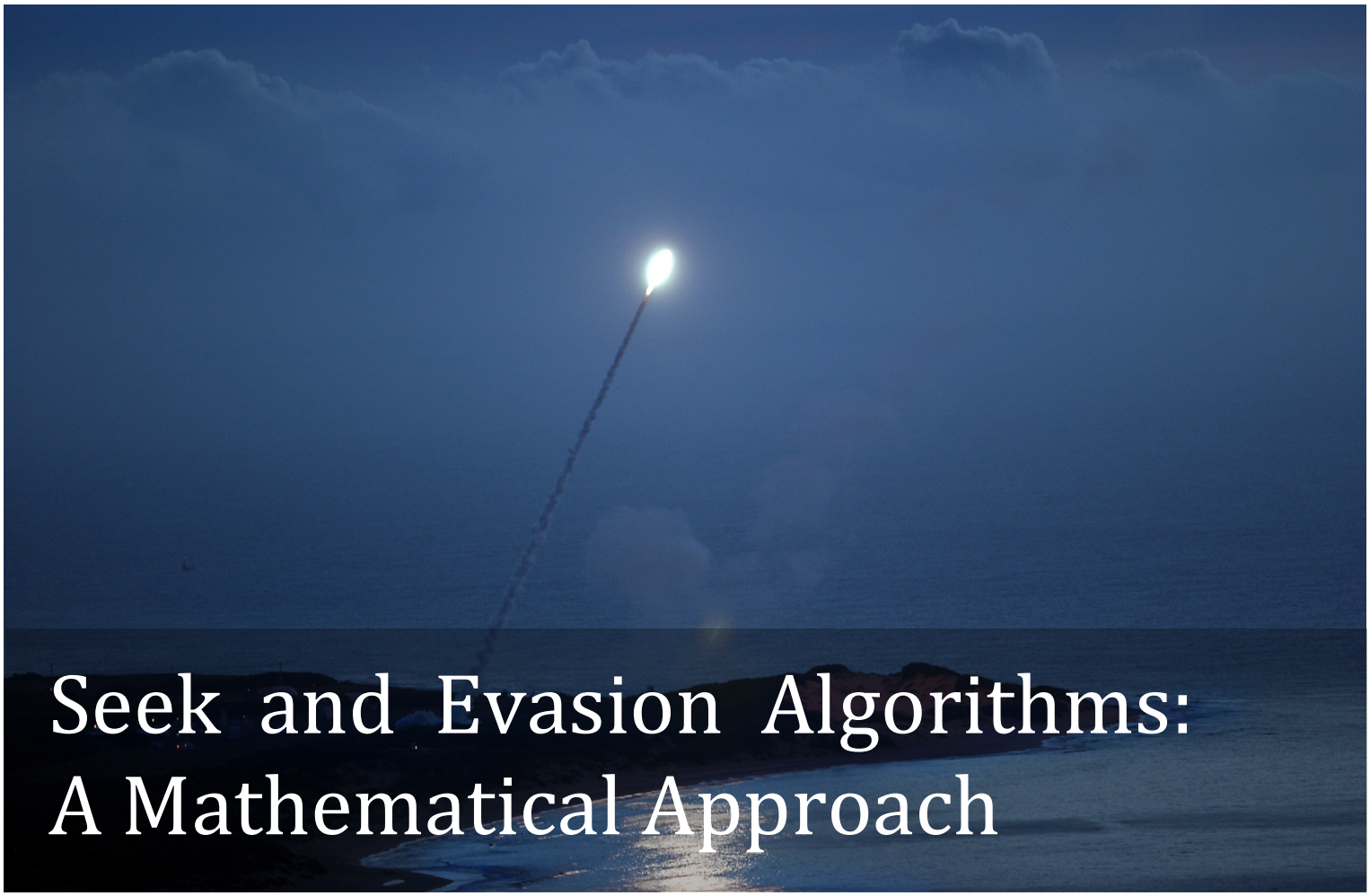


Seek and Evasion Algorithms: A Mathematical Approach

Peter Sharpe, IB Candidate No. 001016-0181

IB HL Mathematics, Instructor Sue Hadden

Internal Assessment

Title photo courtesy of the United States Navy, Photo ID 071106-N-0000X-003. Public domain (A Threat-Representative).

Table of Contents

Rationale

The purpose of this investigation is to determine the most efficient algorithm for a seeker to intercept a target in 2- and 3-dimensional space and the most efficient algorithm for a target to avoid such an interception. This topic caught my attention because of its ubiquity – seek algorithms form a core component of basic behavior modeling and are found naturally in an enormous number of applications. Seek algorithms appear simple, partly because they are very familiar to most people; life demands that we evaluate simple versions of these algorithms every day. ("What is the most efficient way to travel from point A to point B?") However, true seek algorithms involving the kinematic motion of objects require a much higher level of mathematics and are surprisingly complex.

The original impetus for my curiosity was watching a bird-of-prey swoop down on an unsuspecting bird in my backyard. I found it interesting that the attacker was able to intercept the flying bird so gracefully and precisely, and I decided to attempt modeling the human equivalent of avian air-to-air combat – missiles. I first discovered the complexities of seek algorithms when I tried to model the ideal flight paths of birds and missiles. This proved to be extremely challenging, because a successful intercept requires the seeker and the target to share the same x, y, and z position values at the exact same time, when all three position variables and all three respective velocity variables are constantly changing with respect to time. In addition, acceleration magnitudes in all three dimensions are changing as the algorithm manipulates the angle of acceleration. In essence, nine variables must be precisely controlled for an intercept attempt to be successful, while the algorithm is only permitted to manipulate one variable (namely, the seeker's direction of acceleration). Only by solving an extremely complicated series of parametric equations or integrating large nested fractions can one obtain the necessary direction of the seeker's acceleration vector.

In addition to seek algorithms, this investigation will explore another class of algorithms, known as evasion algorithms. Evasion algorithms are the opposite of seek algorithms – instead of manipulating the acceleration of the seeker to intercept a target, the target/evader is given an acceleration or motion vector and attempts to evade interception. Normally, evasion in a two-body system is trivial. (If an intercept is imminent, move. If an intercept is not imminent, don't move.) However, when the other object in an evasion scenario is actively seeking the target/evader, evasion becomes much more difficult because the evader must calculate a specific flight plan that forces the seeker into attempting an impossible intercept. Ideally, the simultaneous use of seek and evasion algorithms will result in a never-ending game of "cat and mouse" between the seeker and the target.

Introduction

Missile seeking uses the same basic mechanics for any number of dimensions, but two-dimensional solutions to the problem have been represented in this investigation because this allows one single angle to represent the direction of the missile's desired acceleration vector. It is possible to directly calculate solutions in three dimensions, but is far more conceptually simple to find a two-dimensional solution first and then extrapolate it to a more applicable three-dimensional solution. Additionally, there are a multitude of real-world problems that involve seek and evasion algorithms that are two-dimensional in practice. For example, ship-to-ship torpedoes and long-range cruise missiles are effectively moving on a two-dimensional plane for the entirety of their "flight" path, and a nap-of-the-earth (extremely low-flying) aircraft moves in a manner where vertical (z-axis) position, velocity, and acceleration are effectively negligible.

Before continuing, it is important to establish a system of variables to represent the position and velocity of both the missile (henceforth the "seeker") and the evader (henceforth the "target"), along with a few other key variables. All position and velocity vectors have been placed on a Cartesian xy coordinate plane and compartmentalized into an (x, y) ordered pair.

Seeker position $= (a, b)$

Target position $= (c, d)$

Seeker velocity $= (e, f)$

Target velocity $= (g, h)$

Magnitude of acceleration available to the seeker $= i, where\ i > 0$

Angle of the desired acceleration/velocity vector, dependent of context $= \theta$ (with domain $0^\circ \leq \theta < 360^\circ$ ($0 \leq \theta < 2\pi$). θ increases in the counterclockwise direction, and the unit vector of direction $\theta = 0^\circ$ is equivalent to $\begin{bmatrix}1\\0\end{bmatrix}$ in the $\begin{bmatrix}x\\y\end{bmatrix}$ Cartesian plane.) θ will be calculated for the specified set of above variables.

Time $= t$, where $t = 0$ represents the time of calculation, $t > 0$ represents points in time in the future, and $t < 0$ represents points in time that have already passed.

It is also important to realize that references to e and i will always refer to the aforementioned variables – their irrational and unreal definitions will never be used in the context of this investigation.

Note that, henceforth, all vectors (whether for position, velocity, or acceleration) represented in the form $\begin{bmatrix} a \\ b \end{bmatrix}$ represent the vector of the indicated property of motion at a units in the x direction and b units in the y direction, unless otherwise specified.

Additionally, one must be aware of the concept known as "timesteps". Often, when conducting physics calculations, time is simulated as a discrete dimension instead of the continuously changing dimension that it actually is. This is for ease of calculation – simulating time as a continuous function requires one to find the integral of velocity with respect to time and the integral of the integral of acceleration with respect to time, which is unnecessarily complex. A timestep is a small unit of time that is used to simulate the motion of objects over discrete time intervals. At each time-step, all position and velocity variables will be updated, and θ will be recalculated for the updated set. While some accuracy will always be lost, this can be controlled and minimized because as the size of the timestep approaches zero, the discrete motion of a simulated object approaches the true continuous motion of the object. In other words, a timestep function is to continuous motion what Riemann sums are to a definite integral. Not all algorithms require the use of a timestep; in this investigation, they are only used whenever θ must be continuously generated throughout the simulation. When used in examples, the interval of the timestep will be indicated with t_{step}.

For each set of initial variables, one distance and one time unit must be selected. Position is expressed in distance units, velocity is expressed as the change in position over each time unit, and acceleration is expressed as the change in velocity over each time unit. All position vectors, velocity vectors, and times are presumed to be expressed in terms of the same units; that is to say that if velocity is expressed in meters per second, position must be expressed in meters, accelerations must be expressed in meters per second per second, and time must be expressed in seconds. However, regardless of whether velocity is expressed in meters per second or miles per hour, the solutions in this investigation will work given that the units are expressed in the same terms.

Finally, it is important to note the nature of motion expressed by the seeker and the target during simulation. The physics calculations in this investigation are premised on the directional authority of missiles and aircraft. Therefore, before delving into the specific physics of the matter, one must realize the fundamental differences between missile (seeker) and aircraft (target) motion in real life. Without including too much technical detail, the primary difference between the two objects comes from differences in *wing loading*, which is an aerodynamic property of objects in fluids expressing the ratio of mass to wing area. Wing loading is proportional to the aerodynamic pressure applied to each unit area of wing during maneuvers. Objects with low wing loading typically have increased aerodynamic attitude (direction) control at the expense of higher induced and form drag. On the contrary, objects with high wing loading have lower drag but fewer capabilities in the way of aerodynamic maneuvering. As a result,

they almost always derive maneuverability from thrust-induced pitching and yawing moments (known as *thrust vectoring*) or a set of motorized spinning flywheels (known as *reaction wheels*), both of which result in the application of a torque about the center of mass. Objects utilizing these methods of attitude control are fortunate in that their attitude authority is not dependent on the direction of relative wind; therefore, for the purposes of this investigation, θ can access its full domain of $0 \leq \theta < 2\pi$. Wing loading tends to be inversely correlated with wing area, and as a result, objects with large wings and fins have much higher wing loading values than their smaller-winged counterparts. Because of this, objects with low wing loading (such as aircraft) are easily able to directly change their direction of velocity, while objects with high wing loading (such as missiles) must change their velocity indirectly by applying thrust to create an acceleration vector in the appropriate direction.

It is important to note that in real life, motion is a continuous and smooth function for any derivative of an object's position function. In other words, position, velocity, acceleration, jerk, jounce, derivative of jounce, and so on can never change instantaneously. However, when simulating motion, there must be a limit on which derivative of position with respect to time one uses. As a result, one parameter of motion must be allowed to change instantaneously. The simplest way to simulate motion is model instantaneous change of position – an instant change in the zeroth derivative of position. This is completely unrealistic, because it is essentially teleportation. A slightly more complex method of simulating motion comes from instantaneous change of velocity – an instant change in the first derivative of position. This is only marginally more accurate, but it is very well suited for objects that tend to retain relatively constant magnitudes of velocity with variable directions, such as cars and aircraft. Because the target is presumed to be an aircraft, this method of motion has been selected to simulate the target. Adding another level of complexity, we develop a system that allows instantaneous change in acceleration – an instant change in the second derivative of position. This method provides a very accurate system for almost all bodies in motion, including ballistic projectiles, rockets, and most bodies where constant force is applied along one or more static vectors. The seeker is modeled after a missile with a constant-thrust rocket engine propelling it, and as a result, it can be simulated with a second-derivative position function.

On top of these functions, the target (aircraft) is given a minimum turning radius and a fixed velocity magnitude, which reflects the limitations of instantaneous velocity control and velocity constraints in aircraft. Similarly, the seeker (missile) is given a fixed acceleration magnitude, which is reflective of the single-ignition engines of many missiles. As a result, changes in aircraft velocity will appear as arcs of a circle, while changes in missile velocity will appear as sections of a parabolic arc.

Aerodynamic factors such as drag are not considered because it is impossible to generalize drag for all shapes and sizes of missiles and aircraft. Gravity is negligible in

missiles because the acceleration imparted by the missile's engine is typically well above 10 times as great as the force of gravity on the rocket. For example, the AIM-9A, one of the most widely-produced missiles of all time, massed 70 kilograms and featured 17.8 kilonewtons of thrust, resulting in a theoretical acceleration of approximately 26 times the magnitude of gravity (Parsch). In this face of this much acceleration, gravity is negligible. In the case of the aircraft, lift from large wings constantly acts against gravity in level flight. The attitude (direction of nose) of the aircraft typically does not differ from the direction of the prograde (direction of motion) vector by more than a few degrees, so the pilot's correction for the influence of gravity is negligible on the aircraft. Even if gravity were implemented into such a simulation, the same gravitational acceleration would be imparted on both the seeker and the target, so their relative position and velocity would not be affected, resulting in identical solutions.

Note that all seeking solutions use an assumption that the target will retain its current velocity. While this seems like an unrealistic assumption, its flaws are negated because seeking solutions are re-evaluated at every time-step. As parameters change in the simulation, the seeker will automatically recalculate intercept solutions. Therefore, the seeker will constantly optimize and correct its flight path to find a firing solution. Although it is true that accuracy is lost due to the assumption that drag does not act on the missile and the aircraft, any errors will constantly be corrected as a result of the recalculation of θ at every timestep. As the time to impact approaches zero, the error caused by gravity also approaches zero. Therefore, it may be more accurate to state that θ is the desired movement/acceleration direction for the individual time step, as θ may change throughout either a simulation or a real life encounter that implements the following algorithms.

Seek Algorithms

Before investigating specific seek algorithms, one must understand the objectives of such an algorithm and why many intuitive approaches do not work.

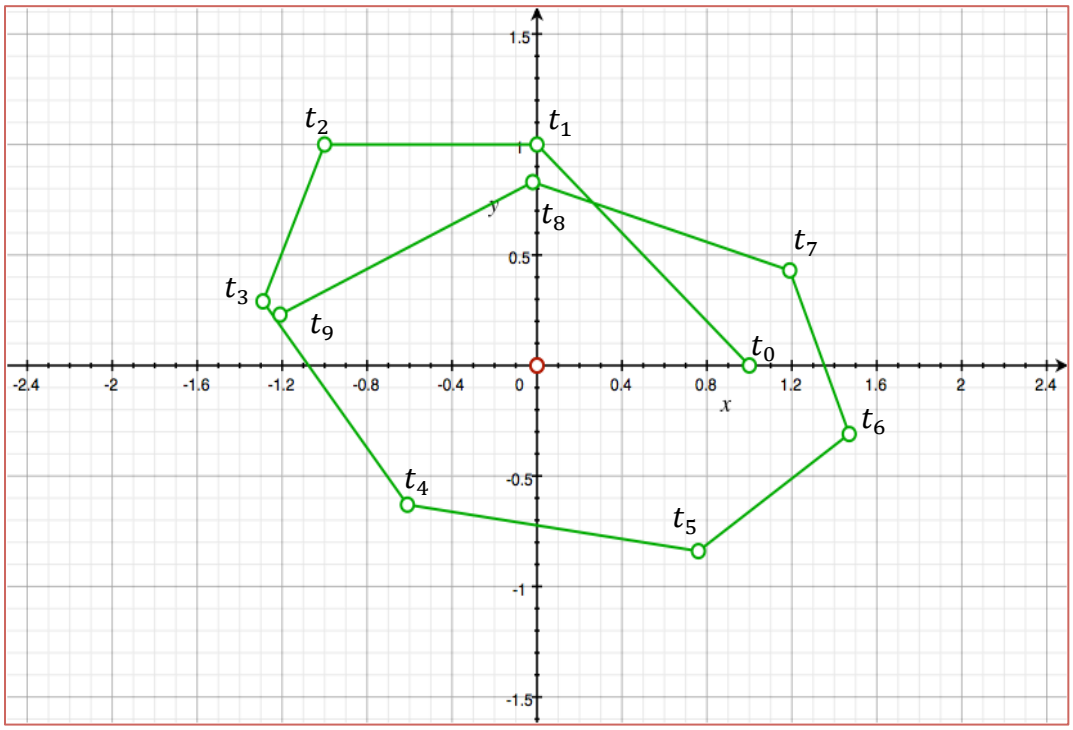
Perhaps the most intuitive solution for seeking a target is to constantly adjust acceleration such that the missile is always accelerating directly at the target. Unfortunately, the only scenario where this approach reliably produces an intercept is when the relative position and relative velocity vectors of the seeker and target are parallel. The utilization of this method almost always result in the seeker undershooting the target, because it does not account for the relative motion of the target. (That is to say, regardless of the velocity of the seeker and the target, acceleration will be applied in the same direction.) To achieve a successful intercept, the seeker must aim for where the target is expected to be, instead of where it currently is. Often, when this direct-acceleration method is employed, the seeker ends up simply orbiting the target as a result of this undershooting. To more easily visualize this concept, picture this scenario:

Let $a = 1, b = 0, c = 0, d = 0, e = 0, f = 1, g = 0, h = 0, i = 1$. That is to say, let the target start at $(0,0)$ with initial velocity $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Let the seeker start at $(1,0)$ with initial velocity $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and an initial acceleration magnitude of 1. Let $t_{step} = 1$. Note that the position of the target will be always be zero, because it starts at the origin and has zero velocity. At $t = 0$, the seeker will be at $(1,0)$. Acceleration will occur in the direction of the target, which is clearly $\theta = 180°$. Given the seeker's acceleration magnitude of 1, acceleration will be at 1 unit towards the origin. Summing the seeker's current velocity vector of $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and the seeker's calculated acceleration vector of $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$ yields the total velocity vector for the initial timestep of $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. The seeker will therefore move by this vector from its current position. The new position will be $(1,0) + \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, or $(0,1)$.

At the next timestep, $t = 1$, the seeker is at $(0,1)$ and will once again accelerate toward the origin with acceleration 1. This results in an acceleration vector of $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$, which, when summed with the current velocity vector of $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$, results in motion along the total velocity vector of $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$. Summed with the current position of $(0,1)$, one obtains the new position of $(-1,1)$.

At $t = 2$, the seeker is at $(-1,1)$, accelerates at $\theta = 315°$, resulting in a total velocity of $\begin{bmatrix} -1 + \sqrt{2}/2 \\ -\sqrt{2}/2 \end{bmatrix}$. The seeker's new position is now $(-2 + \sqrt{2}/2, 1 - \sqrt{2}/2)$.

As the timesteps continue, the seeker continues to circle the target, but never actually hits it. A graph of the first 10 iterations of the algorithm illustrates this:



Here, the seeker and its path are illustrated in green, while the target is shown in red. Clearly, the seeker is not on course to collide with the target. In fact, when this scenario is programmed into a computer and allowed to run for 10,000 iterations, there is not a single point produced that is within 0.5 units of the target. One might be inclined to attribute this to the relatively high timestep that was used, but as t_{step} approaches zero, the path of the seeker simply approaches a circle about the origin with radius 1. This scenario is just one of many that shows the ineffectiveness of direct acceleration towards a target. More evidence can be found in the Conclusion, which compares seek algorithms.

An intuitive solution to the problems of the direct-acceleration method is to split the seeking algorithm into two phases. In the first phase, the seeker accelerated in the direction of the difference between the seeker's velocity vector and the target's velocity vector. In essence, the first phase zeroes the relative velocity of the scenario. In the second phase, the seeker accelerates directly towards the target by finding the difference between the seeker's position vector and the target's position vector. While this method

is always successful, it is a very clumsy solution to the problem as time is wasted in the process of zeroing the relative velocity, especially if the seeker was already on a near-collision course with the target.

The two seek algorithms that I have developed are designed to operate within a single phase to eliminate this wasted time. Both algorithms are based on the process of predicting the future motion of the target and imparting acceleration to the seeker such that they meet the target at the target's projected location in space and time. By using this process, one can elegantly find the necessary θ to meet the target at the intercept location.

Time-Optimal Seek (TOS) Algorithm

I conjecture that for any initial variables from a to i, there exists a constant θ (representing the direction of acceleration) that results in an intercept. Therefore, if acceleration is at a static θ, an intercept will result. This conjecture will be proven later, but it is important to note this truth to understand the approach of this algorithm.

The reason for using a constant θ for the intercept solution is simple: if θ is dynamic, the seeker will almost certainly not accelerate in the same direction for the entirety of its flight. If an algorithm with dynamic θ directs the seeker to accelerate at θ_1 and then θ_2, the total change in velocity for a given time period will be less than if the seeker accelerated at a constant θ at some direction between θ_1 and θ_2.

The problem appears to be extremely difficult, given the number of variables, but it can be simplified by subtracting the respective target position and target velocity from the seeker position and seeker velocity. In essence, we center the coordinate plane about the target's current position and represent the seeker with the seeker's relative position and relative velocity to the target.

Let $j = a - c$ (Seeker's relative x-position to the target)

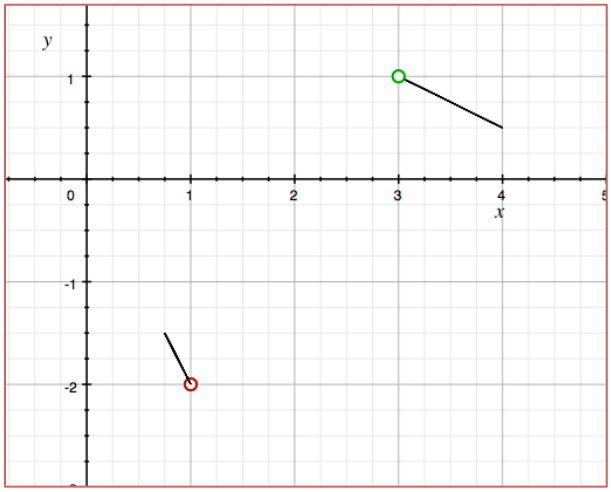
Let $k = b - d$ (Seeker's relative y-position to the target)

Let $l = e - g$ (Seeker's relative x-velocity to the target)

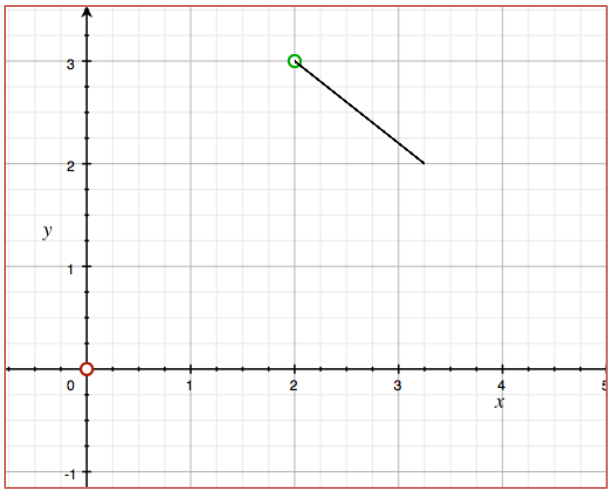
Let $m = f - h$ (Seeker's relative y-velocity to the target)

This position and velocity cancellation is illustrated below, where the green point is the seeker and the red point is the target. The black lines emanating from each point represent the direction and magnitude of its current velocity vector, where the point itself forms the tail end of the vector.

The initial scenario:



Resulting scenario after position and velocity are cancelled:



In this resultant scenario, the seeker is at position (j, k) with velocity (l, m).

To have an intercept, the relative position vector must equal the zero vector at some time t; mathematically, this means that j must equal zero at the exact same time that k equals zero. This can be achieved by compartmentalizing position, velocity, and acceleration into x and y components.

To continue with the solution, one must be familiar with a set of four equations in physics known as the kinematic equations. Together, these equations may be used to calculate unknown variables concerning the motion of objects along one axis given several other known variables. These equations are as follows, where v_i represents initial velocity in the specified direction of motion, v_f represents final velocity in the specified direction of motion, d represents displacement in the specified direction of motion, t represents time, and a represents the acceleration imparted in the specified direction of motion.

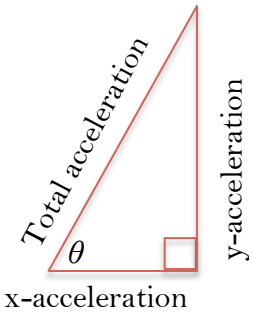
$$d = v_i t + \frac{1}{2} a t^2$$

$$v_f{}^2 = v_i{}^2 + 2ad$$

$$v_f = v_i + at$$

$$d = \frac{v_i + v_f}{2} t$$

In the context of this problem, the first kinematic equation is most useful because it isolates the change in position (displacement) from all the parameters that are known or dynamic in the problem. In the x direction, our desired displacement is $-j$, given that the seeker needs to change its current relative x-position of j to zero. v_i is equal to l, the current relative x-velocity, and t is left as an unknown. To find the acceleration in the x direction, consider the following right triangle and subsequent trigonometric relationships:



$$\sin(\theta) = \frac{y - acceleration}{Total\ acceleration}$$

$$\cos(\theta) = \frac{x - acceleration}{Total\ acceleration}$$

By this logic acceleration in the x direction will equal the product of the total acceleration magnitude i and the cosine of the angle of acceleration, θ. Therefore, $a = i\cos(\theta)$. The equation for motion in the x direction is thus as follows:

$$-j = lt + \frac{1}{2} i\cos(\theta) t^2$$

By similar methods, the equation for motion in the y direction can be constructed, as shown below:

$$-k = mt + \frac{1}{2} i\sin(\theta) t^2$$

These equations are quadratic, a property more visible in the forms below:

$$0 = \frac{1}{2} i\cos(\theta) t^2 + lt + j$$

$$0 = \frac{1}{2} i\sin(\theta) t^2 + mt + k$$

Because there are two variables, t and θ, it is possible to solve the system of equations for both of these variables.

However, while these equations do take on a quadratic form, these equations cannot be solved by the typical means of the quadratic equation because the second-degree term includes a non-constant coefficient. In fact, even computers cannot solve the systems of equations for exact solutions in the current state; to solve this system of equations, the trigonometric functions must first be eliminated and the functions must be combined. There are a few possible ways to achieve this:

1. Taylor series trigonometric approximations
2. Bhaskara I's Sine approximation formula
3. Elimination by the trigonometric identity

In this investigation, only this final method is shown for the sake of brevity. However, the other two methods are capable of producing remarkably accurate approximations when solved for t and θ – depending on the computational aid used to solve the systems of equations, different methods may be appropriate for use.

The third possible method of resolving the trigonometric ratios in the problem involves clever rearranging of the kinematic equations. Recall the system of quadratic functions:

$$0 = \frac{1}{2}icos(\theta)t^2 + lt + j$$

$$0 = \frac{1}{2}isin(\theta)t^2 + mt + k$$

These can be rearranged as such:

$$-lt - j = \frac{1}{2}icos(\theta)\mathrm{t}^2$$

$$-mt - k = \frac{1}{2}isin(\theta)t^2$$

Which allows one to solve for the trigonometric parts of the equations:

$$\cos(\theta) = \frac{-2(lt + j)}{it^2}$$

$$\sin(\theta) = \frac{-2(mt + k)}{it^2}$$

By squaring both sides of both equations, one obtains:

$$\cos^2(\theta) = \frac{4(lt + j)^2}{i^2 t^4}$$

$$\sin^2(\theta) = \frac{4(mt + k)^2}{i^2 t^4}$$

Because of the Pythagorean identity, which states that $\sin^2(\theta) + \cos^2(\theta) = 1$, one can sum the two equations. Note that this results in one equation with one unknown variable (t), which means that it is possible to solve for this variable.

$$1 = \frac{4(lt + j)^2}{i^2 t^4} + \frac{4(mt + k)^2}{i^2 t^4}$$

At this point, it is useful to sum the fractions and expand the numerator of the equation.

$$1 = \frac{4(l^2 t^2 + 2jlt + j^2) + 4(m^2 t^2 + 2kmt + k^2)}{i^2 t^4}$$

Following this, one can multiply both sides by $\frac{i^2 t^4}{4}$.

$$\frac{i^2}{4} t^4 = l^2 t^2 + 2jlt + j^2 + m^2 t^2 + 2kmt + k^2$$

Rearranging and combining like terms yields:

$$0 = -\frac{i^2}{4} t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$$

To solve for t, we must find the roots ("zeroes") of:

$$f(t) = -\frac{i^2}{4} t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$$

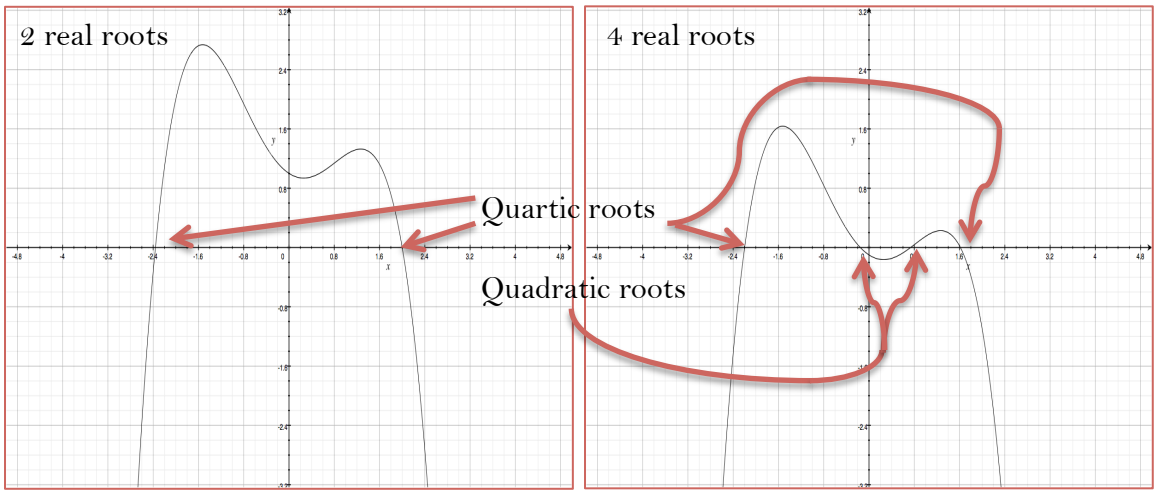
This function is known as a quartic. Quartics are polynomials of the form $f(x) = ax^4 + bx^3 + cx^2 + dx + e$. Quartic equations may have zero, two, or four real roots, before multiplicity is considered. We quickly realize that our quartic must have two or four, because the e term (constant) is always positive (assuming the seeker and the target don't start in the exact same place, which is trivial), while our a term (leading quartic coefficient) is always negative, ensuring that $\lim_{t \to \pm\infty} f(t) = -\infty$. By the intermediate value theorem, which applies to all polynomials, one can conclude that real roots must exist. At least one root must be negative, and at least one root must be positive, since the terminal behavior is negative for positive and negative values of x and $f(0)$ is positive. This is significant because it verifies the initial conjecture of the section: For

any initial variables, there exists a firing solution of constant θ that will result in an intercept.

Next we must prove that there are two roots instead of four roots. First, realize that there is no cubic term in the aforementioned polynomial. Next, think of the quartic as an x^4 term plus a quadratic function. Note that the linear & constant terms are included in the definitions of the quadratic function.

$$f(t) = \left(-\frac{i^2}{4}t^4\right) + \left((l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)\right)$$

As previously stated, the negative x^4 term provides two real roots. If two more real roots are to exist, they must be due to the quadratic function of the equation. Furthermore, because the x^4 term is of a higher degree, the quadratic roots must be within the roots of the quartic. This is shown in the following examples of quartic functions.



For quadratic roots to exist, the quadratic function must be negative at some point in its function. This is because of the positive constant term of the quartic. Keep in mind that the quadratic term (specifically cx^2, not the quadratic function) is always nonnegative because the c value of $(l^2 + m^2)$ is always nonnegative, so any negative values must be the result of the linear term. In addition, the only time when the quadratic term is zero is when l and m both equal zero. When this is the case, the linear coefficient of $(2jl + 2km)$ also equals zero, ensuring that the quadratic function is always nonnegative. Therefore, when proving that only two real roots of the quartic exist, one must only prove for cases where the quadratic coefficient is not zero or negative (i.e. positive).

For reference, the quadratic section is listed below again. Note that, henceforth, $q(t)$ will denote the quadratic section of the quartic.

$$q(t) = (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$$

For the quadratic to have a negative value for any t, the quadratic must equal zero at some point, given that the constant term of $j^2 + k^2$ is positive. As a reminder, when the constant term is zero, the entire problem of seek algorithms is nullified because this requires that j and k equal zero, and thus indicates that the seeker and target are in the exact same location.

Therefore, the quadratic must have at least one root if it is to be negative at some point. We can determine if the quadratic function has one or more roots by completing the square. This is because any function in the form $f(x) = \alpha(x - h)^2 \; \forall \; \alpha, h$ has exactly one root at $x = h$, due to common properties of vertex form. Additionally, a quadratic polynomial is in vertex form with a positive leading coefficient (α), the minimum value of the function is equivalent to the constant term added to the function $f(x) = \alpha(x - h)^2$. To complete the square, we take a generic quadratic to preserve generality:

$$\alpha x^2 + \beta x + \gamma = 0$$

Clearly, $x^2 + \frac{\beta}{\alpha}x + \frac{\gamma}{\alpha} = 0$.

To complete the square, the absolute value of the linear term must equal twice the square root of the constant term, given a quadratic coefficient of 1. It is also possible to complete the square with an equal and opposite linear term. Mathematically, this is expressed as:

$$\frac{\beta}{\alpha} = \pm 2\sqrt{\frac{\gamma}{\alpha}}$$

By multiplying both sides of the equation by α:

$$\beta = \pm 2\alpha\sqrt{\frac{\gamma}{\alpha}}$$

Moving the 2α inside the square root gives:

$$\beta = \pm\sqrt{\frac{4\alpha^2\gamma}{\alpha}}$$

Which simplifies to:

$$\beta = \pm 2\sqrt{\alpha\gamma}$$

Refer to the original $\alpha x^2 + \beta x + \gamma = 0$. Given positive α and γ, which we can assume because the quadratic and constant coefficients of the quartic are both positive in non-trivial cases, β must equal $\pm 2\sqrt{\alpha\gamma}$ to produce an exact minimum of zero. Simple equations of the form reveal that if $\beta \epsilon [-2\sqrt{\alpha\gamma}, 2\sqrt{\alpha\gamma}]$, the minimum of the quadratic is nonnegative.

Now we must apply this to the proof of two quartic roots that is at hand. In this scenario, $\alpha = l^2 + m^2$, $\beta = (2jl + 2km)$, and $\gamma = j^2 + k^2$ where $q(t) = \alpha t^2 + \beta t + \gamma$.

To have a negative minimum, the quadratic section must have the β, or linear, term outside of the range of $[-2\sqrt{\alpha\gamma}, 2\sqrt{\alpha\gamma}]$.

Note that, because the range is between $\pm 2\sqrt{\alpha\gamma}$, we must only concern ourselves with the absolute value of β. Therefore, to complete the proof, one must compare the absolute value of the β term and $2\sqrt{\alpha\gamma}$. In this case, the β term is equal to $2(jl + km)$. The other term, $2\sqrt{\alpha\gamma}$, is as follows due to substitution of α and γ:

$$2\sqrt{\alpha\gamma} = 2\sqrt{(l^2 + m^2)(j^2 + k^2)}$$

By the expansion of the quantity under the radical on the right hand side of the equation:

$$2\sqrt{\alpha\gamma} = 2\sqrt{l^2j^2 + l^2k^2 + m^2j^2 + m^2k^2}$$

Now, the value of the observed β is compared to the value of the β term that is necessary for the quartic to have four roots.

Observed (β)	Necessary term ($2\sqrt{\alpha\gamma}$)
Taken from $q(t)$:	Taken from general quadratic above:
$\beta = 2(jl + km)$	$2\sqrt{\alpha\gamma}$ $= 2\sqrt{l^2j^2 + l^2k^2 + m^2j^2 + m^2k^2}$
Inserting a radical gives:	Simple rearranging gives:
$\beta = 2\sqrt{j^2l^2 + k^2m^2 + 2jlkm}$	$2\sqrt{\alpha\gamma}$ $= 2\sqrt{j^2l^2 + k^2m^2 + l^2k^2 + m^2j^2}$

To compare the values of β and $2\sqrt{\alpha\gamma}$, one can use create an equation with the $\stackrel{?}{=}$ sign to represent two values of unknown relation. After simplification, this symbol will be replaced with an inequality sign or an equality sign.

$\beta \stackrel{?}{=} 2\sqrt{\alpha\gamma}$

Substituted, this yields:

$2\sqrt{j^2l^2 + k^2m^2 + 2jlkm} \stackrel{?}{=} 2\sqrt{j^2l^2 + k^2m^2 + l^2k^2 + m^2j^2}$

Cancellation provides the following statements.

$j^2l^2 + k^2m^2 + 2jlkm \stackrel{?}{=} j^2l^2 + k^2m^2 + l^2k^2 + m^2j^2$

$2jlkm \stackrel{?}{=} l^2k^2 + m^2j^2$

Because neither side was multiplied or divided by a negative number, the sign of inequality of $2jlkm \stackrel{?}{=} l^2k^2 + m^2j^2$ is equal to that of $\beta \stackrel{?}{=} 2\sqrt{\alpha\gamma}$.

Note at this point, that if it is proven that $2jlkm \leq l^2k^2 + m^2j^2 \; \forall \, j, k, l, m$, then $\beta \leq 2\sqrt{\alpha\gamma}$, so the quadratic function has at most one root. If the quadratic function has at most one root, it follows that the quadratic never falls below zero, and as a result, there are only two real roots to the quartic. Let us continue with the inequality:

$2jlkm \stackrel{?}{=} l^2k^2 + m^2j^2$

This is equivalent to the following:

$2jlkm \stackrel{?}{=} l^2k^2 + m^2j^2 - 2jlkm + 2jlkm$

Which can be rewritten as:

$2jlkm \stackrel{?}{=} (lk)^2 - 2jlkm + (mj)^2 + 2jlkm$

The first three terms of the right side can be combined as this is a squared binomial:

$2jlkm \stackrel{?}{=} (lk - mj)^2 + 2jlkm$

Cancellation by subtraction yields:

$0 \stackrel{?}{=} (lk - mj)^2$

The $(lk - mj)^2$ term is always nonnegative, because the minimum possible value of any quantity squared is zero. Therefore, $0 \leq (lk - mj)^2$. Note that, for reasons listed above, the sign of this inequality is equal to that of $\beta \stackrel{?}{=} 2\sqrt{\alpha\gamma}$. Therefore, the quadratic term $q(t)$ is always nonnegative, and as a result, there are only two real roots to the quartic function. We previously established that there is at least one real positive root and one real negative root of the quartic. Since there are only two real roots, one must be positive and the other must be negative.

The negative root can be discarded; it represents an intercept with a time before the simulation began at $t = 0$.

Therefore, for any initial constants, there exists one and only one real positive root of the quartic:

$$f(t) = -\frac{i^2}{4}t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$$

Now we must solve for this real and positive root of t. Unfortunately, this is not as easy as it sounds – solving general quartic equations is notoriously difficult. In fact, it is rumored that when a Spanish mathematician named Valmes claimed to have found a general solution to quartic equations in 1486, he was burned at the stake because the Spanish Inquisition believed that "it was the will of God that such a solution was inaccessible to human understanding." (Beckmann 80)

One method of solving this quartic involves using the quartic formula developed by Lodovico Ferrari in 1540 (Katz 347). (See Appendix A for the full formula.) The quartic formula functions in much the same way that the quadratic formula functions – it requires the values of the coefficients of the polynomial and produces exact solutions for the specified variable. While the quartic formula provides exact solutions, its utility is limited because it involves substituting numbers into an equation of an astonishing 108 terms and nested square- and cubic-root functions. This method is viable, but if exactness is not required, it is often more convenient to solve the quartic in other ways.

Other methods of solving this quartic include the Newton–Raphson method, which iteratively approximates roots for polynomials of any degree. Unfortunately, the iterative nature of the Newton-Raphson algorithm can be computationally expensive as well. While this method is still less tedious than Ferrari's quartic formula, it does sacrifice exactness, which may be valued in certain situations.

Regardless of how you solve the quartic, let t equal your (only) real and positive solution to the quartic.

Recall the earlier system of equations used at the beginning of the time-optimal seek algorithm:

$$\cos(\theta) = \frac{-2(lt + j)}{it^2}$$

$$\sin(\theta) = \frac{-2(mt + k)}{it^2}$$

To find θ, one must take the inverse trigonometric function of one of these two equations that relate θ and t. Because $\mathrm{f(x)} = \arccos(x)$ has a range of $0 \leq f(x) \leq \pi$, it

will be used. This produces a simpler solution, which will be apparent shortly. However, note that it does not actually matter which equation is used.

$$\theta = \arccos\left(\frac{-2(lt+j)}{it^2}\right)$$

Now, if one uses the inverse cosine function, the range is limited to $0 \leq \theta \leq \pi$, which unintentionally eliminates half of all possible angles. To rectify this problem, we must be able to determine whether $\theta \leq \pi$ or $\theta > \pi$. This can be achieved using the sine function. When θ is between 0 and π, inclusive, $\sin(\theta) \geq 0$. When $\pi < \theta < 2\pi$, $\sin(\theta) < 0$. Recall that $\sin(\theta) = \frac{-2(mt+k)}{it^2}$. The denominator is always positive since $i > 0$ and $t^2 \geq 0 \; \forall t$, so the sign of $\sin(\theta)$ is the same as the sign of $-2(mt+k)$. By removing the -2, one can say that $\sin(\theta) \geq 0$ when $mt + k \leq 0$ and $\sin(\theta) < 0$ when $mt + k > 0$.

Therefore, $0 \leq \theta \leq \pi$ when $mt + k \leq 0$ and $\pi < \theta < 2\pi$ when $mt + k > 0$. If $0 \leq \theta \leq \pi$, one can use the inverse cosine function that was previously generated. If $\pi < \theta < 2\pi$, one must subtract this inverse cosine function from 2π because:

$$\cos(\theta) = \cos(-\theta)$$

Because the cosine function has a period of 2π, 2π can be added within the cosine function:

$$\cos(\theta) = \cos(2\pi - \theta)$$

This is used whenever $\pi < \theta < 2\pi$.

Therefore, the solution for the time-optimal seek algorithm is as follows:

If $mt + k \leq 0$, $\theta = \arccos\left(\frac{-2(lt+j)}{it^2}\right)$

If $mt + k > 0$, $\theta = 2\pi - \arccos\left(\frac{-2(lt+j)}{it^2}\right)$

Where $t > 0$ and $0 = -\frac{i^2}{4}t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$.

However, for this to be a true solution, it must only use variables from the original scenario, and $j, k, l,$ and m are variables created for the purposes of the algorithm. Therefore, when using this solution, the following steps must be taken to calculate a solution at any moment:

Let $j = a - c$. Let $k = b - d$. Let $l = e - g$. Let $m = f - h$.

Solve for t, where $t > 0$ and $0 = -\frac{i^2}{4}t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$.

If $mt + k \leq 0$, let $\theta = \arccos\left(\frac{-2(lt+j)}{it^2}\right)$

If $mt + k > 0$, let $\theta = 2\pi - \arccos\left(\frac{-2(lt+j)}{it^2}\right)$

Graphical examples of this algorithm (and all other algorithms) in action can be found in Appendix B.

Simple Seek (SS) Algorithm

While the time-optimal seek algorithm gives the exact answer for the optimal solution every time, it is computationally difficult and is therefore impractical in some applications where computational power is limited and perfect optimization is not required.

This algorithm principally derives itself from the concept of "natural approaches". Given any initial variables where the relative velocity of the seeker and target is nonzero, there will be a time t where they are the closest if the seeker does not accelerate, which will be henceforth referred to as the "natural approach". This may be negative (indicating the natural approach has already happened and the seeker and target are indefinitely moving apart), but it always exists. In essence, this algorithm seeks to find the natural approach of a scenario and then optimize it to make it a true approach. Note that θ is not static throughout the scenario, so it absolutely must be recalculated at every timestep.

Let t_{na} = the time until the natural approach. To find t_{na}, one must calculate the distance between the target and the seeker at time t_{na}. It is worth noting that these calculations assume linear motion on both entities, because it is impossible to know any future changes in motion of the target and because the concept of "natural approaches" implies zero acceleration on the part of the seeker. The target is therefore assumed to continue moving at its current velocity.

One must realize that by finding the product of velocity and time, the total displacement over the time period is found. Thus, the compartmentalized position of the seeker at any given time t is $(a + et, b + ft)$. Similarly, the position of the target at any given time t is $(c + gt, d + ht)$.

Let δ equal the distance between the seeker and the target at time t. Using the equations in the previous paragraph, one can find distance as a function of time, or δ. The following is obtained via the distance formula:

$$\delta = \sqrt{[(c + gt) - (a + et)]^2 + [(d + ht) - (b + ft)]^2}$$

Next, one must minimize δ with respect to t. Note that the square root function passes the vertical line test and is therefore invertible. As a result, $\sqrt{[(c+gt)-(a+et)]^2+[(d+ht)-(b+ft)]^2}$ is minimized whenever $[(c+gt)-(a+et)]^2+[(d+ht)-(b+ft)]^2$ is minimized. For simplicity, let $\Delta = \delta^2$, as such:

$$\Delta = [(c+gt)-(a+et)]^2+[(d+ht)-(b+ft)]^2$$

Next, realize that before the natural approach of a scenario, δ and consequently Δ are continuously decreasing with respect to time as they are approaching each other. After the natural approach, δ and consequently Δ are continuously increasing with respect to time as they drift apart. Thus, the only time when $\frac{d\delta}{dt}$ or $\frac{d\Delta}{dt}$ equal zero is when they are evaluated at the natural approach itself ($t = t_{na}$). It is simpler to find $\frac{d\Delta}{dt}$ than it is to find $\frac{d\delta}{dt}$, so the former will be found.

To find $\frac{d\Delta}{dt}$, simply take the derivative of Δ with respect to time:

$$\Delta = [(c+gt)-(a+et)]^2+[(d+ht)-(b+ft)]^2$$

Simplify:

$$\Delta = [(g-e)t+(c-a)]^2+[(h-f)t+(d-b)]^2$$

Thus, using the chain and power rules of differential calculus:

$$\frac{d\Delta}{dt} = 2[(g-e)t+(c-a)][g-e]+2[(h-f)t+(d-b)][h-f]$$

Now, we must solve for t_{na} by setting $\frac{d\Delta}{dt}$ equal to zero. We may substitute t_{na} for t at this point, because $t_{na} = t$ when $\frac{d\Delta}{dt} = 0$.

$$0 = 2[(g-e)t_{na}+(c-a)][g-e]+2[(h-f)t_{na}+(d-b)][h-f]$$

In the next step, the coefficients of two are divided out in addition to distributing:

$$0 = (g-e)^2 t_{na}+(c-a)(g-e)+(h-f)^2 t_{na}+(d-b)(h-f)$$

Rearranging:

$$-(c-a)(g-e)-(d-b)(h-f) = (g-e)^2 t_{na}+(h-f)^2 t_{na}$$

Multiplying $(c-a)$ and $(d-b)$ by -1 yields $(a-c)$ and $(b-d)$, respectively, so the left side of the equation is rewritten. In addition, like terms can be combined on the right side.

$$(a-c)(g-e)+(b-d)(h-f)=[(g-e)^2+(h-f)^2]t_{na}$$

Finally, both sides can be divided to obtain t_{na}:

$$\frac{(a-c)(g-e)+(b-d)(h-f)}{(g-e)^2+(h-f)^2}=t_{na}$$

Recall that at time t_{na}, the target will be at $(c+gt_{na}, d+ht_{na})$. Because of this, the seeker's necessary motion can be distilled into a few simple requirements:

At $t=0$, the seeker is at (a,b).

At $t=t_{na}$, the seeker must be at $(c+gt_{na}, d+ht_{na})$.

Assuming linear motion on the seeker, we must now determine the average velocity to move from (a,b) to $(c+gt_{na}, d+ht_{na})$ in time t_{na}. This is quite simple, as velocity is simply displacement divided by time.

Therefore, in the x direction, the necessary velocity is the x-displacement of $c+gt_{na}-a$ divided by t_{na}. Similarly, in the y direction, the necessary velocity is the y-displacement of $d+ht_{na}-b$ divided by t_{na}. Written as one vector, the necessary velocity (v) to meet the target at the intercept location is:

$$v=\begin{bmatrix}\frac{c+gt_{na}-a}{t_{na}}\\ \frac{d+ht_{na}-b}{t_{na}}\end{bmatrix}$$

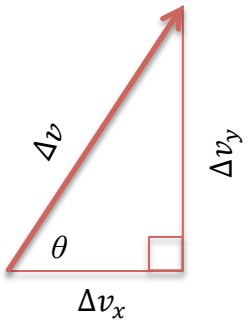
Now, the current velocity of the seeker is $\begin{bmatrix}e\\ f\end{bmatrix}$. The necessary change in velocity (Δv) to intercept at time t_{na} at $(c+gt_{na}, d+ht_{na})$ is:

$$\Delta v=\begin{bmatrix}\frac{c+gt_{na}-a}{t_{na}}-e\\ \frac{d+ht_{na}-b}{t_{na}}-f\end{bmatrix}$$

To simplify this somewhat, a few terms may be cancelled:

$$\Delta v=\begin{bmatrix}\frac{c-a}{t_{na}}+g-e\\ \frac{d-b}{t_{na}}+h-f\end{bmatrix}$$

Therefore, to intercept the target, the seeker should accelerate in the direction (argument) of vector Δv. Consider the following right triangle in order to determine the argument of the vector Δv from its compartmentalized for. Let Δv_x represent the x component of Δv while Δv_y represents the y component of Δv. In other words, $\Delta v_x = \frac{c-a}{t_{na}} + g - e$ and $\Delta v_y = \frac{d-b}{t_{na}} + h - f$.



Using simple right-triangle trigonometry, one can determine that $\tan(\theta) = \frac{\Delta v_y}{\Delta v_x}$. Therefore, $\theta = \arctan\left(\frac{\Delta v_y}{\Delta v_x}\right)$.

When applied to the simple seek algorithm, one realizes that acceleration should be applied at this angle:

$$\theta = \arctan\left(\frac{\frac{d-b}{t_{na}} + h - f}{\frac{c-a}{t_{na}} + g - e}\right)$$

where $t_{na} = \frac{(a-c)(g-e)+(b-d)(h-f)}{(g-e)^2+(h-f)^2}$.

However, there is one problem that stems from the fact that inverse tangent values are not unique within the domain $0 \leq \theta < 2\pi$. This is because the period of tangent is only π, not 2π. To rectify this problem, one must find the value of θ given by the inverse tangent function in the four quadrants. Consider the following:

If vector Δv is in the first quadrant, both Δv_x and Δv_y are positive, so the inverse tangent of $\arctan\left(\frac{\Delta v_y}{\Delta v_x}\right)$ returns the correct value which will be θ given $0 < \theta < \frac{\pi}{2}$.

If vector Δv is in the second quadrant, Δv_x will be negative and Δv_y, so a negative value is passed to the inverse tangent function of $\arctan\left(\frac{\Delta v_y}{\Delta v_x}\right)$. Unfortunately, most computational aids only give results for the inverse tangent function $f(x) = \arctan(x)$

of $-\frac{\pi}{2} < x < \frac{\pi}{2}$. Therefore, performing the inverse tangent function of a Δv in the second quadrant yields results of $-\frac{\pi}{2} < \theta < 0$, which is incorrect. One must realize that tangent is simply equal to $\frac{\sin(x)}{\cos(x)}$, so when given $\frac{\Delta v_y}{\Delta v_x}$ where Δv_x is negative and Δv_y is positive, results will be identical to if both quantities were multiplied by -1. Given a point on the coordinate plane of the form $(-x, y)$, the point $(x, -y)$ will be a point with equal and opposite displacement from the origin. Therefore, with respect to the origin, they are 180° or π apart. Therefore, to find the inverse tangent of a Δv in the second quadrant, one can simply perform:

$$\theta = \arctan\left(\frac{\Delta v_y}{\Delta v_x}\right) + \pi$$

In the third quadrant, a similar problem arises. If Δv is in the third quadrant, both Δv_x and Δv_y are negative, which cancel when divided. The resulting value for θ is in the first quadrant, and, for reasons listed above, is exactly π radians away from the true value of θ. To change the values from $0 < \theta < \frac{\pi}{2}$ to $\pi < \theta < \frac{3\pi}{2}$, we can perform the same operation that was performed to vectors in the second quadrant:

$$\theta = \arctan\left(\frac{\Delta v_y}{\Delta v_x}\right) + \pi$$

Finally, if Δv is in the fourth quadrant, Δv_x is positive and Δv_y is negative. The value of θ that results from the inverse tangent function is correct, but it is in the range $-\frac{\pi}{2} < \theta < 0$, whereas the range of $\frac{3\pi}{2} < \theta < 2\pi$ is desired for consistency and because it is part of the desired domain specified in the introduction to seek algorithms. Noting that the tangent function is periodic over π and therefore 2π, one can simply add 2π:

$$\theta = \arctan\left(\frac{\Delta v_y}{\Delta v_x}\right) + 2\pi$$

To summarize the necessary transformations:

Quadrant of Δv	Necessary Transformation to θ
1st	None.
2nd	Add π to θ.
3rd	Add π to θ.
4th	Add 2π to θ.

To determine the quadrant of Δv, we can analyze the values of Δv_x and Δv_y, which are known. If Δv_x is negative, Δv is in the 2nd or 3rd quadrant, and therefore we must add π to the inverse tangent-derived-θ to find the true value of θ. If Δv_x is positive and Δv_y is negative, we must add 2π to the inverse tangent-derived-θ value. Recall that $\Delta v_x = \frac{c-a}{t_{na}} + g - e$ and $\Delta v_y = \frac{d-b}{t_{na}} + h - f$.

Using techniques similar to those of piecewise functions, the value of θ may be stitched together from the values required in the four quadrants. As a result, the solution for the simple seek algorithm involves the following steps, which must be performed at every timestep since θ is not necessarily static throughout the scenario.

Let $t_{na} = \frac{(a-c)(g-e)+(b-d)(h-f)}{(g-e)^2+(h-f)^2}$.

Let $\theta = \arctan\left(\frac{\frac{d-b}{t_{na}}+h-f}{\frac{c-a}{t_{na}}+g-e}\right)$

If $\frac{c-a}{t_{na}} + g - e < 0$, add π to θ to correct its value.

If $\frac{c-a}{t_{na}} + g - e > 0$ and $\frac{d-b}{t_{na}} + h - f < 0$, add 2π to θ to correct its value.

Evasion (E) Algorithm

In the evasion algorithm, the target/evader is given some constant velocity equal to its initial velocity. This is because much like an airplane, it is able to turn by directly manipulating its velocity vector. The initial velocity, v, is therefore defined as follows, given that g represents the target's x-velocity and that h represents the target's y-velocity:

$$v = \sqrt{g^2 + h^2}$$

When evading a seeker, there are two possible scenarios that are defined on the imminence of an intercept:

1. If a seeker impact is not imminent, the target should move in the opposite direction of the target to maximize the time to impact. ("Passive" evasion)
2. If an impact is imminent, the target should move in a direction perpendicular to the velocity vector of the seeker in a direction away from the seeker. This optimum behavior based on research conducted by the aeronautical think tank Air Power Australia (Kopp). This research makes sense because motion in a direction perpendicular to the seeker's velocity vector requires the seeker to impart the most acceleration to intercept the target. ("Active" evasion)

To create an evasion algorithm that operates in both of these conditions, we must first define imminence in mathematical terms so that the target/evader is able to calculate which course of action to follow.

For the purposes of the development of the evasion algorithm, I have chosen to define imminence on time to impact rather than relative distance. That is to say that a scenario where the seeker and target are very close will not necessarily be defined as imminent, given that they are moving apart at a rapid pace. Because imminence is defined on time to impact, one must derive a formula that represents the time to impact. To this end, let $t =$ the time to impact (or natural approach). Fortunately, this has already been calculated in the seek algorithms that have been previously developed. These solutions for time to impact (or natural approach) have been briefly restated below:

TOS algorithm: Where $j = a - c$, $k = b - d$, $l = e - g$, and $m = f - h$, $t > 0$ and $0 = -\frac{i^2}{4}t^4 + (l^2 + m^2)t^2 + (2jl + 2km)t + (j^2 + k^2)$.

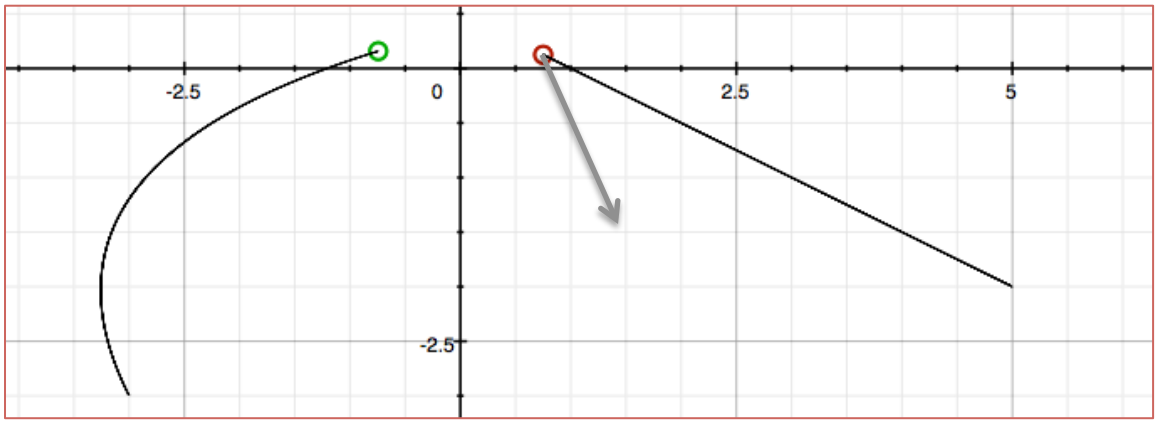
SS algorithm: $t = \frac{(a-c)(g-e)+(b-d)(h-f)}{(g-e)^2+(h-f)^2}$

Given that the time-optimal seek algorithm is a very specific seek algorithm, while the simple seek algorithm objectively represents the time to closest approach, the SS algorithm will be used to estimate the time to impact. An added advantage of using the

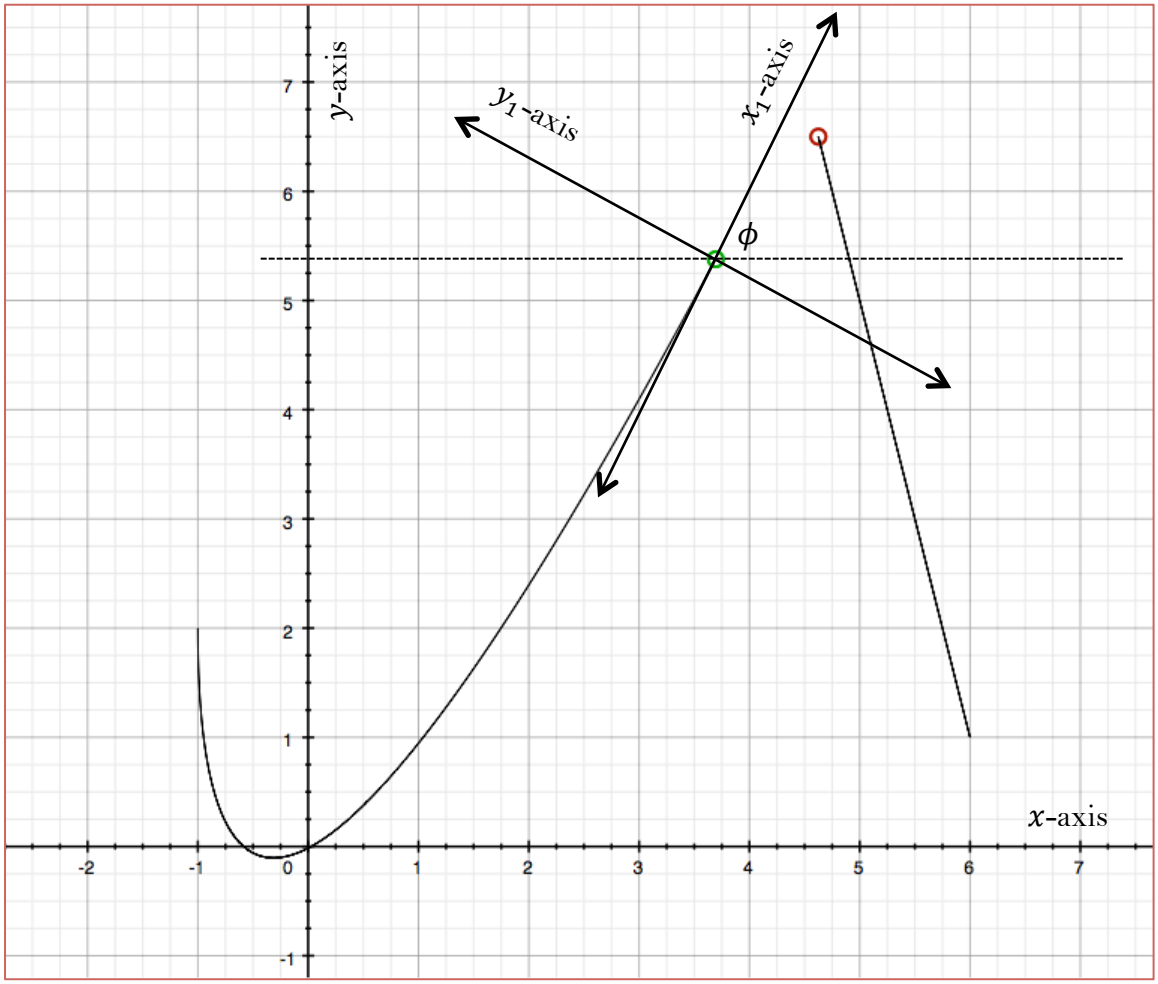
simple seek algorithm as opposed to the time-optimal seek algorithm is the relative simplicity of the SS algorithm.

Next, we must determine the threshold for imminence. Is a situation imminent if there is an impact in 50 time units? 25? 5?

To find the optimum threshold, first note that the optimum movement of the seeker will always follow a parabolic trajectory. If non-parabolic motion is used, efficiency is lost and the seeker will require more time to intercept a target. Also, recall that evasion is best achieved by sudden turns toward the perpendicular of the seeker's velocity vector. Left or right orientation (relative to the seeker's velocity vector) should be determined by relative position. View an example below. As always, the green represents the seeker and the red represents the target, with their paths given in black. The grey arrow represents the optimal evasion direction that the target should take.



When evasion occurs, the seeker will accelerate in a direction perpendicular to the seeker's velocity vector. This can be represented by drawing two additional and perpendicular axes of motion, known as the x_1 and y_1 axes. We can represent the situation in a position graph where the x_1 axis is parallel to the seeker's velocity vector when the evasion begins. The y_1-axis will therefore be parallel to the seeker's predicted acceleration vector after evasion, because the x_1 and y_1 axes are perpendicular. Keep in mind that although the axes are oriented differently, they continue to represent position. Let ϕ equal the counterclockwise angle created between the x- and x_1-axes. Define $(0,0)$ in the x_1y_1-plane to be equivalent to (a, b) in the xy-plane. This axis rotation and shift is demonstrated here, exactly when evasion begins.



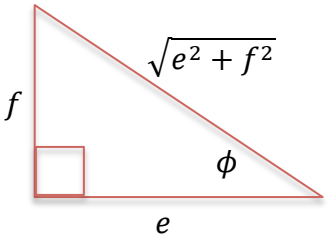
Now one must determine how to convert points from the x and y coordinate plane to the x_1 and y_1 coordinate plane. Obviously, the plane is centered at the seeker (a, b). It is also rotated by ϕ, which is equal to the argument of the seeker's velocity vector $\begin{bmatrix} e \\ f \end{bmatrix}$. This angle is equal to $\arctan\left(\frac{f}{e}\right)$. By this logic, the points are translated by vector $\begin{bmatrix} a \\ b \end{bmatrix}$ and rotated by ϕ, or $\arctan\left(\frac{f}{e}\right)$.

Translation yields the point $(c - a, d - b)$. Rotation can be accomplished by writing the translated vector as matrix $\begin{pmatrix} c-a \\ d-b \end{pmatrix}$ and multiplying by the clockwise rotation matrix of $\begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}$. In this case, the product of these matrices is as such:

$$\begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} c-a \\ d-b \end{pmatrix} = \begin{pmatrix} (c-a)\cos(\phi) + (d-b)\sin(\phi) \\ (d-b)\cos(\phi) + (c-a)\sin(\phi) \end{pmatrix}.$$

Therefore, the coordinates of the target on the x_1 and y_1 axes are $((c-a)\cos(\phi)+(d-b)\sin(\phi), (d-b)\cos(\phi)+(c-a)\sin(\phi))$. Because the new axes are centered on the position of the seeker, the position of the seeker is at the origin.

Before continuing, it may be useful to simplify the coordinates for the target's position in the new axes. Visualize a triangle where $\phi = \arctan\left(\frac{f}{e}\right)$. This can be expressed by setting one leg of the triangle equal to e and the other to f. Obviously, the Pythagorean theorem can be used to solve for the hypotenuse, which equals $\sqrt{e^2+f^2}$.



From this diagram, it is clear that $\sin(\phi) = \frac{f}{\sqrt{e^2+f^2}}$ and $\cos(\phi) = \frac{e}{\sqrt{e^2+f^2}}$.

Since $\phi = \arctan\left(\frac{f}{e}\right)$, $\sin(\phi) = \sin\left(\arctan\left(\frac{f}{e}\right)\right) = \frac{f}{\sqrt{e^2+f^2}}$ and $\cos(\phi) = \cos\left(\arctan\left(\frac{f}{e}\right)\right) = \frac{e}{\sqrt{e^2+f^2}}$.

Therefore, the target's position in the x_1-y_1 plane can be expressed as:

$$\begin{pmatrix} \frac{(c-a)e}{\sqrt{e^2+f^2}} + \frac{(d-b)f}{\sqrt{e^2+f^2}} \\ \frac{(d-b)e}{\sqrt{e^2+f^2}} + \frac{(c-a)f}{\sqrt{e^2+f^2}} \end{pmatrix}$$

Or, more simply, as:

$$\begin{pmatrix} \frac{(c-a)e+(d-b)f}{\sqrt{e^2+f^2}} \\ \frac{(d-b)e+(c-a)f}{\sqrt{e^2+f^2}} \end{pmatrix}$$

Now let q equal the "evasion threshold." That is to say that if $t < q$, active evasion by moving perpendicular to the velocity vector of the seeker should occur.

At time q after evasion starts, the target will be at this point, assuming that it dodges away from the seeker's axis of motion. This is because the target's velocity ($\sqrt{g^2+h^2}$)

will be multiplied by the time spent at that velocity (q) and then added to the y_1-axis position.

$$\left(\frac{(c-a)e+(d-b)f}{\sqrt{e^2+f^2}}, \frac{(d-b)e+(c-a)f}{\sqrt{e^2+f^2}} + q\sqrt{g^2+h^2}\right)$$

The seeker, assuming it also begins accelerating perpendicular to its current direction of motion, will be at:

$$\left(q\sqrt{e^2+f^2}, \frac{1}{2}iq^2\right)$$

The x_1 coordinate of the seeker is derived from the seeker's x_1-axis velocity ($\sqrt{e^2+f^2}$) multiplied by the time spent at that velocity (q). The y_1 coordinate of the seeker is derived from the fourth kinematic equation, which states that $d = v_i t + \frac{1}{2}at^2$, where d is y_1 displacement (Δy_1), v_i is initial y_1 velocity (0), t is time (q), and a is the seeker's acceleration (i).

Realize that one may assume that the target's x_1-position value will stay constant throughout the entire active evasion sequence, since the entirety of the target's velocity will be applied in a direction parallel to the y_1 axis.

During evasion, the target and seeker will accelerate in the same direction. Initially, the compartmentalized y_1 velocity of the target will exceed that of the seeker. However, as the seeker continues to accelerate, eventually the compartmentalized y_1 velocities of the two objects will equal. Vertical separation will steadily increase until this point, where the seeker and target have equal y_1 velocity.

Maximum separation occurs when q is such that the target's and seeker's vertical velocities are equal when they pass each other on the x_1 axis.

These are the velocities of the seeker and target, which are calculated by finding $\frac{dx_1}{dq}$ and $\frac{dy_1}{dq}$ for both points.

Seeker: $(\sqrt{e^2+f^2}, iq)$

Target: $\left(0, \sqrt{g^2+h^2}\right)$

The target must begin evasion such that their y_1-velocities equal as their x_1-positions equal. Mathematically, this is expressed in the following relationship:

$$iq = \sqrt{g^2+h^2}$$

Therefore:

$$q = \frac{\sqrt{g^2 + h^2}}{i}$$

From this, one can conclude that active evasion should begin when $t \leq \frac{\sqrt{g^2+h^2}}{i}$. Evasion must be perpendicular to the seeker's velocity vector and in the direction of the y_1-position of the target in the x_1-y_1 plane. Realize that the target's y_1-position of $\frac{(d-b)e+(c-a)f}{\sqrt{e^2+f^2}}$ always has the same sign as its numerator $(d-b)e+(c-a)f$ due to the fact that the denominator is nonnegative for all e and f.

Therefore, the direction of active evasion may be expressed in this simple set of statements: When $(d-b)e+(c-a)f \geq 0$, turn 90° left of the seeker's velocity vector, and when $(d-b)e+(c-a)f < 0$, turn 90° right of the seeker's velocity vector.

To find the direction of motion perpendicular to the seeker's velocity vector in the xy plane, we can calculate the counterclockwise angle created between the horizontal and the $\begin{bmatrix} e \\ f \end{bmatrix}$ vector in the xy plane. Recall that this is equal to our definition of ϕ, and $\phi = \arctan\left(\frac{f}{e}\right)$.

Just as with the arctangent function in the SS algorithm, the arctangent in this evasion algorithm must be corrected for the quadrant that point (f, e) falls into.

To avoid redundancy, the quadrant-by-quadrant analysis of this arctangent will not be written within the evasion section of this investigation, but the concepts behind it may be found in the SS section of this investigation.

Using these concepts, the optimal evasion flight path may be calculated for any given target and seeker.

Conclusion

In this investigation, two seek algorithms have been developed for particles in two-dimensional space. In addition to this, other seek algorithms such as the direct-acceleration method have been examined. To compare the viability of these various algorithms, I have created a set of ten various scenarios and measured the amount of time that each seek algorithm required to intercept the targets. The average time to solve each scenario is given for each algorithm, with my original algorithms in italics:

Seek Algorithm	**Avg. Time to Solve (seconds)**
Direct-Acceleration	*52.277*
Direct-Acceleration with leading	*38.196*
Simple Seek	*8.630*
Time-Optimal Seek	*1.642*

From this graph, it is immediately obvious that the two algorithms developed in this investigation (the Simple Seek and the Time-Optimal Seek algorithms) are far superior to traditional and intuitive methods of seeking. It is not surprising the Time-Optimal Seek algorithm is the fastest algorithm, given the fact that it finds the mathematically optimal solution. However, this data does reveal a significant difference in the viability of the Simple Seek and the Time-Optimal Seek algorithms.

In addition to evaluating the effectiveness of these algorithms, I'd also like to discuss possible areas for further exploration in this investigation. Perhaps the most obvious area for further investigation lies in the extrapolation of these algorithms from two-dimensional systems to three-dimensional systems. As stated in the introduction, this is actually quite simple once a two-dimensional solution is found.

In the Time-Optimal Seek algorithm, a third dimension results in a third parametric equation (z-axis) for the seeker, which is then summed with the other equations to cancel the trigonometric functions. However, a three-dimensional system results in a polynomial of the 6^{th} degree as opposed to a polynomial of the 4^{th} degree. While this may seem trivial, it changes how the equation is solved because there is no general formula for zeroes in polynomials beyond the 4^{th} degree. This was mathematically proven by the Abel-Ruffini theorem in 1823, and as a result, it is not possible to find an exact general formula for seeker solutions in three dimensions (Abel-Ruffini). (Note that these algorithms do not necessarily need to work with exact solutions due to the fact that they are applied recursively. However, including three dimensional solutions in this investigation would have obscured the concepts required to derive these seek algorithms.)

Another area where this investigation could be expanded is in the data collection. In this investigation, seekers are omniscient – that is, they are constantly aware of all variables in the simulation. However, in real life this is often not the case. Missiles must rely on radar pings or infrared signatures to discern the location and movement of enemy aircraft, and birds-of-prey must rely on their keen eyesight to discern the location and movement of their next meal. All of these methods of data collection are subject to noise, or randomness. To create a full-featured seek algorithm, some type of signal processing must be implemented to pass variables to the equation.

One method of signal processing that is particularly intriguing is called a Kalman filter (Vergez). Kalman filters compare noisy information from the current timestep to noisy information from previous timesteps. Transition modeling is used to predict the effect that various inputs will have on the output of a system. By observing how noise affects the progression of variables with respect to time, a Kalman filter can filter noise to estimate the true position and movement of a target.

As stated in the introduction, the seek and evasion algorithms that have been developed in this investigation have a huge number of possible applications. In a society where automated cars, unmanned aerial vehicles, and robots are poised to become the workhorses of the future, seek and evasion algorithms will be crucial. In addition to this, the number of aeronautical applications for these algorithms is massive. Moving towards a goal and moving away from threats are two of the most primal and basic types of behavior modeling in both humans and machines. Any system of automated movement must include these two key features, so it is of utmost importance that effective and fast seek and evasion algorithms are developed.

Appendix A: Solving the Quartic

The quartic formula is extremely complicated. It consists of four separate equations, labeled r_1, r_2, r_3, and r_4, with each corresponding to one of the four possible roots of the quartic. For reasons beyond the scope of this investigation, equation r_1 will always produce the positive real root if only one such root exists, as is the case in the time-optimal seek algorithm. In the equation below, r_1 of the quartic formula has been used to solve for t given the quartic at hand. I apologize in advance for the lack of formatting; there are no widely available software programs capable of rendering the following equation, as it is almost 1,500 characters in length.

$t =$ -1/2 sqrt((8 (l^2+m^2))/(3 i^2)-(128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3)/(3 2^(1/3) i^2)-(2^(1/3) (16 (l^2+m^2)^2-48 i^2 (j^2+k^2)))/(3 i^2 (128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3)))-1/2 sqrt(-(16 (j l+k m))/(i^2 sqrt((8 (l^2+m^2))/(3 i^2)-(128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3)/(3 2^(1/3) i^2)-(2^(1/3) (16 (l^2+m^2)^2-48 i^2 (j^2+k^2)))/(3 i^2 (128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3))))+(16 (l^2+m^2))/(3 i^2)+(128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3)/(3 2^(1/3) i^2)+(2^(1/3) (16 (l^2+m^2)^2-48 i^2 (j^2+k^2)))/(3 i^2 (128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2+sqrt((128 (l^2+m^2)^3+1152 i^2 (j^2+k^2) (l^2+m^2)-1728 i^2 (j l+k m)^2)^2-4 (16 (l^2+m^2)^2-48 i^2 (j^2+k^2))^3))^(1/3)))

Solving this produces a purely radical and therefore exact solution for t, which may be then used to find exact values for θ.

Appendix B: Examples of Algorithms in Action

In this appendix, credit must be given to my close friend and fellow IB HL Math student, John Peurifoy. John used the simple seek and evasion algorithm that I have developed to create a simple missile game on the computer. Several images from John's game have supplemented my own examples of the algorithms, and they are far superior from a graphical standpoint. In the examples below, it should be immediately clear which path represents the seeker and target, because the target will travel linearly while the missile will curve in an arc. However, for additional reference, the initial point of the target has been marked with a small plus sign (+) in my models. In John's images, the seeker is blue and the target is red, with their respective flight paths shown in the same color.

Time-Optimal Seek Algorithm

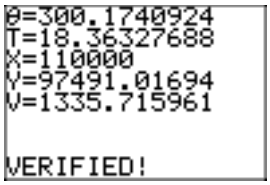
Various examples are given with inputs from realistic data about missile encounters. Units will be in meters, meters per second, and meters per second squared.

Example 1

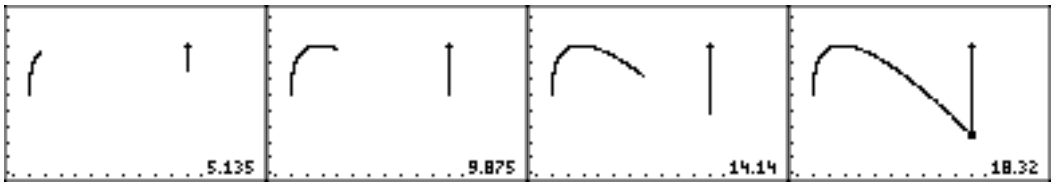
In this example, a difficult scenario is illustrated. A missile is fired in passing; to hit the target the missile must perform an almost 180 degree turn.

Let $a = 100000, b = 100000, c = 110000, d = 103000, e = 0, f = 800, g = 0, h = -300, i = 118.$

Solving for the variables yields the following. X and Y represent the coordinates of the intercept, and V represents the relative velocity magnitude at intercept (impact speed).



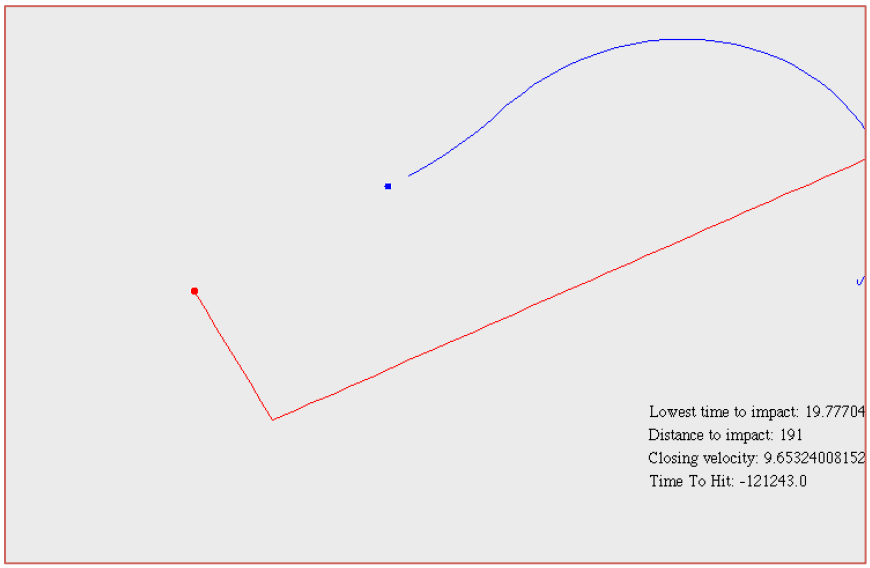
Which, when simulated, shows the intercept at 18.3 seconds after launch:



Example 2

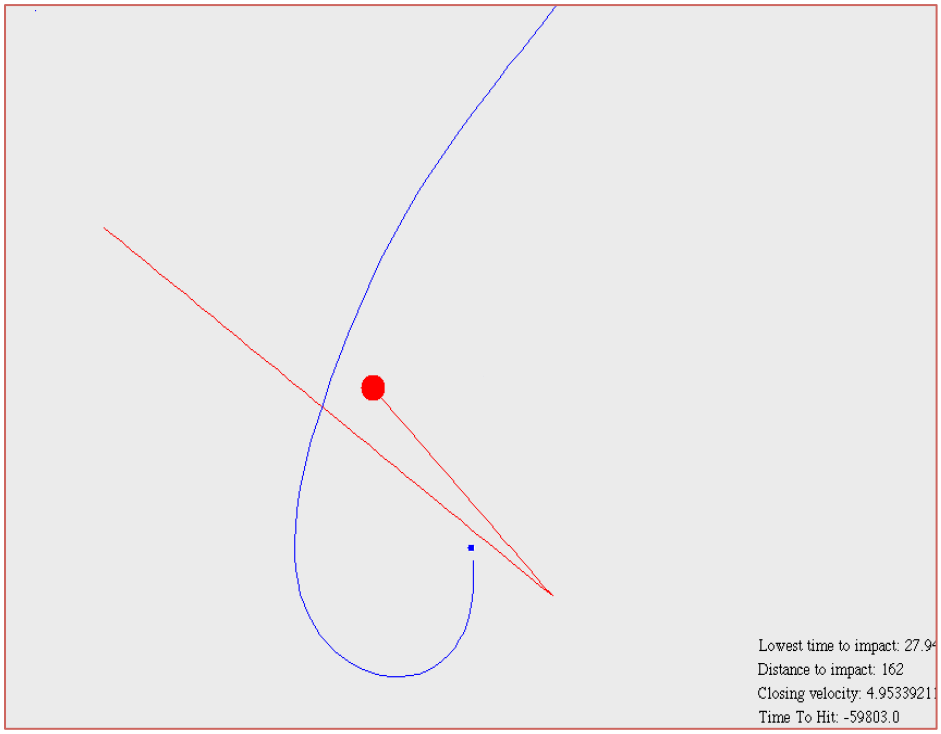
In this example, a seeker chases a dynamically moving target. The target randomly changes its direction approximately one time per second, and the seeker recalculates the

optimal solution using my Time-Optimal Seek algorithm at each timestep. As the direction of the target changes, the seeker changes the direction of its acceleration vector to ensure intercept. This can be seen in the following model:



Example 3

In the following example, the target makes a turn of nearly 180° to evade the seeker. Despite the target's extreme maneuvering, the seeker retains its missile lock, recalculates its flight path, and intercepts the target.



Appendix C: Works Cited

"Abel–Ruffini Theorem." *Princeton University*. N.p., n.d. Web. 09 Feb. 2014.

Beckmann, Petr. *A History of Pi*. New York: St. Martin's, 1971. Print.

Katz, Gabriel, and Vladimir Nodelman. *The Shape of Algebra in the Mirrors of Mathematics: A Visual, Computer-Aided Exploration of Elementary Algebra and Beyond*. Singapore: World Scientific, 2012. Print.

Kopp, Carlo. "Evading the Guided Missile." *Evading the Guided Missile*. Australian Aviation and Air Power Australia, 2004. Web. 04 Dec. 2013. <http://www.ausairpower.net/TE-Evading-Missiles.html>.

Parsch, Andreas. "Raytheon (Philco/General Electric) AAM-N-7/GAR-8/AIM-9 Sidewinder." *Raytheon AIM-9 Sidewinder*. Designation Systems, 2002. Web. 18 Nov. 2013. <http://www.designation-systems.net/dusrm/m-9.html>.

A Threat-Representative Target Is Launched from the Pacific Missile Range Facility (PMRF) to Be Intercepted as Part of a Missile Defense Agency Test of the Sea-Based Capability Under Development. 2007. Photograph. U.S. Navy Photo Archives, Kekaha, Hawaii. *Wikimedia Commons*. Wikipedia, 6 Nov. 2007. Web. 1 Dec. 2013. <http://commons.wikimedia.org/wiki/File:US_Navy_071106-N-0000X-003_A_threat_representative_target_is_launched_from_the_Pacific_Missile_Range_Facility_(PMRF)_to_be_intercepted_as_part_of_a_Missile_Defense_Agency_test_of_the_sea-based_capability_under_development.jpg>.

Vergez, P. L., and R. K. Liefer. "Target Acceleration Modeling for Tactical Missile Guidance." *(AIAA)*. Aerospace Research Central, May-June 1984. Web. 09 Feb. 2014.