAS', 'x': -979, 'y': -619, 'shelter': 'HAS'},
                {'type': 'fighterHAS', 'x': -961, 'y': -577, 'shelter': 'HAS'},
                {'type': 'fighterHAS', 'x': -983, 'y': -392, 'shelter': 'HAS'},
                {'type': 'fighterHAS', 'x': -935, 'y': -363, 'shelter': 'HAS'},
            ],
            'empty_has': [
                # Empty HAS
                {'x': -906, 'y': -323},
                {'x': -3, 'y': -1170},
                {'x': -886, 'y': -470},
                {'x': -853, 'y': -436},
                {'x': -781, 'y': -407},
                {'x': -596, 'y': -167},
                {'x': -537, 'y': -142},
                {'x': -497, 'y': -303},
                {'x': -425, 'y': -303},
                {'x': -440, 'y': -162},
                {'x': -354, 'y': -136},
                {'x': -304, 'y': -245},
                {'x': -44, 'y': -1083},
                {'x': 10, 'y': -1101},
                {'x': -75, 'y': -1167},
                {'x': -3, 'y': -1170},
                {'x': -148, 'y': -1170},
            ]
        }
    ]

    # Run scenarios
    for i, scenario in enumerate(scenarios, 1):
```

```
        print(f"\nProcessing {scenario['name']}...")

        try:
            avg_results = monte_carlo(
                n_runs=1,  # <------------------------------------------------------------------
Monte Carlo runs
                missiles=scenario['missiles'],
                aircraft=scenario['aircraft'],
                empty_has=scenario.get('empty_has', [])
            )

            # Print results
            print_detailed_results(i, avg_results, scenario)

            # Show visualization for first run
            try:
                if avg_results['full_aircraft_runs'] and avg_results['impact_logs']:
                    first_run_data = avg_results['full_aircraft_runs'][0]
                    first_run_aircraft = first_run_data['aircraft_status']
                    first_run_empty_has = first_run_data.get('empty_has_status', [])
                    first_run_impacts = avg_results['impact_logs'][0]

                    print(f"\nGenerating visualization for {scenario['name']}...")
                    plot_simulation(
                        first_run_aircraft,
                        first_run_impacts,
                        first_run_empty_has,
                        title=f"{scenario['name']} - Missile Impact Run",
                        shapefile_path=r"C:\Users\Acer\OneDrive\Bureaublad\GEOTEST\Amari.shp"
                    )
                else:
                    print("No data available for visualization")
            except Exception as viz_error:
                print(f"Visualization error: {viz_error}")

        except Exception as e:
            print(f"Error processing scenario {i}: {e}")
            import traceback
            traceback.print_exc()
            continue

    print(f"\nSimulation completed successfully!")
    print("Configuration used:")
    for key, value in CONFIG.items():
        print(f"  {key}: {value}")
```