Missile Guidance in the Presence of Constant Error

Stephen Huan, in collaboration with Sergey Blinov

September 28, 2021

1 Problem Statement

A missile is heading towards the origin at a constant speed, however, its guidance system has an error — it is always deflected by an angle of α. How long will it take to reach the origin, compared to the straight-line path? See Figure 1 for a visual depiction.

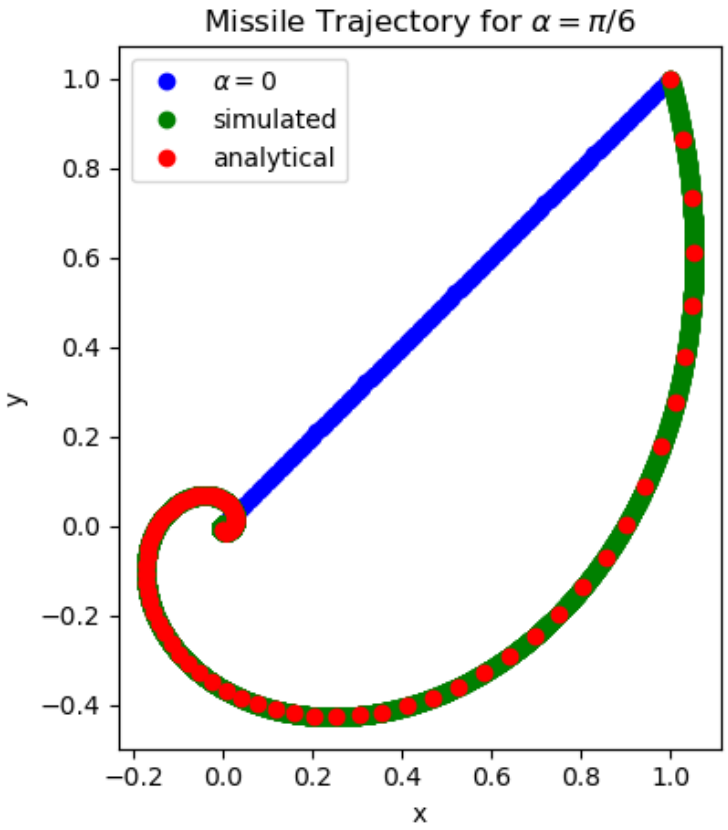

Figure 1: Missile path for $\alpha = \frac{\pi}{6}$

2 Solution

Let $\vec{r}$ be a vector representing the current position of the missile. Normally we would move in the direction of $-\vec{r}$, or the vector pointing towards the origin. However, we're shifted by the angle of α, so we multiply $-\vec{r}$ by the rotation matrix T:

$$T(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \sin\alpha \end{bmatrix}$$

T can be derived through standard linear algebra, see subsection 3.1 of the appendix.

2.1 Differential Equation Setup

So if our current position is $\vec{r}_n$ at index n, we have the following recurrence for the next position over an infinitesimal step size:

$$\vec{r}_{n+1} = \vec{r}_n + T(\alpha)(-\frac{\vec{r}_n}{\|\vec{r}_n\|})\Delta s \tag{1}$$

where we normalize the direction by dividing by its magnitude, and step Δs, the amount we move the direction. Since we assume the direction doesn't change over our small enough step, then Δs is exactly the distance we move along the curve, or the arc length of the curve. Subtracting $\vec{r}_n$ from both sides and dividing by Δs,

$$\frac{\vec{r}_{n+1} - \vec{r}_n}{\Delta s} = T(\alpha)(-\frac{\vec{r}_n}{\|\vec{r}_n\|})$$

In the limit as Δs tends towards 0, then this becomes a differential equation:

$$\frac{d\vec{r}}{ds} = -T(\alpha)(\frac{\vec{r}}{\|\vec{r}\|}) \tag{2}$$

However, equation (2), the differential equation parameterized in terms of arc length, is difficult to solve, since we're dividing by a highly nonlinear term. Instead, it would be nice if the equation looked something like this, without the magnitude term:

$$\frac{d\vec{r}}{dt} = -T(\alpha)\vec{r} \tag{3}$$

Luckily, I've hinted how we can use equation (3) without $\|\vec{r}\|$: it simply represents a different *parameterization* of the same underlying curve, in particular in "time" rather than in arc length. We know the arc length is the sum of changes over time:

$$s(t) = \int_0^t \left\|\vec{r}'(\tau)\right\| d\tau$$

so by taking the derivative, we recover the integrand:

$$\frac{ds}{dt} = \left\|\vec{r}'(t)\right\|$$

We know the what the derivative of $\vec{r}$ is from differential equation (3),

$$= \|-T(\alpha)\vec{r}\|$$

Because a rotation doesn't change the magnitude of the vector, this is simply

$$= \|\vec{r}\|$$

So if we interpret $\vec{r}(t)$ as a different parameterization, namely, $\vec{r}(\int_0^t \|\vec{r}'(\tau)\|\, d\tau)$, then

$$\frac{d\vec{r}}{dt} = \frac{d\vec{r}}{ds}\frac{ds}{dt}$$

Expanding $\frac{d\vec{r}}{ds}$ from differential equation (2),

$$\begin{aligned} &= -T(\alpha)\left(\frac{\vec{r}(s)}{\|\vec{r}(s)\|}\right)\|\vec{r}(s(t))\| \\ &= -T(\alpha)\vec{r}(s(t)) \\ &= -T(\alpha)\vec{r}(t) \end{aligned}$$

The interesting thing is that the differential equation does not care which parameter we access $\vec{r}$ from, but only its *actual position*. Obviously $\vec{r}(s)$ and $\vec{r}(t)$ are equivalent if they give the same vector $\vec{r}$, as $\vec{r}(s(t))$ is parameterized in t but through the intermediate s.

From this exercise we realize that the differential equation (3) without $\|\vec{r}\|$ is just a different *parameterization* of the same curve, where t varies from 0 to ∞, as the derivative $\frac{d\vec{r}}{dt}$ becomes progressively smaller, contributing less and less to the arc length.

In general, suppose we have an arbitrary parameterization $\vec{r}(f(t))$ where f must fulfill certain properties: it must be monotonically increasing so that increasing t goes in the same direction as the original curve, and so that f is bijective, i.e. we transition continuously and without overlap, so $f'(t) > 0$. In that case, by chain rule,

$$\frac{d\vec{r}(f(t))}{dt} = f'(t)\vec{r}'(f(t))$$

Expanding from the differential equation (3),

$$= f'(t)[-T(\alpha)]\vec{r}(f(t))$$

So multiplying by an everywhere positive arbitrary scalar function $f'(t)$ only changes the *parameterization* of $\vec{r}$, and the specific parameterization can be recovered with $\vec{r}(\int_0^t f'(\tau)\, d\tau) = \vec{r}(f(t))$. If we think from the perspective of computational simulation, this is adaptively changing the *step size*. Assuming we update with the discrete update (1), then the precision of our updates is how small Δs is. If we multiply the right side of the differential equation by some factor, then that is equivalent to multiplying Δs by the factor, making the step size larger or smaller, which does not change the actual shape of the curve. In practice, if Δs is too big, then the process will diverge from the actual answer, but in the limit as Δs approaches 0, all parameterizations will converge.

To summarize, we have argued through the lens of parameterization and step size that instead of solving the differential equation (2), we solve the simpler equation (3):

$$\begin{aligned} \frac{d\vec{r}}{ds} &= -T(\alpha)\left(\frac{\vec{r}}{\|\vec{r}\|}\right) \\ \frac{d\vec{r}}{dt} &= -T(\alpha)\vec{r} \end{aligned}$$

2.2 Solving the Differential Equation

The differential equation is a first-order linear system of equations, so we can solve it straightforwardly with linear algebra like in *Differential Equations and Geosystems.*

If we start at some initial point (x_0, y_0), then we have the initial value problem:

$$\vec{r}(0) = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$
$$\frac{d\vec{r}}{dt} = -T(\alpha)\vec{r}$$

Solving for eigenvalues of $-T(\alpha)$, we use the characteristic equation:

$$\begin{aligned} \det(-T(\alpha) - \lambda I) = \begin{vmatrix} -\cos\alpha - \lambda & \sin\alpha \\ -\sin\alpha & -\cos\alpha - \lambda \end{vmatrix} &= 0 \\ (-\cos\alpha - \lambda)^2 + \sin^2\alpha &= 0 \\ -\cos\alpha - \lambda &= \sqrt{-\sin^2\alpha} \\ \lambda &= -(\cos\alpha \pm i\sin\alpha) \end{aligned}$$

Taking $\lambda = -\cos\alpha + i\sin\alpha$ arbitrarily, we solve for its associated eigenvector:

$$\begin{bmatrix} -i\sin\alpha & \sin\alpha \\ -\sin\alpha & -i\sin\alpha \end{bmatrix} \vec{x} = \vec{0}$$

Letting $\vec{x}_1 = 1$, then

$$\begin{aligned} -i\sin\alpha + \vec{x}_2\sin\alpha &= 0 \\ \vec{x}_2 &= i \\ \vec{x} = \begin{bmatrix} 1 \\ i \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + i\begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned}$$

From here, we can get two linearly independent solutions, as described in subsection 4.1 of *Differential Equations and Geosystems*: if the eigenvalue λ is of the form $a + bi$ and its associated eigenvector is of the form $\vec{u}_1 + i\vec{u}_2$, then the solutions are:

$$\vec{r}_1 = e^{at}[\vec{u}_1\cos bt - \vec{u}_2\sin bt]$$
$$\vec{r}_2 = e^{at}[\vec{u}_2\cos bt + \vec{u}_1\sin bt]$$

Plugging in with our particular eigenvalue and eigenvector, and letting $\gamma = b = \sin\alpha$,

$$\vec{r}_1 = e^{(-\cos\alpha)t}[\begin{bmatrix} 1 \\ 0 \end{bmatrix}\cos\gamma t - \begin{bmatrix} 0 \\ 1 \end{bmatrix}\sin\gamma t]$$
$$\vec{r}_2 = e^{(-\cos\alpha)t}[\begin{bmatrix} 0 \\ 1 \end{bmatrix}\cos\gamma t + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\sin\gamma t]$$

Simplifying,

$$\vec{r}_1 = e^{(-\cos\alpha)t}\begin{bmatrix}\cos\gamma t\\ -\sin\gamma t\end{bmatrix}$$

$$\vec{r}_2 = e^{(-\cos\alpha)t}\begin{bmatrix}\sin\gamma t\\ \cos\gamma t\end{bmatrix}$$

Our general solution is a linear combination of the two:

$$\vec{r}(t) = c_1\vec{r}_1(t) + c_2\vec{r}_2(t)$$

Solving for the coefficients by plugging in the initial condition,

$$\begin{aligned}\vec{r}(0) &= \begin{bmatrix}x_0\\ y_0\end{bmatrix}\\ &= c_1\vec{r}_1(0) + c_2\vec{r}_2(0)\\ &= c_1e^0\begin{bmatrix}1\\ 0\end{bmatrix} + c_2e^0\begin{bmatrix}0\\ 1\end{bmatrix}\\ &= \begin{bmatrix}c_1\\ c_2\end{bmatrix} \implies c_1 = x_0, c_2 = y_0\end{aligned}$$

So our final solution is just:

$$\boxed{\vec{r}(t) = e^{(-\cos\alpha)t}[x_0\begin{bmatrix}\cos\gamma t\\ -\sin\gamma t\end{bmatrix} + y_0\begin{bmatrix}\sin\gamma t\\ \cos\gamma t\end{bmatrix}]} \tag{4}$$

A bit of intuition about this solution. First, we assume $|\alpha| < \frac{\pi}{2}$. If $\alpha = \frac{\pi}{2}$, then the missile moves in a circle, and any greater in absolute value means it will go further and further away instead of going towards the origin. Assuming $|\alpha| < \frac{\pi}{2}$, then $0 < \cos\alpha \leq 1$, so $\cos\alpha$ represents the *decay factor*. $\vec{r}(t)$ is of the form $e^{(-\cos\alpha)t}$ times a vector. We will verify later that the magnitude of the vector is always just the initial distance, $\sqrt{x_0^2 + y_0^2}$. So we get closer to the origin only through the scalar multiple, as $\cos\alpha$ is strictly positive, so as time increases $e^{(-\cos\alpha)t}$ grows exponentially smaller. The larger $\cos\alpha$ is (or the smaller α is), the quicker the term grows smaller, so the faster it decays to the origin.

2.3 Calculating Arc Length

Since we want to know how long the missile takes to reach the origin, we just need to calculate the length of the curve — the missile moves at constant speed, so the time it takes is just the length divided by the speed, v. Using the explicit $\vec{r}(t)$ we computed,

$$s(t) = \int_0^t \left\|\vec{r}'(\tau)\right\| d\tau$$

We know the what the derivative of $\vec{r}$ is from the differential equation,

$$= \int_0^t \left\|-T(\alpha)\vec{r}(\tau)\right\| d\tau$$

Because a rotation doesn't change the magnitude of the vector, this is simply

$$= \int_0^t \|\vec{r}(\tau)\|\, d\tau$$

Computing $\|\vec{r}(\tau)\|$ directly,

$$\begin{aligned}
\|\vec{r}(t)\| &= \left\| e^{(-\cos\alpha)t}[x_0 \begin{bmatrix} \cos\gamma t \\ -\sin\gamma t \end{bmatrix} + y_0 \begin{bmatrix} \sin\gamma t \\ \cos\gamma t \end{bmatrix}] \right\| \\
&= e^{(-\cos\alpha)t} \left\| \begin{bmatrix} x_0\cos\gamma t + y_0\sin\gamma t \\ -x_0\sin\gamma t + y_0\cos\gamma t \end{bmatrix} \right\| \\
&= e^{(-\cos\alpha)t} \sqrt{(x_0\cos\gamma t + y_0\sin\gamma t)^2 + (-x_0\sin\gamma t + y_0\cos\gamma t)^2}
\end{aligned}$$

Because the cross terms cancel,

$$\begin{aligned}
&= e^{(-\cos\alpha)t} \sqrt{x_0^2(\cos^2\gamma t + \sin^2\gamma t) + y_0^2(\cos^2\gamma t + \sin^2\gamma t)} \\
&= e^{(-\cos\alpha)t} \sqrt{x_0^2 + y_0^2}
\end{aligned}$$

Substituting back into $s(t)$, and taking t to infinity to get the entire length of the curve,

$$\begin{aligned}
\lim_{t\to\infty} s(t) &= \lim_{t\to\infty} \int_0^t e^{(-\cos\alpha)\tau} \sqrt{x_0^2 + y_0^2}\, d\tau \\
&= \sqrt{x_0^2 + y_0^2} \lim_{t\to\infty} \int_0^t e^{(-\cos\alpha)\tau}\, d\tau \\
&= \sqrt{x_0^2 + y_0^2} \lim_{t\to\infty} [-\frac{1}{\cos\alpha} e^{(-\cos\alpha)\tau}]_0^t \\
&= \frac{\sqrt{x_0^2 + y_0^2}}{\cos\alpha} \lim_{t\to\infty} [e^{(-\cos\alpha)\tau}]_t^0 \\
&= \frac{\sqrt{x_0^2 + y_0^2}}{\cos\alpha}
\end{aligned}$$

So the time the missile will take to reach the origin at a constant speed of v is just

$$t_{\text{arc}} = \frac{\sqrt{x_0^2 + y_0^2}}{v\cos\alpha}$$

Since the missile starts at the point (x_0, y_0), without the angle, it would take

$$t_{\text{normal}} = \frac{\sqrt{x_0^2 + y_0^2}}{v}$$

We conclude that it would take

$$\frac{t_{\text{arc}}}{t_{\text{normal}}} = \boxed{\frac{1}{\cos\alpha}}$$

times longer than it otherwise would.

3 Appendix

3.1 Rotation Matrices

We know a rotation is a linear transformation, from geometric intuition: rotating a scaled vector is equivalent to scaling the rotated vector, and adding two vectors and then rotating is equivalent to rotating both vectors and then adding. So we can parameterize the transformation as a matrix by just seeing what it does to the basis vectors:

$$T(\begin{bmatrix}1\\0\end{bmatrix}) = \begin{bmatrix}\cos\alpha\\\sin\alpha\end{bmatrix}$$

and for $(0,1)$, we know $(1,0)$ maps to (x,y), so we can map $(0,1)$ to (y,x) and negate an entry to keep the dot product 0 (orthogonal). To determine which entry to negate, if we imagine $\alpha = \frac{\pi}{4}$, then $(0,1)$ will map to negative x and positive y so we map to $(-y,x)$

$$T(\begin{bmatrix}0\\1\end{bmatrix}) = \begin{bmatrix}-\sin\alpha\\\cos\alpha\end{bmatrix}$$

So our matrix T is just

$$\begin{aligned} T &= \begin{bmatrix} T(\begin{bmatrix}1\\0\end{bmatrix}) & T(\begin{bmatrix}0\\1\end{bmatrix}) \end{bmatrix} \\ &= \begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix} \end{aligned}$$

3.1.1 Rotation Matrices as Change of Basis

Another derivation is to see a rotation as a *change of basis*. The idea is that when we do rotations, the coordinate system also rotates, forming a new basis, and the coordinates in our new basis are the same as the coordinates in our old basis, which can be seen in Figure 2. If $\mathcal{B} = \{(1,0),(0,1)\}$ is the standard basis and $\mathcal{B}' = \{(\cos\alpha,\sin\alpha),(-\sin\alpha,\cos\alpha)\}$ is our rotated basis, then we are claiming:

$$\begin{aligned} [\vec{x}]_{\mathcal{B}} &= [T_{\mathcal{B}\to\mathcal{B}}\vec{x}_{\mathcal{B}}]_{\mathcal{B}'} \\ &= (\mathcal{B}')^{-1}T_{\mathcal{B}\to\mathcal{B}}[\vec{x}]_{\mathcal{B}} \end{aligned}$$

If this is true for all $[\vec{x}]_{\mathcal{B}}$, then we know

$$\begin{aligned} (\mathcal{B}')^{-1}T_{\mathcal{B}\to\mathcal{B}} &= I \\ T &= \mathcal{B}' \end{aligned}$$

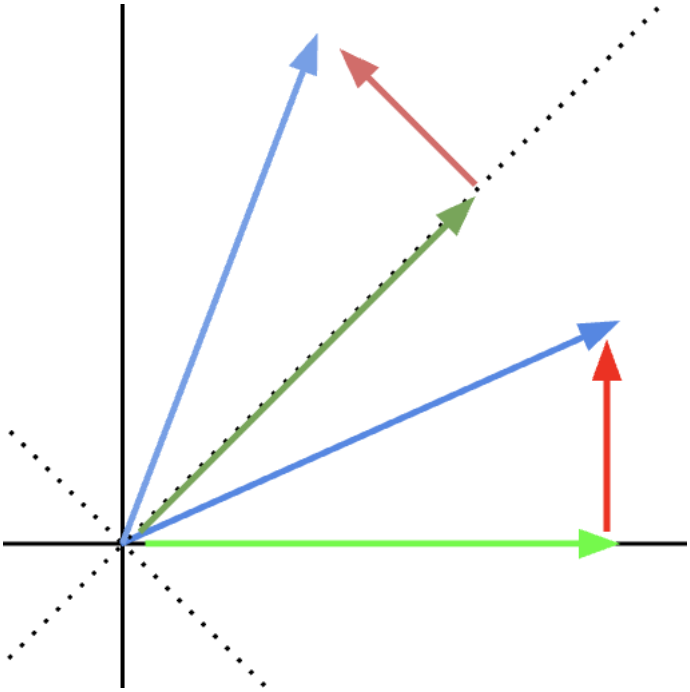

Figure 2: A vector with coordinates in the standard basis, then rotated, has the same coordinates in the rotated basis since the basis vectors rotate with the vector.

4 Code

The code is fairly simple; we start at the initial position and update according to equation (1), arbitrarily taking a (relatively) small step size of 10^{-3}. This yields the same curve as if we multiplied by $\|\vec{r}\|$ or if we used the analytic form in equation (4). To determine when to stop the simulation, we can end after a certain number of iterations or after convergence (when we are a certain ε distance from $\vec{0}$, which is equivalent to checking $\|\vec{r}\| < \varepsilon$). To numerically compute the arc length, we assume our step size is small enough such that when moving to the next point, we assume the curve is perfectly approximated by the line between the next point and the current point, i.e. the direction hasn't changed in between. We simply sum up the lengths of the adjacent pairwise steps $\|\vec{r}_{n+1} - \vec{r}_n\|$. Finally, we record each $\vec{r}_i$ and graph each point in Matplotlib to generate Figure 1.

simulation code

```
from typing import Callable
import numpy as np
import matplotlib.pyplot as plt

P0 = np.array([1, 1])  # initial position
ALPHA = np.deg2rad(60) # angle
VEL = 1e-3             # velocity
MAX_ITERS = 10**5      # maximum number of iterations
EPSILON = 1E-3         # distance until convergence

# rotation matrix
T = np.array([[np.cos(ALPHA), -np.sin(ALPHA)],
              [np.sin(ALPHA),  np.cos(ALPHA)]])

def update1(r: np.array) -> np.array:
    """ Updates the position with a normalized velocity. """
    return r + -VEL*T@(r/np.linalg.norm(r))

def update2(r: np.array) -> np.array:
    """ Updates the position with proportional velocity. """
    return r + -VEL*T@r

def curve(t: float) -> np.array:
    """ Analytical parametrization of the missile's path. """
    gamma = np.sin(ALPHA)
    x = P0[0]*np.array([np.cos(gamma*t), -np.sin(gamma*t)])
    y = P0[1]*np.array([np.sin(gamma*t),  np.cos(gamma*t)])
    return np.exp(-t*np.cos(ALPHA))*(x + y)

def simulate(update: Callable[[np.array], np.array],
             criterion: str="iterations", max_iters: int=MAX_ITERS) -> tuple:
    """ Simulates the missile's path by discrete sampling. """
    iters = lambda t, r: t < max_iters
    criteria = {
        "iterations": iters,
        "distance": lambda t, r: np.linalg.norm(r) > EPSILON and iters(t, r),
    }
    criterion = criteria.get(criterion, criteria["iterations"])

    points = [P0]
    arc_length = 0
    t = 0
    while criterion(t, points[-1]):
        points.append(update(points[-1]))
        arc_length += np.linalg.norm(points[-1] - points[-2])
        t += 1
    return tuple(zip(*points)), arc_length, t

def sample(curve: Callable[[float], np.array],
           start: float, stop: float, step: float) -> zip:
    """ Samples a parameterized curve along an interval. """
    return zip(*(curve(t) for t in np.arange(start, stop, step)))
```

```python
if __name__ == '__main__':
    (x1, y1), _, iters = simulate(update1, "iterations", 10000)
    (x2, y2), arc_length, iters = simulate(update2, "distance")
    print(f"arc length/initial distance: {arc_length/np.linalg.norm(P0):.3f}")
    print(f"{' '*18}1/cos({np.rad2deg(ALPHA):.0f}): {1/np.cos(ALPHA):.3f}")

    fig, ax = plt.subplots()
    ax.set_aspect("equal")

    plt.plot(*((np.arange(0, 1, 0.01),)*2), "bo", label=r"$\alpha = 0$")
    # plt.plot(x1, y1, "bo", label="arc length")
    plt.plot(x2, y2, "go", label=r"simulated")
    plt.plot(*sample(curve, 0, 10, 0.1), "ro", label="analytical")

    plt.title(r"Missile Trajectory for $\alpha = \pi/6$")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.legend()
    plt.tight_layout()
    plt.savefig("missile.png")
    # plt.show()
```
