Ballistic Delivery of Humanitarian Aid

Team Members: Steven Vanden Noven, David Shunk, Caden McDevitt, Dylan Cha

Course: AME-30251

Instructor: Professor McClarren

Date: 5/2/25

Google Colab Link:
https://colab.research.google.com/drive/1D7YGl6aex7qq84Sj-ojRjJAghHC22dVe?usp=sharing

Part 4 Video Link:

Introduction

Throughout the past 60 years there have been attempts to utilize ballistic projectiles to deliver aid. Notable attempts include Project HARP, Rocket Lab, and SpaceX's Starship. In the project, the team attempted to model trajectories as well as analyze different situations. Project objectives include numerical simulation of a projectile's trajectory, optimization of trajectory, predictive modeling, radar coverage and detection modeling, and visualization of projectile and radar detection volume. Goals for each part are as follows: integrate equations of motion to predict projectile landing given initial conditions, utilize optimization to find launch angles (θ, ψ) and initial speed (v_0) required for the shortest path to target, develop a machine learning model to estimate projectile landing position given velocity and measurements, and model detection volume and coverage of a military radar station in Siberia to determine time of detection for a trajectory launched from Minot AFB.

Part 1: Numerical Simulation of Projectile Trajectory

In Part 1 of the project the objective was to numerically integrate the equations of motion to predict projectile landing location given initial conditions. The following assumptions were made: no external forces acting on the rocket aside from gravity, drag, coriolis force, and thrust, the earth is given as a semi-spheroid under the WGS84 model, the rocket was modeled with StarShip like parameters, including the addition of thrust and mass loss due to fuel as variables in this simulation, and the rocket performs no unusual maneuvers (pitchover, full orbits, etc.) as well as follows a parabolic trajectory with thrust only acting in the direction of movement. Thrust force was modeled off of Starship. The following formulas were used to transform from geodetic to ECEF coordinates:

$$N = a / \sqrt{(1 - e^2 \cdot \sin^2(\varphi))}$$

$$X = (N + h) \cdot \cos(\varphi) \cdot \cos(\lambda)$$

$$Y = (N + h) \cdot \cos(\varphi) \cdot \sin(\lambda)$$

$$Z = [N \cdot (1 - e^2) + h] \cdot \sin(\varphi)$$

Where a is the semi major axis of the earth (6,378,137) N is the vertical radius of curvature, e is the eccentricity of the Earth (.08181), φ is the geodetic latitude from the equator, λ is the geodetic longitude from the prime meridian, h is the height above Earth's surface, and X, Y, and Z are the ECEF coordinates in meters.

To convert a velocity vector from the local ENU frame to the ECEF frame, the following rotation matrix is applied:

$$R = \begin{bmatrix} -\sin(\lambda), & -\sin(\varphi)\cdot\cos(\lambda), & \cos(\varphi)\cdot\cos(\lambda) \\ \cos(\lambda), & -\sin(\varphi)\cdot\sin(\lambda), & \cos(\varphi)\cdot\sin(\lambda) \\ 0, & \cos(\varphi), & \sin(\varphi) \end{bmatrix}$$

Then the velocity in ECEF is given by:

$$v_{ECEF} = R * v_{ENU}$$

where v_ENU is the input velocity vector in the local East-North-Up frame (with components [v_East, v_North, v_Up]), and v_ECEF is the output velocity vector in ECEF coordinates. φ and λ are the geodetic latitude and longitude of the local origin, expressed in radians.

To convert ECEF coordinates back to geodetic coordinates, an iterative method is used. The longitude is computed first:

$$\lambda = \operatorname{atan2}(y, x)$$

An initial estimate of the latitude is made with:

$$\varphi = \operatorname{atan2}(z, \sqrt{(x^2 + y^2)})$$

The latitude and altitude are then refined iteratively using:

$$N = a / \sqrt{(1 - e^2 \cdot \sin^2(\varphi))}$$

$$h = \sqrt{(x^2 + y^2)} / \cos(\varphi) - N$$

$$\varphi = \operatorname{atan2}(z, \sqrt{(x^2 + y^2)} \cdot (1 - e^2 \cdot N / (N + h)))$$

In these equations, x, y, and z are the input ECEF coordinates in meters, and a and e are the semi-major axis and eccentricity of the Earth, respectively. φ, λ, and h are the geodetic latitude, longitude, and height above the ellipsoid, returned in degrees and meters.

To convert a velocity vector from ECEF to ENU coordinates, the inverse of the ENU-to-ECEF rotation matrix is used:

$$R^{-1} = \begin{bmatrix} -\sin(\lambda), & \cos(\lambda), & 0 \\ -\sin(\varphi)\cdot\cos(\lambda), & -\sin(\varphi)\cdot\sin(\lambda), & \cos(\varphi) \\ \cos(\varphi)\cdot\cos(\lambda), & \cos(\varphi)\cdot\sin(\lambda), & \sin(\varphi) \end{bmatrix}$$

Then the local velocity vector is computed with:

$$v_{ENU} = R^{-1} * v_{ECEF}$$

Where v_ECEF is the input velocity vector in the Earth-Centered, Earth-Fixed frame, and v_ENU is the resulting velocity vector in the local East-North-Up frame. The variables φ and λ are the geodetic latitude and longitude of the local frame's origin, expressed in radians.

For the equations of motion atmospheric conditions, and general physics based formulas for gravity, and coriolis were all considered. The atmosphere is divided into layers, each defined by a base altitude h_b, a top altitude h_t, a lapse rate L, a base temperature T_b, and a base pressure P_b. These parameters define the structure of the U.S. Standard Atmosphere and allow pressure, temperature, and density to be calculated as a function of altitude. The standard atmospheric layers used in this model include altitudes from 0 to 84.852 km, with varying lapse rates depending on the stratum. The pressure at altitude h is computed using different formulas depending on whether the lapse rate L is zero (an isothermal layer) or non-zero. If the lapse rate L = 0, the pressure is calculated using the formula

$$P = P_b * \exp(-g_0(h - h_b) / (R * T_b))$$

whereas if the lapse rate is non-zero, the temperature at height h is first computed using

$$T = T_b + L(h - h_b)$$

and then the pressure is calculated using

$$P = P_b * (T / T_b)^{(-g_0 / (L * R))}.$$

In these equations, P is the atmospheric pressure at altitude h in Pascals, P_b is the base pressure of the current atmospheric layer, T_b is the base temperature in Kelvin, L is the lapse rate in Kelvin per meter, g_0 is standard gravitational acceleration (9.80665 m/s²), R is the specific gas constant for dry air (287.05 J/kg·K), and h_b is the base altitude of the layer in meters. For altitudes above 85 km, a constant pressure of approximately 0.3734 Pa is used.

The temperature at a given altitude is calculated using

$$T = T_b + L(h - h_b)$$

if the lapse rate $L \neq 0$, or simply $T = T_b$ if $L = 0$. Once temperature and pressure are known, the air density ρ can be found from the ideal gas law: $\rho = P / (R * T)$. Here, ρ is the air density in kg/m^3, P is pressure in Pascals, T is temperature in Kelvin, and R is again the specific gas constant.

The speed of sound c at altitude is determined by the formula

$$c = \text{sqrt}(\gamma * R * T)$$

where γ is the specific heat ratio for air (typically 1.4), R is the specific gas constant, and T is temperature in Kelvin.

Gravitational acceleration at a given position vector r in ECEF coordinates is given by

$$g = -GM / |r|^2 * (r / |r|)$$

where G is the universal gravitational constant, M is the mass of the Earth, r is the ECEF position vector, and $r / |r|$ is the unit vector in the direction of r.

Drag force is calculated using the equation

$$F_d = -0.5 * C_d * \rho * A * v^2 * (v / |v|),$$

where C_d is the drag coefficient (which varies with Mach number), ρ is air density, A is the reference area of the projectile, v is the velocity magnitude, and $v / |v|$ is the unit direction vector of velocity. If the Mach number M (the ratio of velocity to speed of sound) is less than 1, the drag coefficient is a constant C_{D0}. If $M > 1$, the coefficient increases using

$$C_d = C_{D0} + C_{D1} / \text{sqrt}(\max(M^2 - 1, \varepsilon)),$$

where ε is a small value to prevent division by zero.

Coriolis acceleration is calculated as

$$a_{coriolis} = -2 * \omega \times v$$

where ω is the angular velocity vector of Earth's rotation and v is the velocity of the projectile in the ECEF frame. The current mass of the projectile at time t is determined by evaluating whether the fuel burn has completed or not. If the user-specified thrust duration is less than the possible burn time (based on fuel available and burn rate), then the fuel mass linearly decreases until thrust ends. Otherwise, the mass decreases until fuel is exhausted, and the remaining mass is just the dry mass of the vehicle.

Thrust acceleration is calculated using

$$a\_thrust = (F\_thrust / m) * (v / |v|)$$

where F_thrust is the magnitude of thrust, m is the current mass at time t, and v / |v| is the direction of the current velocity vector. If the speed is too small or the thrust duration has ended, no thrust is applied. The full equations of motion for the projectile are based on Newton's Second Law in vector form and include gravitational, drag, Coriolis, and thrust accelerations. The total acceleration vector is

$$a\_total = a\_grav + a\_drag + a\_coriolis + a\_thrust$$

The resulting state derivative consists of the velocity components and the corresponding acceleration components

$$d/dt [x, y, z, v\_x, v\_y, v\_z] = [v\_x, v\_y, v\_z, a\_x, a\_y, a\_z].$$

Finally, the integration of the motion is stopped using an event function once the geodetic altitude of the projectile reaches zero, which indicates it has hit the ground.

In order to simulate the simulate_trajectory method is introduced, which takes the location and state of the rocket and uses solve_ivp to solve for the values at every location throughout the duration of the flight. An event method that calculates the impact location of the rocket was included, which when triggered stops the solve_ivp method. All of the parameters of the flight are plotted to ensure that the flight makes sense, including the altitude vs. time, range vs time, speed vs time, gravitational force vs time, coriolis force vs time, and drag vs. time. The final landing coordinates are also printed. The trajectory of the rocket is projected onto an interactive model of earth, with initial and final impact points plotted, as well as the trajectory above the planet. The launch and impact points correspond with the coordinates on the real planet Earth. All results are plotted below:

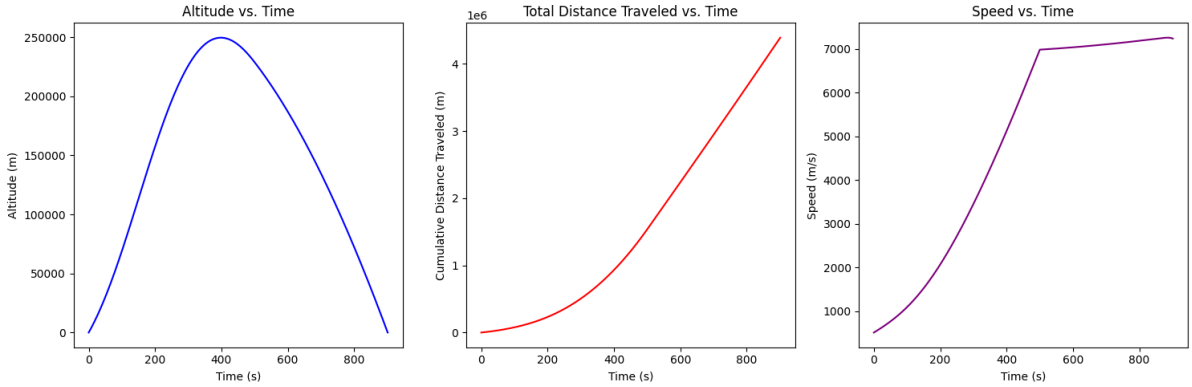

Figure 1: Altitude vs Time, Distance vs Time, and Speed vs Time graphs

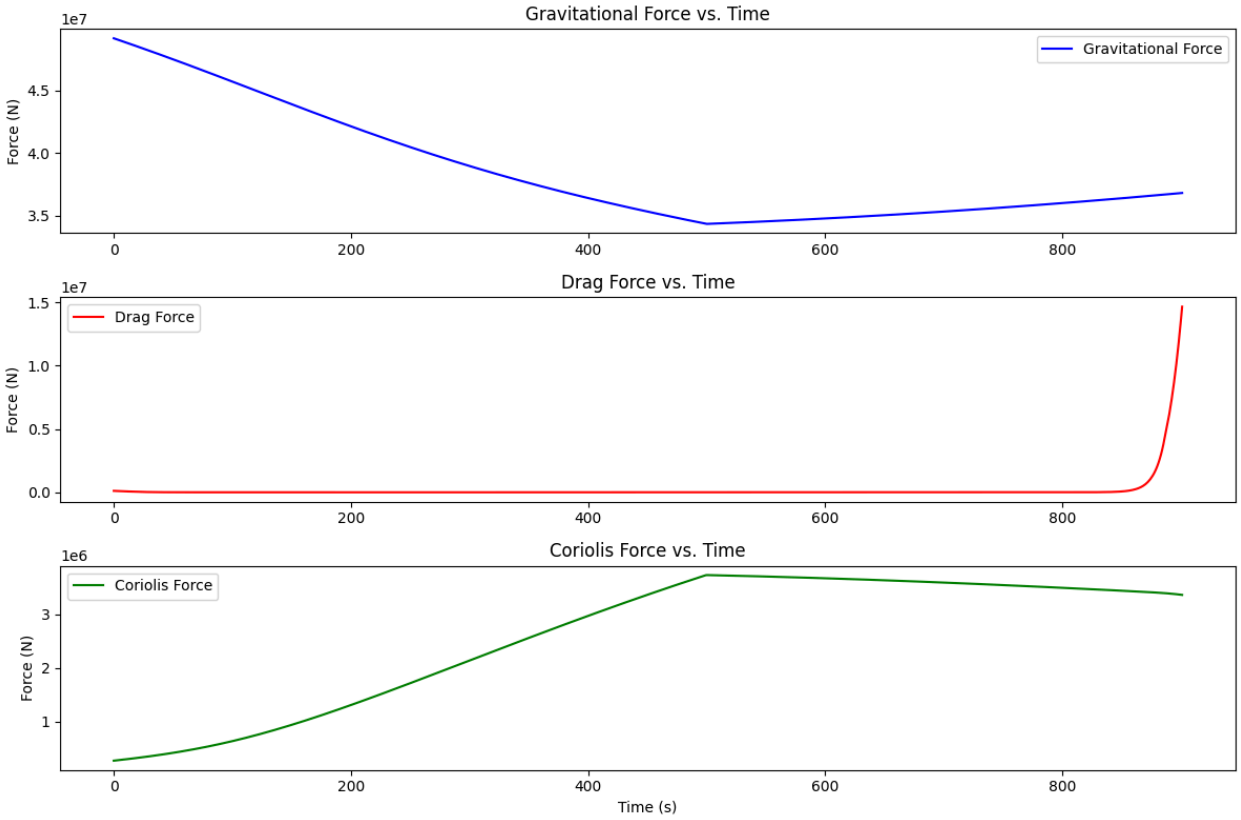

Figure 2: Gravitational force vs Time, Drag force vs Time, and Coriolis Force vs Time Graphs

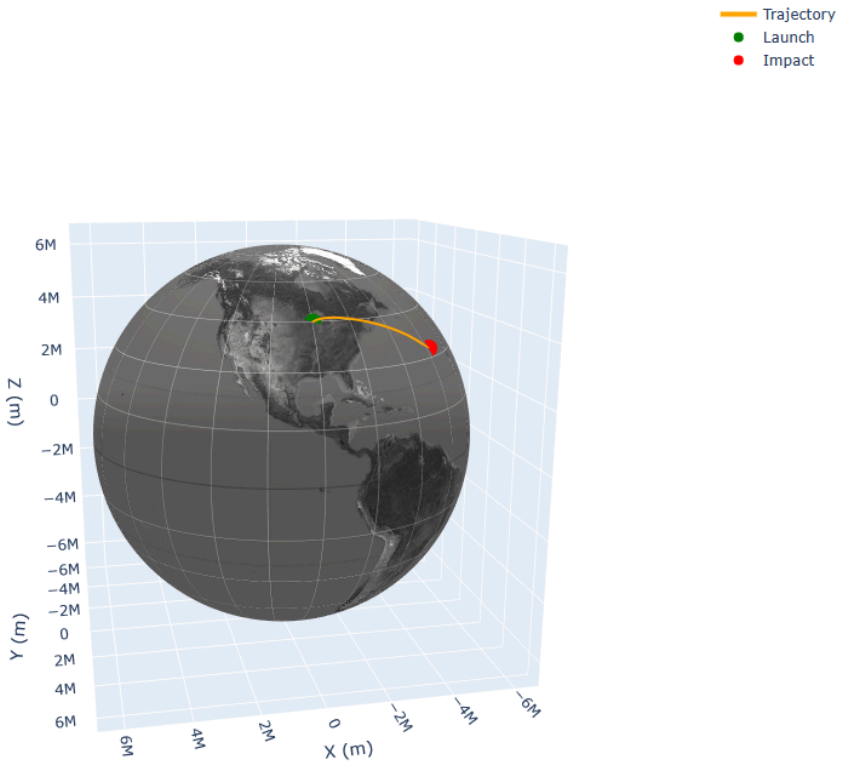

Figure 3. 3D figure of launch and impact with trajectory

Part 2: Trajectory Optimization

In this section, initial trajectory parameters (v_0, theta, psi) are optimized for minimum initial speed *v_0* with impact coordinate constraints of +/- 0.1 degrees of geodetic distance from given target coordinates. Using the optimization function and the determined optimal lunch assignments, the trajectory parameters for launches to targets in Table 1 from launch sites in Table 2 are optimized, given that each base can launch up to four projectiles. Assumptions made for the optimization problem include that launch distance is proportional to launch time - therefore target assignments were based on distance between targets. Furthermore, optimization constraints were satisfied when impact coordinates were within +/- 0.1 degrees of geodetic distance which translates to about 111 kilometers. In all instances of optimization, all other initial parameters (ie. burn rate, thrust force) were held constant.

Aid delivery location	**Latitude (N)**	**Longitude (E)**
Dombarovsky (Yasny, Orenburg Oblast)	51.1100°	59.8400°
Kozelsk (Kaluga Oblast)	54.0340°	35.7800°
Tatishchevo (Saratov Oblast)	51.6670°	45.1170°
Uzhur (Krasnoyarsk Krai)	55.3330°	89.8170°
Teykovo (Ivanovo Oblast)	56.8510°	40.5320°
Yoshkar-Ola (Mari El Republic)	56.6330°	47.8710°
Nizhny Tagil (Sverdlovsk Oblast)	58.0830°	60.0500°
Novosibirsk (Novosibirsk Oblast)	55.3330°	83.0000°
Irkutsk (Irkutsk Oblast)	52.3170°	104.2330°

Table 1: Latitude and Longitude Coordinates of Aid Delivery Locations.

Missile Base	**Latitude (N)**	**Longitude (W)**
Francis E. Warren AFB (Wyoming)	41.1265°	-104.8668°
Minot AFB (North Dakota)	48.41583°	-101.35806°
Malmstrom AFB (Montana)	47.5028°	-111.1857°

Table 2: Latitude and Longitude Coordinates of U.S. Aid Launch Silos.

Table 3 below lists target assignments and optimized trajectory parameters for each launch where $v0$ is the initial speed, θ is the elevation angle with respect to horizontal and ψ is the launch azimuth angle with respect to geographic north with positive degrees moving clockwise.

Target Site	Launch Site	$v0$ (m/s)	θ (°)	ψ (°)
Dombarovsky	Warren	1470.80	51.70	2.23
Kozelsk	Malmstrom	1287.45	56.87	13.00
Tatishchevo	Malmstrom	1344.51	55.21	7.88
Uzhur	Warren	1480.27	53.13	-15.28
Teykovo	Malmstrom	1257.34	57.82	9.47
Yoshkar-Ola	Malmstrom	1289.21	57.35	5.33
Nizhny Tagil	Warren	1387.81	54.51	1.46
Novosibirsk	Warren	1474.90	52.94	-11.48
Irkutsk	Minot	1419.58	55.53	-22.71

Table 3. Optimized Trajectory Parameters

Each launch trajectory is simulated and visualized in Figure 4:

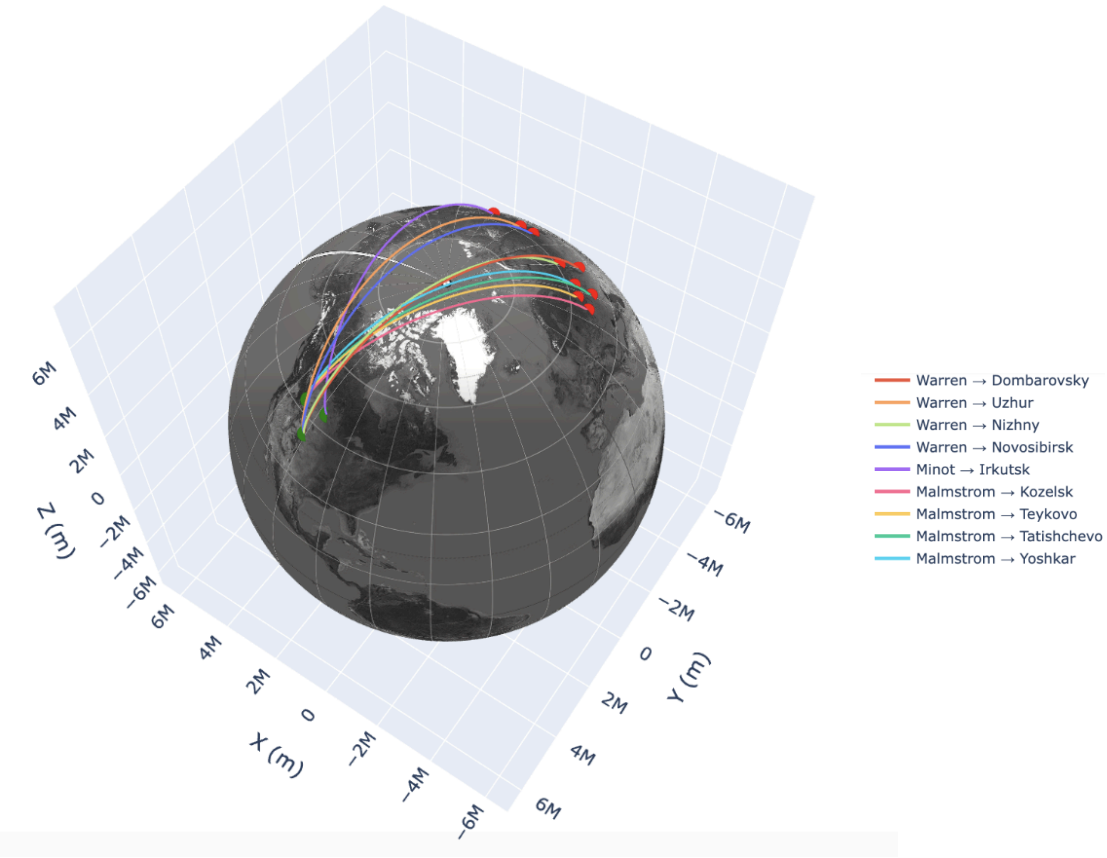

Figure 4. 3D figure of optimized launch trajectories

As seen in Figure 4, to minimize initial velocity for each projectile the optimal course was to fly north over the Arctic to reach the target sites. The coriolis force on the north-bound projectiles results in an eastward deflection, causing the optimizer to aim its initial azimuth angle ψ farther west. Overall, results for trajectory optimization are successful, as each projectile reaches its target within the specified constraints while following a reasonable flight path.

Part 3: Predictive Modeling

To predict the landing position of ballistic trajectories based on initial launch conditions, multiple decision tree based machine learning models were implemented and evaluated. After comparing performance across several configurations, an XGBoost Regressor was selected as the final model due to its superior accuracy and efficiency. The final predictive model uses the XGBoost regression algorithm, configured with 100 trees and a maximum tree depth of 18. XGBoost was chosen for its ability to handle non-linear interactions and its robustness to overfitting, thanks to its built-in regularization. Compared to other tree-based methods like Random Forests, XGBoost

provides better control over bias-variance tradeoff and is optimized for speed and performance. Its method of boosting, where it corrects its previous mistakes for sequential tree building, allows for greater accuracy than the bagging utilized by Random Forests, where many independent trees are aggregated.

To train this model, 2000 cases were simulated using Part 1. The inputs were latitude, longitude, altitude, and the three components of velocity. Unfortunately, one very limiting factor of the training data generation was the range of the inputs. Increasing the ranges to include higher velocity and higher altitude cases would have helped validate the model and improve prediction for later parts of the project, but unfortunately increasing those values was too computationally expensive, so a high range of training values had to be sacrificed for computational cost. In the end, it was decided that reasonable altitude values that were not too computationally expensive were 0 to 20000 meters, reasonable velocity values were -1000 to 1000 meters per second in the east and north directions, and reasonable upwards velocity values were 0 to 1000 meters per second. The target variables were the geodetic coordinates of the final impact location, filtered to only include impacts at low altitude to ensure consistency in model output. Then, 90 percent of that data was used for training and 10 percent was used for validation and testing. The performance of the machine learning models was then evaluated using a mean squared error calculation between the predicted and actual landing coordinates and a percent error calculation for each test case based on Haversine distance between the predicted and actual landing locations and the relative distance to the launch site. The equations for mean squared error and Haversine distance can be seen below.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left((y_i^{\text{pred}} - y_i^{\text{true}})^2\right)$$

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where n is the number of samples, y_i^{pred} is the predicted value, y_i^{true} is the true value, ϕ is the latitudes in radians, λ is the longitude in radians, and r is the radius of the Earth in meters, or 6371000 meters.

Overall, the XGBoost model was able to achieve a median percent error of 4.4% for its 200 test cases. However, despite the machine learning models proving to be mostly accurate, the dataset was slightly skewed by outliers. By the interquartile method, there were 27 outliers, in this case terms with over a 43.7% percent error, which led to a mean percent error of 21.4%. Some of these outliers are due to very small distance flights having a very small margin of percent error; on a shorter flight a small inaccuracy in distance would yield a much larger percent error. Other outliers can be attributed to high velocity or high altitude edge cases that did not get

simulated enough during training. Unfortunately, during validation, the model did not perform very well, which can be attributed to the lack of high altitude and high velocity cases it was trained off of during its construction. The percent error statistics can be seen in the figure below.

	percent_error
count	196.000000
mean	21.369376
std	44.573766
min	0.350742
25%	2.144221
50%	4.389652
75%	18.782175
max	358.619124

Figure 5. Percent error statistics of the ML model.

Overall, the XGBoost model performed well, yielding a relatively low median error. In comparison to the Random Forest Regressor, XGBoost had faster inference and training time**,** produced more accurate predictions, especially in cases involving long-range trajectories or complex input relationships, and required far less estimators and complexity than the Random Forest model.

Part 4: Radar Coverage and Detection

In this part of the project, we aim to assess the real-time detection performance of two ground-based radars situated in Norilsk and Yakutsk against a high-velocity projectile launched from Minot Air Force Base. Building upon the trajectory integration developed in Parts 1, 2, and 3, we incorporate Earth-curvature (horizon) effects to determine exactly when each radar first "sees" the projectile. At that moment, we feed the projectile's instantaneous position and velocity into a trained XGBoost regression model to predict its landing coordinates. Finally, we compare the ML-predicted impact point to the true impact from the numerical solver and visualize the entire scenario—including radar coverage volumes and the flight path—in three dimensions.

We initialize the rocket's propulsion parameters by specifying a constant thrust magnitude along with wet mass, dry mass, and propellant burn-rate. The user-requested burn duration is capped by the available propellant, ensuring the rocket does not consume more mass than it carries. This establishes the time interval during which thrust is applied, after which the vehicle coasts under gravity and aerodynamic drag.

Two radar sites are defined by their geodetic coordinates (latitude and longitude). Each radar is characterized by a maximum slant-range of 500 km, which was increased from the

problem statement to ensure detection, a full 360° azimuthal scanning capability, a 90° elevation field of regard, and an antenna height of 10 m above ground. These parameters together describe a spherical sector volume in which objects may be detected, subject to line-of-sight constraints.

To account for Earth's curvature, we employ the classic geometric horizon formula for both the radar antenna height and the projectile altitude at every time step, seen below:

$$R_h = \sqrt{2R_{Earth}h} + h$$

By summing the horizon distances from each, we calculate the maximum ground-range over which a direct line-of-sight exists. This ensures that the radar will only detect the projectile if both the slant-range is within its maximum range and the ground-range lies below the combined horizon limit. Using the solver developed previously, we simulate the projectile's flight from launch to impact. The initial launch location is set to Minot AFB, and an East-North-Up velocity vector is specified based on the optimization model in Part 2 of the project. The Part 1 solver integrates the equations of motion—including gravitational acceleration, aerodynamic drag dependent on altitude, and Coriolis effects—producing time‑stamped position and velocity vectors in Earth‑Centered Earth‑Fixed coordinates.

For each radar site, we convert the simulated positions into slant-range (straight‑line distance) and ground-range (great‑circle distance) relative to the radar. At every time step, we compare the slant-range against the maximum detection range and the ground-range against the horizon limit. The first time index satisfying both criteria marks the moment of detection. If detection occurs, we record the detection time, corresponding ranges, and the sum of the radar's and projectile's horizon distances. If no detection occurs, we record missing values for these metrics.

Immediately upon detection, we convert the projectile's instantaneous state—latitude, longitude, altitude, and velocity components in local East-North-Up coordinates—into a single‑row feature set. This feature set is passed to a pre-trained XGBoost regression model, which returns predicted landing latitude and longitude. We measure the model's inference latency using high-resolution timing to evaluate whether the prediction can be performed in real time. Predicted coordinates are adjusted for consistency with the geodetic reference frame. All detection and prediction metrics for each radar site—detection flag, detection time, slant and ground ranges, horizon total, ML latency, predicted landing coordinates, as well as radar area, volume, and horizon distance—are assembled into a structured results table. We then compute the true impact location from the final state of the numerical solver and build a concise comparison table showing, for each radar, the ML-predicted versus the actual landing coordinates. Prediction errors (differences in latitude and longitude) are computed to quantify model accuracy.

To provide an intuitive understanding of the scenario, we generate an interactive 3D plot of Earth, the projectile trajectory, radar coverage volumes, and both true and predicted impact points. An equirectangular Earth texture is mapped onto a spherical mesh, then rotated to align longitudes appropriately. The trajectory appears as an orange colored line, with markers indicating launch location, true impact, and ML-predicted impact. Radar coverage is depicted as semi-transparent blue conical sectors emanating from each radar site, highlighting the spatial regions in which detection is possible.

The radar volume and horizon were calculated as 1.308997e17 m^3 and 11294.374m respectively, utilizing a radar sphere of 500 km. The results from the first simulation targeting Novosibirsk (Novosibirsk Oblast), were reported in a PANDAS data frame, and are as follows:

	Radar	Detected	Detect Time (s)	ML Latency (s)	Slant Range (m)	Ground Dist (m)	Horizon Total (m)
0	Norilsk	True	1441.2406	0.0057	499977.114	32712.0224	2582637.947
1	Yakutsk	False	NaN	NaN	NaN	NaN	NaN

Pred Land Lat (deg)	Pred Land Lon (deg)	Radar Area (m^2)	Radar Volume (m^3)	Radar Horizon (m)
67.3674	87.2881	7.853982e+11	1.308997e+17	11294.372
NaN	NaN	7.853982e+11	1.308997e+17	11294.372

```
=== Predicted vs. Actual Landing Locations ===
```

	Radar	Pred Land Lat (deg)	Pred Land Lon (deg)	Actual Lat (deg)	Actual Lon (deg)
0	Norilsk	67.3674	87.2881	55.5864	82.9555
1	Yakutsk	NaN	NaN	55.5864	82.9555

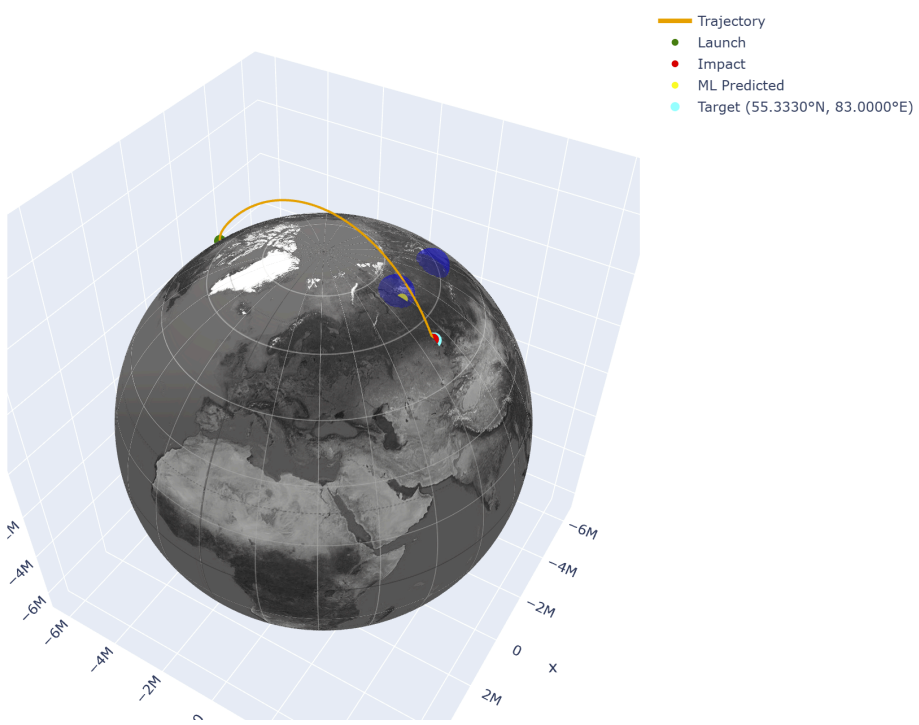

Figure 6. First Radar Predictions

The globe above is the interactive globe model showing the trajectory, launch and landing locations, the ML model prediction, target, and Radar spheres of detection. The LM model was able to predict and classify the projectile 1441.2406 seconds into the flight, in approximately 0.0057 seconds, reporting a slant range of 499977.114m, ground distance of 32712.0224m, and horizon total of 2582637.947m. The ML model prediction had a latitude of 67.3674 degrees, and longitude of 87.2881 degrees. The actual landing location was 55.5864 degrees latitude, and 82.9555 degrees longitude.

Below is the second usage of Part 4, attempting to launch a rocket targeting 39.6N, 124.7E. This used a Venu from part 2 of [-621.78885, 704.3224241379983, 1238.3876894568]. The results from this simulation are as follows:

	Radar	Detected	Detect Time (s)	ML Latency (s)	Slant Range (m)	Ground Dist (m)	Horizon Total (m)
0	Norilsk	False	NaN	NaN	NaN	NaN	NaN
1	Yakutsk	False	NaN	NaN	NaN	NaN	NaN

Pred Land Lat (deg)	Pred Land Lon (deg)	Radar Area (m^2)	Radar Volume (m^3)	Radar Horizon (m)
NaN	NaN	5.026548e+11	6.702064e+16	11294.372
NaN	NaN	5.026548e+11	6.702064e+16	11294.372

```
=== Predicted vs. Actual Landing Locations ===
```

	Radar	Pred Land Lat (deg)	Pred Land Lon (deg)	Actual Lat (deg)	Actual Lon (deg)
0	Norilsk	NaN	NaN	51.477	139.2225
1	Yakutsk	NaN	NaN	51.477	139.2225

```
=== Predicted vs. Actual Landing Locations ===
```

	Radar	Pred Land Lat (deg)	Pred Land Lon (deg)	Actual Lat (deg)	Actual Lon (deg)
0	Norilsk	NaN	NaN	39.5391	125.617
1	Yakutsk	NaN	NaN	39.5391	125.617

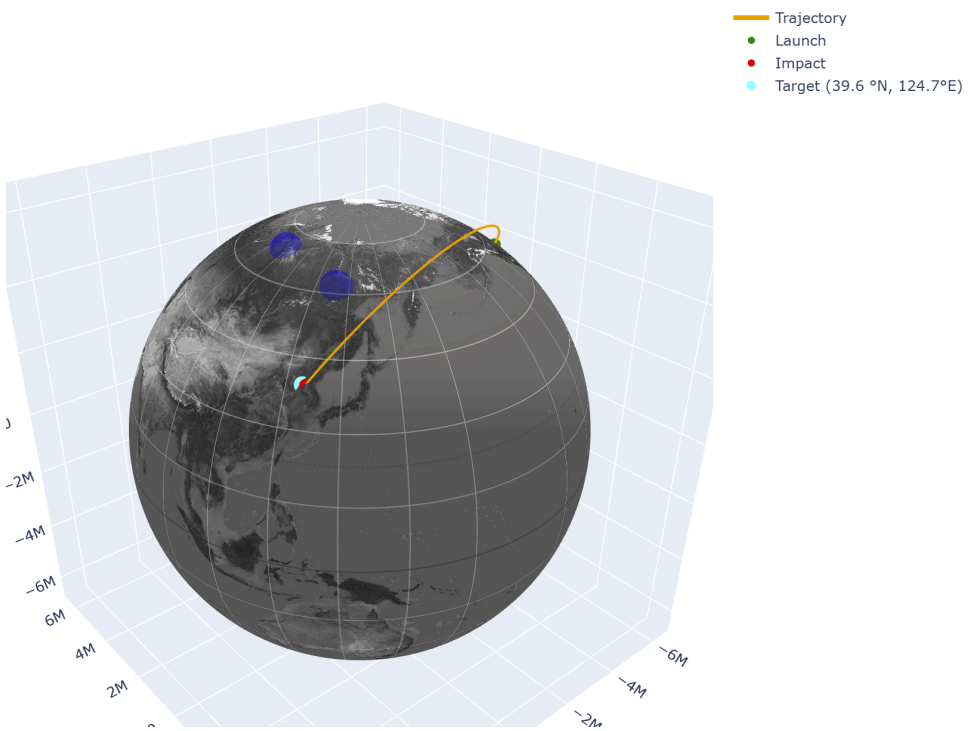

Figure 7. Second Radar Prediction

As seen on the interactive graph above, the rocket was able to again hit its target, but in this case, never passed through a radar detection sphere, allowing it to go on unnoticed to its target. Therefore, the machine learning model was never used to attempt to predict its final location, as no data was available to it. Again, the model successfully plotted the launch and landing locations, the target location, and the trajectory over the surface of the earth.

Appendix

A. Important Functions:

simulate_trajectory:

- Simulates a projectile's trajectory
- Args:
    - lat, lon, h0: floats (geodetic coordinates of launch site) (deg, deg, m)
    - v_enu: list (velocity vector in ENU coordinates) (m/s)
- Returns:
    - sol: (return of solve_ivp of trajectory simulation)

objective_func:

- Objective function for scipy optimization of trajectory parameters
- Args:
    - x: list (v0, theta, psi) (m/s, deg, deg)
- Returns:
    - v0: (initial velocity to be minimized) (m/s)
    - np.inf: only returns if impact not detected

generate_test_data:

- Generates random test data for neural network
- Args:
    - num_samples, seed: ints (number of samples and seed for data generation)
- Returns:
    - data: Pandas dataframe (test data generated)