Techniques for Missile Tracking, Projection and Interception

Cooper Lorsung
Institute of Applied Computational Science
Harvard University
`clorsung@g.harvard.edu`

Sujay Thakur
Institute of Applied Computational Science
Harvard University
`sujay_thakur@g.harvard.edu`

David Zheng
Institute of Applied Computational Science
Harvard University
`davidzheng@g.harvard.edu`

December 10, 2019

Abstract

Our group developed a missile tracking, projection and interception simulator. We framed our simulator in the context of an area defence system similar to the Israeli's Iron Dome system. We baselined our tracking model through Lagrange interpolation. We then utilised Kalman filtering to create a more advanced model that accounts for uncertainty between measurements. We implemented our projection model utilising the rocket equation and finite differences approximation of the timestepped missile position data. Finally, We used the projected path to generate targeting data for our intercept missile and dynamically updated the targeting path, post-launch utilising kinematics. Our model was successful at intercepting enemy missiles when the flight path came close to our counter-missile launch point. Our system struggled when the enemy missile had an abnormal flight pattern.

1 Introduction

This section introduces the motivation behind our work, gives a survey of previous work and details our exact objectives.

1.1 Motivation

Rocket attacks are commonly utilised by adversaries in the Middle East against American forces [1]. Thus a system that can effectively track and intercept enemy projectiles could potentially save lives.

Our group is focused on two separate but related problems: missile tracking and missile interception. We are solving these problems in the context of an area defence system. If we can project where a missile will impact then we can evacuate that area to prevent casualties. If we can intercept the enemy missile then we can eliminate the threat altogether.

1.2 Previous Work in this Area

The Israeli's Iron Dome system is one of the most advanced counter-projectile defence systems in the world[2]. Israel stated the system scored a 90 percent interception rate of Palestinian rockets in 2014 [3]. The system's effectiveness has led the United States military to purchase two Iron Dome weapons systems this year [4].

Figure 1: Iron Dome in action

1.2.1 Overview of Iron Dome system [4]

1. The system utilises radar to detect an incoming projectile.
2. Radar tracks the projectile while alerting the other system components:
   - The battle management and weapons control component
   - The counter-projectile launcher
3. Estimates where incoming projectiles will hit
   - Only focuses on those threats that will fall in the area the system is meant to protect.
4. Targets projectiles predicted to land in the protected zone
5. Launches intercept missile once enemy projectile is within range

1.3 Project Objective

We aim to create a counter missile simulator that follows a similar procedure compared to the Iron Dome:

1. Projectile tracking

2. Projectile path projection

3. Counter-projectile targeting, launch and interception

We will apply multiple numerical methodologies we learned in AM205 in addition to some advanced methods suggested during our group meeting with Professor Rycroft.

2 Methodology

We will discuss the various tracking, projection and interception methods we employed. We needed methods for tracking, projecting, and intercepting the enemy missile. For tracking, we first used perfect tracking. That is, we knew the exact position of the enemy missile at each point in time after it had been detected. A Kalman Filter was used to work with noisy, incomplete tracking data. After tracking the missile, we projected its movement forward in time. This is done so that the response missile is not limited to being behind the enemy missile. We used a combination of finite-differences and kinematics to project enemy missile flight. Lastly, interception was done using an algorithm we developed to select an aim point based on current distance to the enemy missile, and the enemy missile's projected position.

2.1 Tracking

2.1.1 Lagrange Interpolation

The baseline model for missile tracking uses the simple data fitting technique of Lagrange interpolation[5], which fits a polynomial to the data. Assuming we have the control points $\{(x_k, y_k)\}_{k=0}^{n}$ which correspond to sensor measurements, this uses Lagrange polynomials of the form

$$L_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j} \tag{1}$$

with the interpolant being

$$p_n(x) = \sum_{k=0}^{n} y_k L_k(x) \tag{2}$$

Note that the form of the Lagrange polynomial results in

$$L_k(x_i) = \begin{cases} 0, & i \neq k, \\ 1, & i = k. \end{cases} \tag{3}$$

which means that the interpolation matrix is just identity, with a condition number of 1. This allows for stable data fitting which is less sensitive to perturbations.

2.1.2 Kalman Filter

Kalman filtering is an algorithm that allows for inference of unknown state variables based on a series of potentially incomplete and noisy measurements over time [6]. It incorporates the state dynamics and measurements made to compute an entire probability distribution over the states at each time step [7][8]. This can be done in a completely online fashion, allowing for real-time state updates as measurements are made. We assume Gaussian state and noise distributions, such that we can derive analytical forms for the posteriors. For our application, the state vector is:

$$\mathbf{x} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{pmatrix} \tag{4}$$

where p denotes position and v denotes velocity of the enemy missile. At each time step k, the state estimate follows a $\mathcal{N}\left(\hat{\mathbf{x}}_k, \mathbf{P_k}\right)$ distribution. We assume that the kinematics follow the particular form:

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2 \\ v_k &= v_{k-1} + a\Delta t \end{aligned} \tag{5}$$

Using this, the state transitions are captured with the matrix $\mathbf{F}$, which in our case is:

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

The external influence is captured with the control matrix $\mathbf{B}$ and the control vector $\mathbf{u}$, which in our case is:

$$\mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2}\Delta t^2 \\ 0 \\ 0 \\ \Delta t \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} -g \end{pmatrix} \tag{7}$$

We note that there are also forces we may not be able to accurately account for, e.g. drag. We could model the uncertainty that these forces add using Gaussian noise $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. Since our estimation distributions are Gaussians, it can be shown that

$$\begin{aligned} \hat{\mathbf{x}}_k &= \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u} \\ \hat{\mathbf{P}}_k &= \mathbf{F}\hat{\mathbf{P}}_{k-1}\mathbf{F}^T + \mathbf{Q} \end{aligned} \tag{8}$$

We can now refine our state estimates using sensor measurements. In the general case, these measurements may not directly correspond to the states of our system, and we could model this transformation by using an observation matrix $\mathbf{H}$. However, since we would be directly measuring position and velocity in our case, the measurements $\mathbf{z}$ are in the same units and scale as the states $\mathbf{x}$, and hence we can take $\mathbf{H}$ to be identity. We can also model uncertainty in these measurements by assuming measurement Gaussian noise $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. It can be shown that with measurements available at time step k, our refined state estimates follow a $\mathcal{N}\left(\hat{\mathbf{x}}'_k, \mathbf{P_k}'\right)$ distribution, where

$$\begin{aligned} \hat{\mathbf{x}}'_k &= \hat{\mathbf{x}}_k + \mathbf{K}'(\mathbf{z}_k - \hat{\mathbf{x}}_k) \\ \mathbf{P}'_k &= \mathbf{P}_k - \mathbf{K}'\mathbf{P}_k \\ \mathbf{K}' &= \mathbf{P}_k\left(\mathbf{P}_k + \mathbf{R}\right)^{-1} \end{aligned} \tag{9}$$

Here, $\mathbf{K}'$ is the Kalman gain, which controls the relative importance assigned to sensor measurements. Note that our system does not expect to make measurements for all time steps, since this corresponds more accurately to the real-life scenario of discrete radar measurements. If no measurements are taken for time step k, the Kalman gain $\mathbf{K}' = \mathbf{0}$, since we are relying purely on state updates using the dynamics. Note that since we are getting full joint Gaussian distributions over the states, the covariance matrix $\mathbf{P}'_k$ would correspond to uncertainty at each time step. These uncertainty estimates are useful when combining this model with an interception system, since we could minimise an objective function like

$$\mathcal{R} = \mathcal{L} + \lambda \left\|\mathbf{P}'_k\right\|_F \tag{10}$$

where $\mathcal{L}$ is the objective that the interception system was originally minimising, and $\|\mathbf{P}'_k\|_F$ is an additional penalty term that ensures we only make an interception in regions with low uncertainties. Note that λ would then determine the relative importance of the two terms.

2.2 Projection

2.2.1 Rocket Equation

The rocket equation is a well known problem that can be solved analytically. Seen below, the basic formulation leads to a relatively simple solution.

$$m\frac{dv}{dt} = -c\frac{dm}{dt} \tag{11}$$

has solution $\Delta v = c \ln \frac{m_0}{m_f}$, where m_0 is the initial mass of the missile and fuel, m_f is the final mass of just the missile. In order to make our situation both more interesting, and also analytically impossible, we add the gravitational force, as well as velocity and altitude dependent drag forces. Our rocket equation is now

$$m\frac{d\mathbf{v}}{dt} = \mathbf{F} - c\frac{dm}{dt}\hat{\mathbf{v}} \tag{12}$$

where our force $F = -g\hat{\mathbf{z}} - D\hat{\mathbf{v}}$ Here, the drag force is $D = \frac{1}{2}\rho v^2 AC_D$, where A is the cross-sectional area of the projectile, C_D is the drag coefficient of the projectile, v is the magnitude of the velocity, and rho is the air density. We add the altitude dependence of the drag as $\rho(z) = \rho_0 e^{-z/H}$, where ρ_0 is air density at sea level, and H is the 'scale height' of the atmosphere. For all simulations we ran, $H = 8000$ [9]. With the altitude dependence of drag, we have the following set of independent equations that must be solved numerically:

$$\begin{aligned}
m\frac{dv_x}{dt} &= \left(-\frac{1}{2}\rho_0 e^{-z/H} v^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_x \\
m\frac{dv_y}{dt} &= \left(-\frac{1}{2}\rho_0 e^{-z/H} v^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_y \\
m\frac{dv_z}{dt} &= \left(-mg - \frac{1}{2}\rho_0 e^{-z/H} v^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_z
\end{aligned} \tag{13}$$

We can solve these equations with the forward Euler numerical scheme, where we update the velocity of each component as:

$$\begin{aligned}
v_{x,t+1} &= v_{x,t} + \frac{dt}{m}\left(-\frac{1}{2}\rho_0 e^{-z/H} v_t^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_{x,t} \\
v_{y,t+1} &= v_{y,t} + \frac{dt}{m}\left(-\frac{1}{2}\rho_0 e^{-z/H} v_t^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_{y,t} \\
v_{z,t+1} &= v_{z,t} + \frac{dt}{m}\left(-mg - \frac{1}{2}\rho_0 e^{-z/H} v_t^2 AC_D - c\frac{dm}{dt}\right)\hat{\mathbf{v}}_{z,t}
\end{aligned} \tag{14}$$

The unit vectors $\hat{\mathbf{v}}_x$, $\hat{\mathbf{v}}_y$, and $\hat{\mathbf{v}}_z$ are used to control the direction of the missile launch. This gives us a foundation of simulation with the rocket equation. In order to propagate the missile forward, we need kinematics.

2.2.2 Finite Differences and Kinematics

Kinematics equations are used both to update the positions of the missile during rocket equation simulation, and to project the position of the enemy missile[10]. The kinematics equations of interest are simply $\mathbf{x} = \mathbf{x}_0 + \mathbf{v}t$, and $\mathbf{v} = \mathbf{v}_0 + \mathbf{a}t + \frac{1}{2}\mathbf{j}^2$ and $\mathbf{a} = \mathbf{a}_0 + \mathbf{j}t$, where j is the time derivative of acceleration, jerk.

In the case of missile propagation, we update the velocities according to the rocket equation in the previous section, and update the positions with the kinematics equation above: $\mathbf{x} = \mathbf{x}_0 + \mathbf{v}t$.

Enemy missile projection is harder to simulate. After collecting four data points of the enemy missile's location, we approximate the velocity, acceleration, and jerk using finite differences. With these approximations, we project the enemy missile by using the drag-dependent kinematics equation

$$\mathbf{x} = \mathbf{x}_0 + \frac{\mathbf{v}vt}{g}\left(1 - e^{-gt/vt}\right) \tag{15}$$

where vt is the terminal velocity. After one step of the kinematics equation, we update the velocity according to

$$\mathbf{v} = \mathbf{v}_0 + \mathbf{a}t + \frac{1}{2}\mathbf{j}^2 \tag{16}$$

and lastly update the acceleration according to

$$\mathbf{a} = \mathbf{a}_0 + \mathbf{j}t \tag{17}$$

2.3 Interception

2.3.1 Optimisation-based Framework

The baseline model for missile interception uses a simple optimisation-based approach[11] that would allow the response missile to intercept the enemy missile as early as possible. Formally, we can frame the task as

$$\begin{aligned}
\min_{t,\mathbf{v}'} \quad & \frac{1}{2}t^2 \\
\text{s.t.} \quad & x'_0 + v'_x(t-\delta) = x_0 + v_x t \\
& y'_0 + v'_y(t-\delta) = y_0 + v_y t \\
& z'_0 + v'_z(t-\delta) + \frac{1}{2}g(t-\delta)^2 = \quad z_0 + v_z t + \frac{1}{2}gt^2 \\
& t - \delta \geq 0 \\
& \|\mathbf{v}'\| \leq V
\end{aligned} \tag{18}$$

where (x, y, z) corresponds to the enemy missile and (x', y', z') corresponds to the response missile. Note that δ is the time lag between the enemy launching their missile and us launching the response and V is the maximum speed that the response missile can achieve. The Optimisation formulation is essentially finding the response missile velocity that minimises the time that the enemy missile is allowed to travel for, corresponding to intercepting it before it reaches critical areas. This is carried out subject to the constraints that the response missile hits the enemy missile at time $t \geq \delta$ and has speed $\|\mathbf{v}'\| \leq V$.

2.3.2 Numerical-based Framework

Numerical interception was performed by a method we developed specific to this problem. Due to the solution of the rocket equation with altitude dependent drag being numerical, being unable to fully measure the properties of the enemy missile, and the enemy missile's ability to change directions, we were unable to pose this problem as a simple optiimple Optimisation problem, like in

Section 2.3.1. To get around these issues, we performed a series of data collection, projection, and approximate interception steps. The procedure is outlined in Algorithm 1.

Algorithm 1: Numerical Interception

```
input  : Initial location of response missile
output : Enemy and response trajectories
Detect enemy;
Collect four points from enemy trajectory;
while not intercept do
    Project enemy missile $N$ timesteps forward;
    for $k$ in $[1, ..., N]$ do
        for $t$ in $[1, ..., k]$ do
            Integrate one timestep for response missile;
            Integrate one timestep for enemy missile;
            $d_t$ = Distance between response and enemy missiles;
            if intercept_projection then
                $k_{optimum} = k$;
                Break $k$ loop;
            if $d_t \geq d_{t-1}$ then
                Store $d_{t-1}$;
                Store $k$;
                Break $t$ loop;
        end
    end
    if not intercept_projection then
        $k_{optimum} = k$ corresponding to min $d_{t-1}$;
    Set response missile towards $k_{optimum}$;
    Integrate $T$ timesteps for response missile;
    Collect four points from enemy missile;
end
```

The time between projection-prediction cycles, T, can be adjusted to wait more or fewer timesteps before the next projection-interception cycle. In our simulations we waited 5 timesteps. A collision is

3 Results and Discussion

3.1 Tracking

This section shows and analyses the results of the various missile tracking methods we implemented. Our objective is to project the future path of an enemy projectile utilising Langrange interpolation and Kalman filters. We also aim to model the uncertainty of an incoming projectile between measurements.

3.1.1 Lagrange Interpolation

Figure 2 shows the results of Lagrange interpolation on simulated two-dimensional enemy missile data. Since it performs exact data fitting and goes through all the points, this method can be seen to be very sensitive to the measurement noise $\mathbf{v}_k$, which explains the wild swings observed.

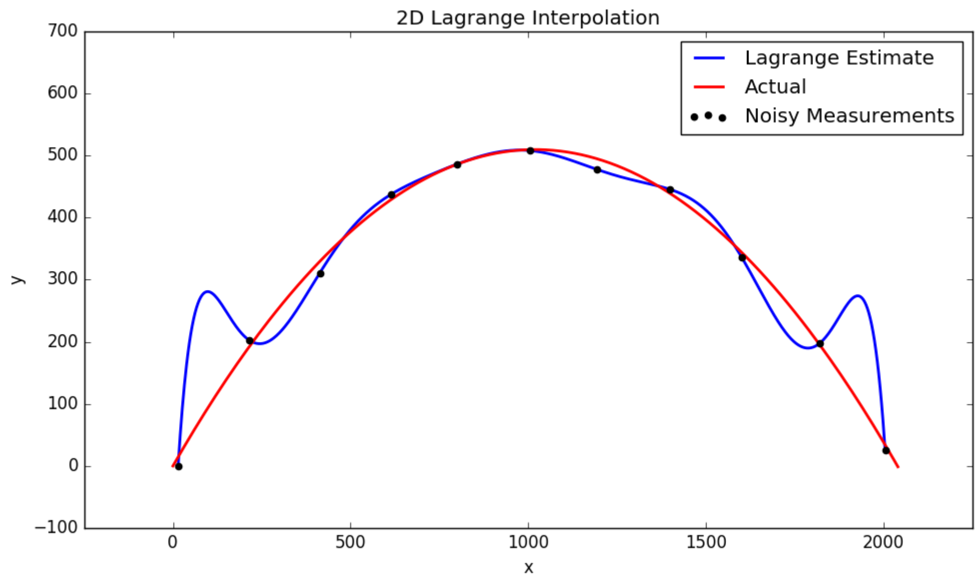

Figure 2: 2D missile data fitting using Lagrange interpolation.

3.1.2 Kalman Filter

Figure 3 shows the results of Kalman filtering on simulated two-dimensional enemy missile data. Since it has information of the actual dynamics and makes weighted state estimates using this, the tracking can be seen to be much more accurate. Additionally, since we obtain an entire joint distribution over the states, we can also get an understanding of the certainty of the estimate. It can be seen that the model is more certain in regions where a measurement was recently made, which is what we would expect. The obtained sensor readings are being used to refine the model's state estimates, as we derived in Section 2.1.2.

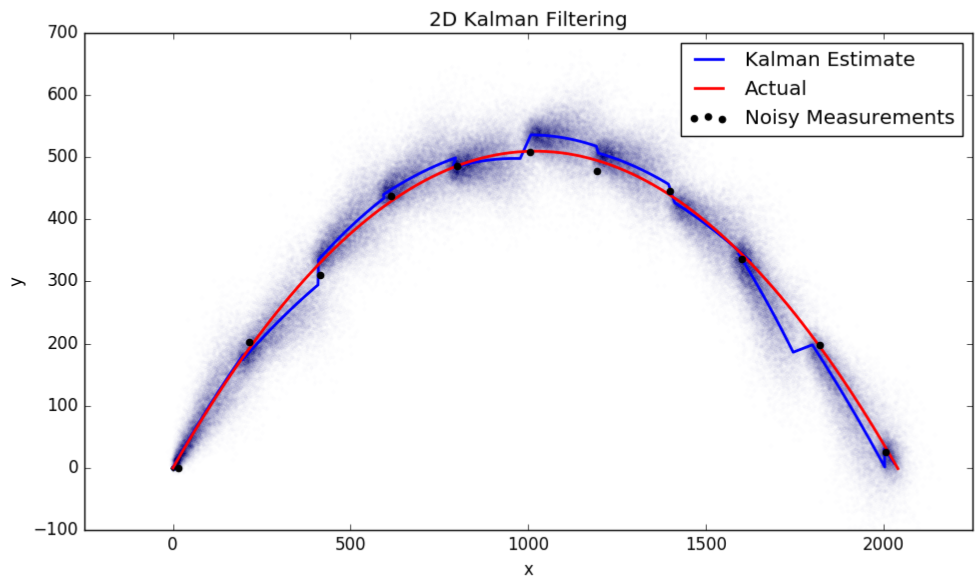

Figure 3: 2D missile data fitting using Kalman filtering, showing uncertainty estimates.

The Kalman filter can also be easily extended to three-dimensional data, as seen in Figure 4. This now corresponds exactly to our missile tracking case.

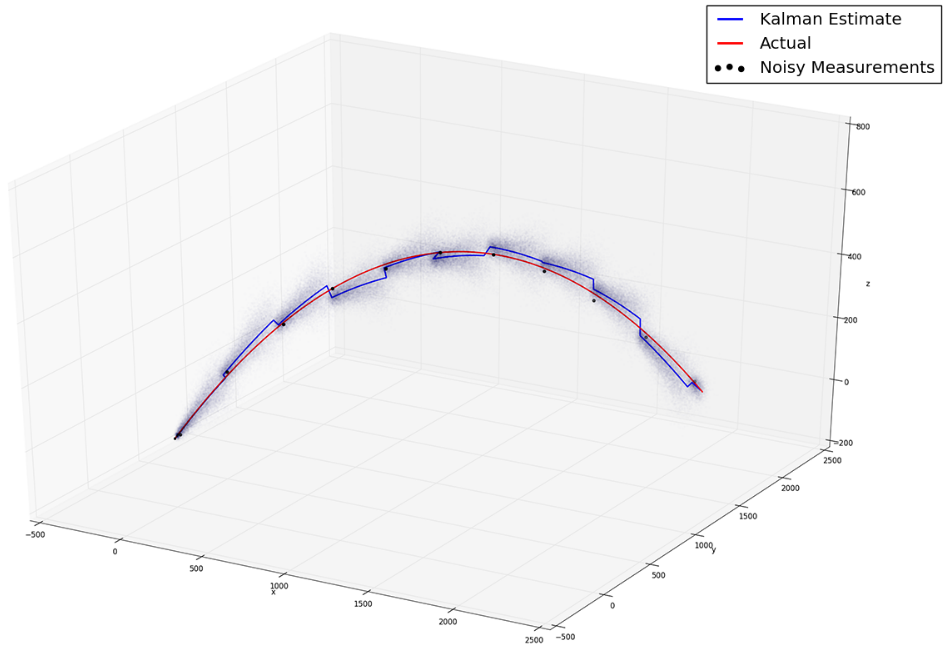

Figure 4: 3D missile data fitting using Kalman filtering, showing uncertainty estimates.

3.1.3 Comparison

We can compare the mean squared error (MSE) of the two techniques against the same simulated ground truth trajectory. These comparisons were carried out for 1000 different trajectories, and the resultant MSEs are shown in Table 1.

Table 1: MSEs of the tracking methods for 1000 different trajectories.

TRACKING METHOD	MSE
LAGRANGE	2617.8 ± 2175.4
KALMAN	224.11 ± 92.29

We can see that, as expected, Kalman filtering provides much better tracking that is also more consistent when repeated for different trajectories. This is because it explicitly uses the actual dynamics as part of its state estimates. Lagrange interpolation, on the other hand, attempts to overfit to all measurements perfectly and hence cannot account for the sensor noise well.

3.2 Forward Projection

Forward projection of the enemy missile is done with simple kinematics. Because the drag force is dependent on many variables we cannot reasonably measure from the ground, we approximate the drag force by setting the terminal velocity in the kinematics equations.

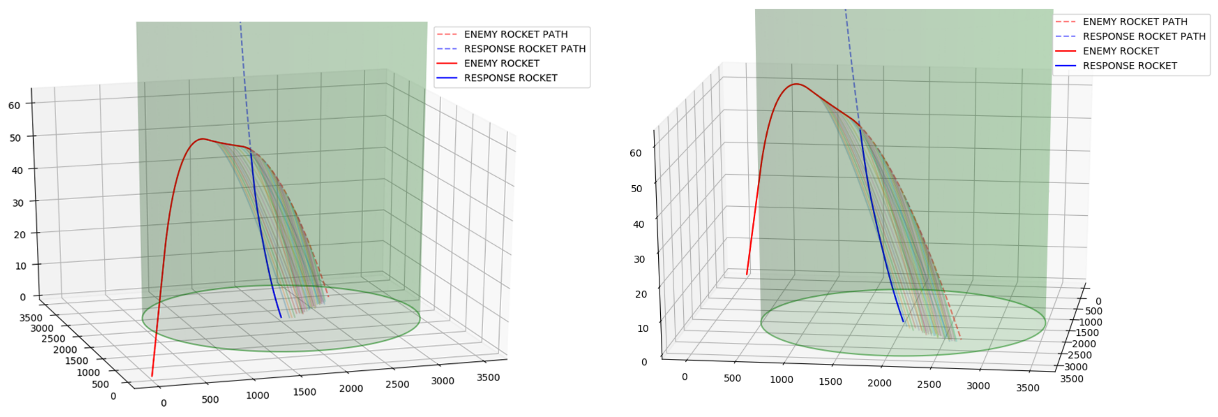

Figure 5: Enemy Missile projected paths and Response Missile intercept

Figure 5 shows the enemy missile and our response missile. The dashed line represents the true path of both missiles if there was no interception. The multiple lines coming off the enemy missile are the projected paths for every 5 time steps after the enemy missile enters our area. The projected paths differ because we programmed the enemy missile to alter its flight path when approaching the targeted area. We assume that we only get information of the enemy missile once it is near enough to our base station, and this is captured by the green cylinder around our response missile.

3.3 Interception

This section shows and analyses the results of the various missile interception methods we implemented. We utilised both Optimisation and numerical frameworks for our counter-projectile missile. This section includes figures that demonstrate the effectiveness of the methods we implemented.

3.3.1 Optimisation-based Framework

The simple optimisation-based baseline interception scheme was implemented on simulated three-dimensional data. Successful interceptions are shown for various parameters in Figure 6.

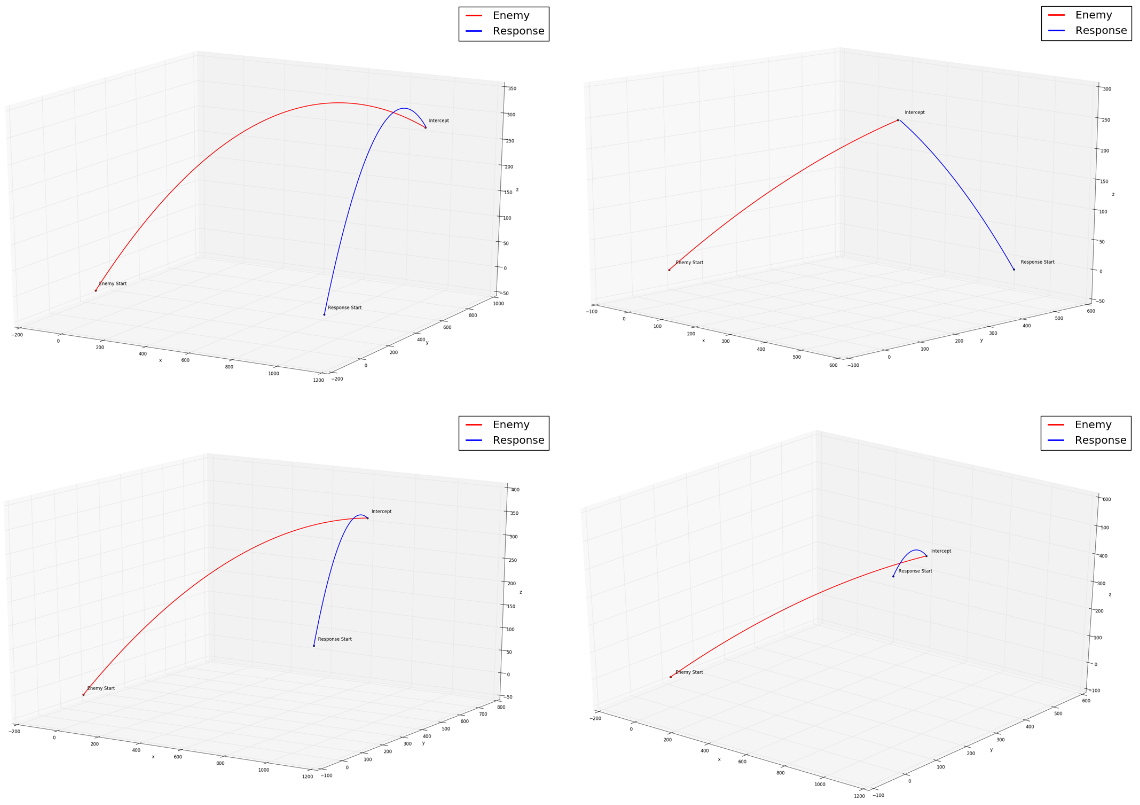

Figure 6: 3D enemy missile (initialised at origin) interception using optimisation-based framework. The response missile was initialised from $(1000, 100, 0)$, $(500, 500, 0)$, $(1000, 100, 150)$, $(1000, 100, 500)$ (top-left to bottom-right) and makes successful interceptions in all cases. Enemy missile had initial velocity $(80, 80, 80)$ while response missile was allowed a maximum speed of 100.

As seen, this performs interceptions well, and is robust to various parameter changes. However, while it is a good baseline model, a severe limitation of this approach is that we need an analytical trajectory expression for both missiles. Hence, it can only be used with relatively simple kinematics, like those shown in Section 2.3.1. If we want to incorporate more realistic trajectories, e.g. ones that involve drag and thrust, the trajectories need to solved numerically, and this approach cannot be used.

3.3.2 Numerical-based Framework

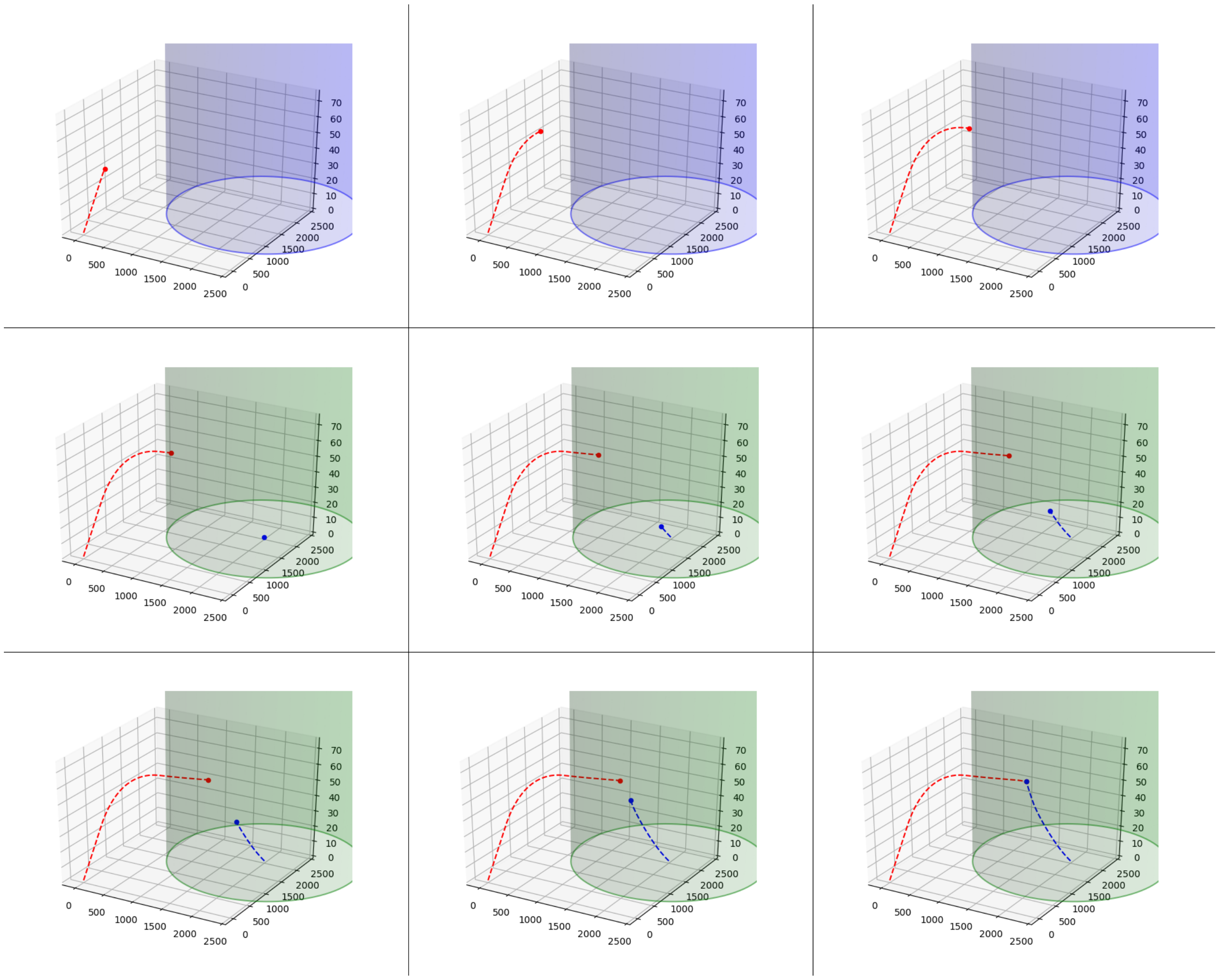

Figure 7: Successful missile interception progression

Figure 7 represent a timeline of the enemy missile's position to the response missile's position. Note that the response missile is not launched until the enemy missile enters our area, indicated by the blue cylinder turning green. Also note how the flight path of the response missile is not linear. This is due to the response missile dynamically adjusting its flight path based on the latest positioning data of the enemy missile. We programmed the enemy missile to adjust its flight path to force our response missile to track it in real-time. The final graph shows a successful interception.

Our interception methodology was very effective at projecting and intercepting enemy missiles that flew close to our counter-missile launch point. However our systems did encounter some struggles when attempting to intercept missiles with abnormal flight paths.

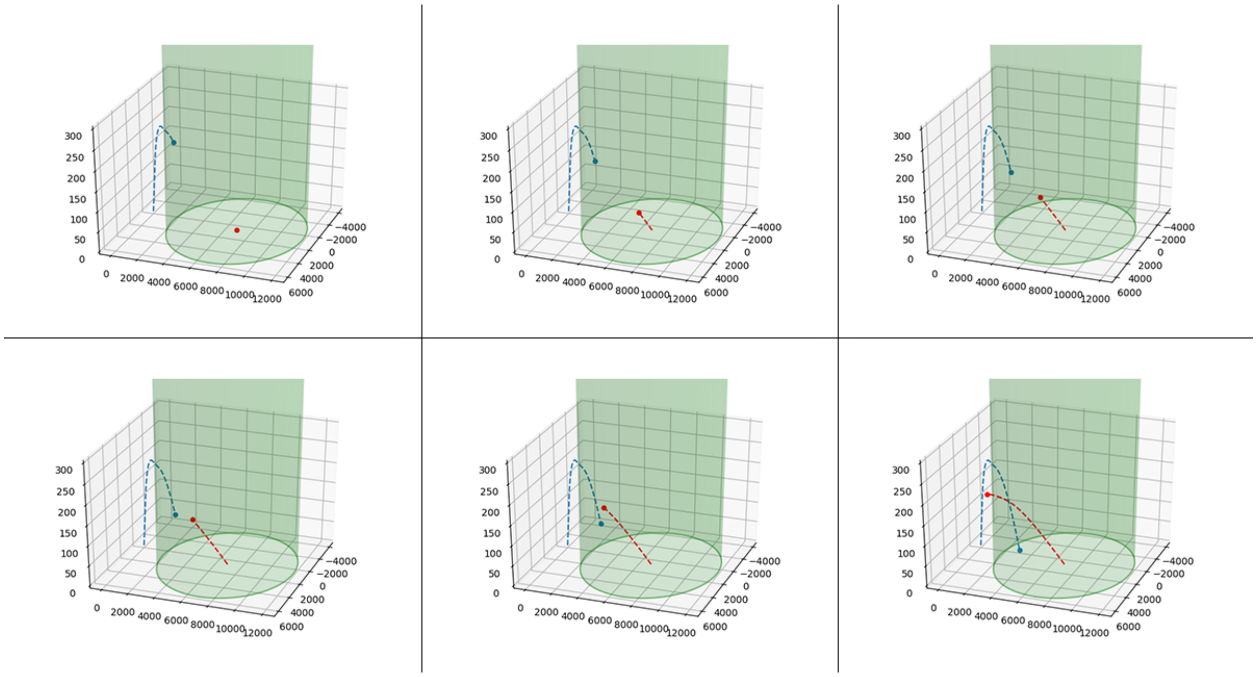

Figure 8: Example of a failed interception.

In cases where the enemy missile flies within the detection, but close to the outer edge, the system struggles with reliable interception. This struggle holds true even with significantly increased response missile speed. Similarly, our system struggles when the enemy missile lands within our detection radius, but close to the enemy launch point. The example seen above has both difficulties, and due to this, our response missile was over 400m away at the closest encounter.

4 Conclusion

We reached our goal of successfully projecting the path of an enemy projectile using our own algorithm. Both the Kalman filter and the finite differences/kinematics methodologies generated accurate projection data. Both methods accurately updated the enemy missile's projected path as new position data became availible.

Our interception algorithm has limitations, though. Reliability stands to be improved in all cases of interception, especially the difficult cases. We were able to successfully intercept enemy missiles that flew close to our counter-missile launch position. However we were less successful at intercepting missiles that did not approach our launch position.

4.1 Future Work

Future work can be done to improve reliability of the model. First, the thrust vectoring was done by breaking the total change A more accurate method of projection would significantly improve results, as ours is fairly naive. Additional information about the enemy missile's mass, size, and thrust force would help improve projection accuracy. These could potentially be measured with a more sophisticated detection and tracking system. Additional improvements could be made by firing multiple response missiles, similar to the Iron Dome of Israel. Our response missile tends to miss behind the enemy missile, so an additional response missile biased to aim farther ahead could potentially lead to a higher rate of successful interception. Lastly, our algorithm may not be ideal. Selecting the point of nearest miss may not be ideal, especially considering our projection does not model the enemy missile well after a few timesteps. To better understand these issues, and give insight into potential solutions, establishing thresholds for enemy missile position, direction, velocity, and trajectory would help greatly.

References

[1] "Rockets target iraqi base that houses american troops, officials say," 2019 (accessed December 8, 2019). [Online]. Available: https://www.cbsnews.com/news/rockets-target-iraqi-base-that-houses-american-troops-officials-say/

[2] M. J. Armstrong, "Modeling short-range ballistic missile defense and israel's iron dome system," *Operations Research*, vol. 62, no. 5, pp. 1028–1039, 2014.

[3] D. Williams, "Rockets target iraqi base that houses american troops, officials say," 2014 (accessed December 8, 2019). [Online]. Available: https://www.reuters.com/article/us-palestinians-israel-irondome/israel-says-iron-dome-scores-90-percent-rocket-interception-rate-idUSKBN0FF0XA20140710

[4] "Israel says iron dome scores 90 percent rocket interception rate," 2019 (accessed December 8, 2019). [Online]. Available: https://www.businessinsider.com/how-the-uss-new-iron-dome-missile-defense-system-works-2019-8

[5] M. T. Heath, *Scientific computing: an introductory survey*. SIAM, 2018, vol. 80.

[6] P. Zarchan and H. Musoff, *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, Inc., 2013.

[7] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed kalman filtering and smoothing," *IEEE Transactions on automatic control*, vol. 55, no. 9, pp. 2069–2084, 2010.

[8] B. Burchett and M. Costello, "Specialized kalman filtering for guided projectiles," in *39th Aerospace Sciences Meeting and Exhibit*, 2001, p. 1120.

[9] J. Peraire and S. Widnall, "Lecture l14 - variable mass systems: The rocket equation," 2008. [Online]. Available: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec14.pdf

[10] N. A. Shneydor, *Missile guidance and pursuit: kinematics, dynamics and control*. Elsevier, 1998.

[11] D. F. Lawden, "Analytical techniques for the optimisation of rocket trajectories," *The Aeronautical Quarterly*, vol. 14, no. 2, pp. 105–124, 1963.