Optimal Real Time Evasion against High Speed Pursuer Using Evolutionary Programming

Istas F. Nusyirwan, Cees Bil
The Sir Lawrence Wackett Centre for Aerospace Design Technology
RMIT University
Melbourne VIC 3001, Australia
s3093201@student.rmit.edu.au,cees.bil@rmit.edu.au

Abstract: This paper discusses the use of evolutionary programming to help a fighter pilot or to guide an UAV out manoeuvre a very agile and fast pursuer such as a missile. There are many techniques used to find the optimal evader's trajectory but these techniques have their specific drawbacks such as being time consuming and computing intensive. This is not acceptable when the solution requires high reliability and must be generated quickly. An evolutionary technique has been developed to search for an optimal trajectory for an evader against a very agile and fast pursuer. In an air combat scenario, the technique searches for trajectories that conform to aircraft's performance and aerodynamic constraints. Two scenarios were considered. The pursuer simply home-in to the evader and the evader follows the trajectory suggested from the algorithm. In both scenarios, the evader is able to steer itself from interception.

1. Introduction

The challenge in the guidance and control of an autonomous vehicle is the difficulty to generate optimal trajectory, in a computationally efficient manner that exploits vehicle nonlinear dynamics in real time.

The classic techniques of gradient descent, deterministic hill climbing and many others generally have an unsatisfactory performance when applied to nonlinear optimisation problems, especially those with stochastic, temporal, and chaotic components.

The problem of searching optimal vehicle trajectories comes from calculus of variations by formulating a continuous-time optimal control problem with necessary conditions for optimality. It is usually necessary to reduce the problem to finite-dimensional space for computing tractability by formulating an approximating nonlinear programming (NLP)[1] problem. Typical NLP problem statement uses equality constraint functions to dictate boundary conditions and enforce nonlinear model feasibility, while inequality constraints bound the states and controls, and describe obstacles in the vehicle path.

Many modern techniques for generating vehicle motion employ differentially flat, approximately flat, or other output-space and inverse dynamics parameterisations [2,3,4]. These techniques provide algorithms with a direct handle on the output signals that best describe vehicle motion while avoiding costly forward integrations and sensitivity calculations [3].

Most of these methods even those that exploit highly simplified models for vehicle motion [5], typically resort to an on-line NLP solution procedure. Although NLP can be used in some specalised cases, NLPs have several key limitations when considered for real-time implementation:-

- extreme sensitivity to initial solutions guesses;
- no guarantees of convergence to optimal (or even feasible) solutions; and
- trajectory overparameterisation.

Trajectory overparameterisation means having too many variables thus increasing algorithm dimensionality while allowing superfluous solution options. One suggestion from Dever [3], is to bundle sets of useful manoeuvres into continuous classes, organising the flight envelope into its equivalent human pilot's mental model, and simultaneously providing a more direct method to synthesize motion. This approach could produce a sub-optimal solution but it would be good enough against an intelligent pursuer.

In theory, in perfect information game, the pursuer is guaranteed success against any evader strategy, such robustness is lost in realistic situations. This is due to noise corrupting the states and estimators are required to implement the guidance law based on the optimal pursuer strategy against unknown (not necessarily optimal) evasive manoeuvres [7].

For a successful evasion from a very agile pursuer, a method has been developed to assist the evader to find an optimal trajectory. The method must be fast enough for real time use and thus must require minimal computing power.

Evolutionary computation seems to be a good candidate for the type of problem that consists of nonlinearity, temporal and chaotic components.

2. Methodology

2.1. Path Representation for the Evader

In this paper, a trajectory is defined as a set of heading angle and pitch angle describe at every time step. A point-mass model is used to represent an aircraft. The motion of its center of mass is defined by seven variables, namely the three position coordinates *x*, *y* and *z*, the speed *V*, the flight path angle γ, the heading angle $\psi$and the mass *m*. This is best described in illustration in Figure 1.

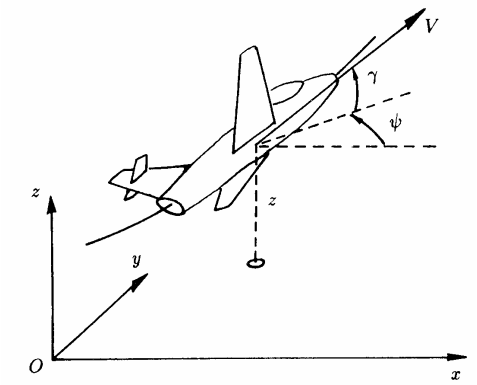

Figure 1: The aircraft is free to move in any direction as long as the path is within limit.[6]

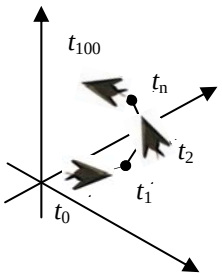

Figure 2: Change of direction at every time step.

A manoeuvre class is defined as a family of related feasible vehicle trajectories and an associated collection of parameters describing boundary conditions that govern the feasibility of the trajectories. Feasible trajectories are trajectories that make the aircraft fly within its aerodynamic, performance and stability boundaries. It is described as

$$P = \{(\psi t)_{T_0} \ldots (\psi t)_{T_f}\} \quad (1)$$

where

P = trajectories
ψ= next heading angle with respect to aircraft's body centerline
γ = next flight path angle with respect to aircraft's body centerline

The range of heading angle, ψ and flight path angle, γ, are between -30^0 to 30^0 with 2.5 degrees interval. This means, at any time the maximum heading and flight path angles change is 30^0. The manoeuvres need to comply with the constraints before being considered feasible. Each trajectory is only for a certain time frame, i.e. 100 seconds. It is best illustrated in Figure 2.

Figure 2 shows the path taken by the aircraft from time t_0 to t_f. At time t_0, the aircraft's heading and flight path angles are $\psi_{t=1}$ and $\theta_{t=1}$. The aircraft changes its heading and flight path angle at time t_1, t_2 until t_f, i.e. the final time which is 100s. These changes of heading and flight path angles are stored in computer memory as integer numbers. Each number is assigned to a specific change of heading and flight path angle as show in Table 1, to make it possible for use by the stochastic selection method. There are 625 possible combinations of change of heading and flight path angle.

For 100 second play, the algorithm generates a population of paths or strategies. A path consist of 100 possible combinations of heading and flight path angles. The selection of these combinations is randomised. As an example, for a path is represented by a series of numbers such as in Figure 3. In Figure 3, the first three digits are 347 which means the aircraft needs to turn 2.5 degrees to the left and climb 22.5 degrees. In next second the aircraft needs to change is heading

with respect to its current position, 22.5 degrees to the right and -27.5 degrees downward. This continues until 100 seconds.

Table 1: Change of heading and flight path angles assignment for use by the computer code

Number	Change of heading angle in degrees	Change of Flight Path Angle in degrees
1	-30	-30
2	-30	-27.5
3	-30	-25
:	:	:
625	30	30

```
34707705218740123143415457061229146931717655643759554029750 80
35250152350336466331282354983431580245881192073555080224933 3
84473751060331374372925763455421392485803774324200180041745 58
61906512615847841715351010941460423429657456529840453138631 83
52211335481616046081510226322354034111460500993184464 98
```

Figure 3 : Example of a representation of a path.

2.2. Evolutionary Programming

Evolutionary programming (EP) is one of a class of paradigms for simulating evolution to iteratively generate increasingly improving solutions (i.e. good trajectories for the evader) within a static or dynamically changing environment. EP evolves a set of solutions which exhibit optimal behaviour with regard to an environment and desired payoff or cost function. It could also be considered as an optimisation technique wherein the algorithm iteratively optimises the behaviours, parameters, or other constructs.

In its basic form, the EP utilises the four main components, i.e. initialisation, variation, evaluation and selection. “Learning” is a by product of the evolutionary process as successful individuals (or solutions) are retained through stochastic trial and error. Variation (e.g. mutation) provides the means for moving solutions around in the search space, thus preventing entrapment in local minima.

The evaluation function defines fitness of each solution with regard to the environment. Finally, the selection process probabilistically culls suboptimal solutions from the population, providing an efficient method for searching the topography.

The basic EP algorithm starts with a population of trial solutions which are initialised by random means. The size of the population is set to 100 for ease of computation. This paper uses the intrinsic random function in FORTRAN to generate the initial population.

2.3. Pursuer

The pursuer is an aircraft that has higher performance capability and is faster. The pursuer’s ability to manoeuvre is constrained against its aerodynamic and performance limitations, which is explained in the preceding section. The pursuer’s guidance system considered in this paper is “pure pursuit”. As describe by [8], pure pursuit means the pursuer is simply homing in to the evader.

2.4. Equations of Motion

In this paper, aircraft motion is modeled as a point mass. The reference frame is considered as a flat-earth. The mass of the aircraft is a function of time as fuel being burnt. Newton’s second law, Equation 2, is used to calculate the aircraft’s dynamics.

$$\boldsymbol{F(t)_B} = m(t) \times \boldsymbol{a(t)_B} \tag{2}$$

where

$\boldsymbol{F(t)_B}$ = vector of forces due to aircraft's engine, aerodynamic forces, external forces, such as gravitational forces and wind gusts with respect to aircraft's body-axis
$m(t)$ = the mass of the aircraft (kg), which is the function of time with respect to aircraft's body axis.
$\boldsymbol{a(t)_B} = D(\boldsymbol{v(t)})$ = the acceleration vector acting at aircraft's center of mass with respect to inertial frame.
t = time in second

The aircraft has its distinct lift coefficient, C_L. The lift coefficient in reality is a function of Mach Number, Reynold's Number and angle-of-attack. To make the analysis close to real, these values are derived from experiments and test-flight, and are represented in tabular form. The lift coefficient is given as

$$L = \frac{1}{2}\rho V^2 S C_L \tag{3}$$

When the aircraft goes faster, the drag increases quadratically as given in Eq. (4). Climbing will significantly contribute to the aircraft slowing down due to gravity.

$$D = \frac{1}{2}\rho V^2 S C_D + mg \sin\gamma \tag{4}$$

and

$$C_D = C_{D_0} + kC_L^2 \tag{5}$$

The maximum turning rate, $\psi_{\max}$, is governed by equation (6). n_{max} is the maximum load factor for the aircraft. The value of n_{max} is given in Table 2.

$$\psi_{\max} = \frac{g\sqrt{n_{\max 2}^2 - 1}}{V} \tag{6}$$

by assuming the aircraft is flying at relatively low altitude and high subsonic. Thus the maximum turning rate is restricted by the structural limit of the aircraft. From Equation (6), it is obvious that the higher the speed, V, the smaller the turning rate. When turning involves change of altitude [6], the analysis is more appropriate in terms of power. The load factor, n, is the ratio between lift and weight.

The rate of fuel consumed is given by its Thrust Specific Fuel Consumption (TFSC). The amount (kg) of fuel burnt at every second with the given thrust is

$$TFSC \times \frac{1}{3600} \times Thrust\ (N) \tag{7}$$

Eq. (7) is used to calculate the weight of the aircraft at any time step,

$$Mass_{t+1} = Mass_t - TFSC \times \frac{1}{3600} \times T_{Engine}(\mathrm{t}) \ \ldots \ \mathrm{kg} \tag{8}$$

The relationship between altitude and the maximum available thrust is given in Equation 9. *PowerSetting* values is between 0 and 1.

$$T_{alt} = T_{sealevel}\left(\frac{\rho_{alt}}{\rho_{sealevel}}\right)^{1.1} \times PowerSetting \tag{9}$$

2.5. Constraints

At any instance, the aircraft must satisfy all constraints. They are:

1. Thrust required for manoeuvre must be equal to the power setting, i.e. the power setting is fixed as 0.9.
2. Fuel weight must be greater than a minimum required to perform the manoeuvre in a given time range;
3. Turning rate must be within limits, as given in Equation 6;
4. Angle of attack must be within limits.; and
5. Load factor must be within given values, i.e. -4 and +9;

For the evader, any possible trajectory that fails one of the constraints will be discarded. Thus a rigorous and fast checking is done on each trajectory to ensure its validity.

The pursuer, in its pursuit toward the evader, needs to fly within its limit. This is done by calculating the aerodynamics and performance limits of the aircraft at every time step such as the maximum turning rate and climb rate. The maximum values changes because turning rate and climb rate are indirectly a function of altitude and speed. For example, the maximum turning rate is governed by Equation 6.

2.6. Quaternion

Euler angles were used to calculated aircraft position relative to Flat-Earth-Axis, but the method fails when the aircraft is making a vertical loop which causes a phenomenon called 'gimbal lock' to occur. Gimbal lock is the phenomenon where two rotational axis of an object pointing in the same direction. The advantages of quaternions are as follows:

- the representation of axis/angle avoids gimbal lock;
- modifying a rotation is relatively easy; and
- avoids costly alignment of matrix drift;

The rotation is calculated using quaternion before being converted back into angles.

2.7. Cost Function

The pursuit-evasion starts at t=0s. The cost function is the miss distance

$$J = \left| x_{t_f} \right| \tag{10}$$

between the pursuer and the evader. The pursuer wants to minimise J and the evader wants to maximise it. Both players have perfect information about the current state of each other, but no information about the future states.

3. Application to Air Combat between Two Aircraft

A combat simulation between two aircrafts is a complex problem to solve. Many attempts have been made to find the optimal solution for both players including differential games. This paper searches the possibility to find an optimal strategy by generating thousands of possible strategies for a given initial condition.

Both players have a complete knowledge about each other's state. The pursuer starts at a position relative to the evader. Both players' initial conditions are known *a priori*.

The pseudo-code of the problem is given below

i. generate a population of strategies
ii. for each strategy, tested it against the pursuer and give fitness
iii. the best 50 strategies are kept (elitism) and mutated to generate another 50 strategies.
iv. the population is tested and given the fitness values;
v. the best strategy, i.e. the one that has the highest fitness value, is chosen for the evader path.
vi. repeat this process for the next segment.

The strategies have to satisfy ALL constraints in order to be considered good. A 'bad' strategy would make the evader:

i. it makes the evader hit the ground; or
ii. exceed the stall limit; or
iii. exceed the maximum turning rate; or
iv. over the maximum load factor; or
v. being intercepted.

The algorithm has to be able to find 'good' strategies. These strategies must be able to out manoeuvre the agile pursuer. A good strategy creates a large miss distance to the pursuer, and the pursuer has to make a turn in order to catch up with the evader. While the pursuer is making a turn, the evader takes the advantage by flying away from the pursuer.

4. Scenarios

4.1. Scenario 1

A game between a highly agile pursuer and a less agile evader is simulated. The initial parameters are given in Table 2. The maximum thrust of the pursuer is set as 26000 kN in this scenario.

Table 2: Initial Parameters

Parameters	Value	
	Evader	Pursuer
v_t (m/s)	140	210
x-position (m)	2000	0
y-position (m)	0	0
z-position (m)	5000	5000
Heading Angle (deg)	0	0
Flight path angle (deg)	0	0
Thrust (N)	1.8×10^6	2.6×10^6
Interception Radius(m)	-	20
Maximum Turning Rate (deg/s)	30	20
Min and max load factor	-4,+9	-4,+9
Mass (kg)	8500	6875
$C_{L,max}$	2.3	1.9
C_{Do}	0.045	0.0412
k	0.117	0.132

Both players' thrust settings are kept constant at 0.9.The pursuer has a higher thrust than the evader. This gives the pursuer an advantage in terms of speed and allows the pursuer to chase the evader sooner.

The evader runs the optimisation routine to find the best path. Both players must satisfy their respective constraints during the game. Those constraints, among others, are turning rate, maximum load factor and fuel limit. The game is restricted for only 100 seconds in order to see if optimal solution could be found.

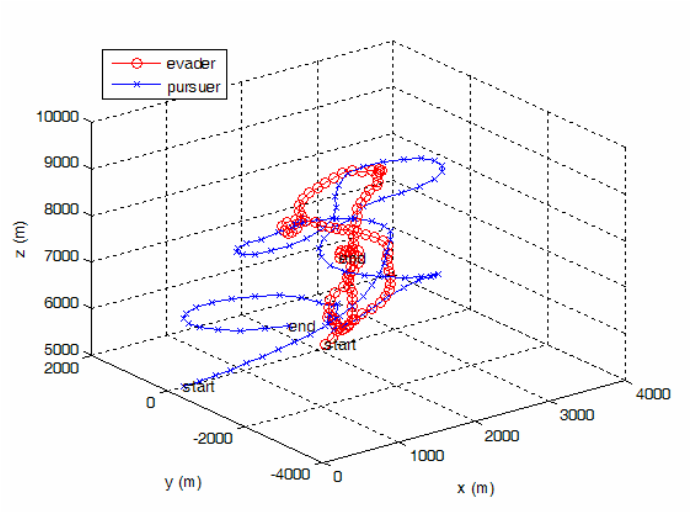

Figure 4: Scenario 1 in three dimensions.

Figure 4 shows the game in three dimensions. The evader manages to find the optimal trajectory to guide itself away from interception. The closest distance between the players is 126.5 m. The number of populations is 40 in this simulation. The first 20 populations did not provide a single acceptable strategy, but several good strategies were found after the 20th population. In total, 4000 possible strategies were produced of which only 1 percent was found to be acceptable.

The change of heading and flight path angles can be seen in Figure 5 and 6. The evader performs a hard climb and a hard turn up to 30 degrees/second. Figure 6 shows the change of heading angle in degrees. Note that the maximum permitted rate of heading change in this study is 30 degrees per second.

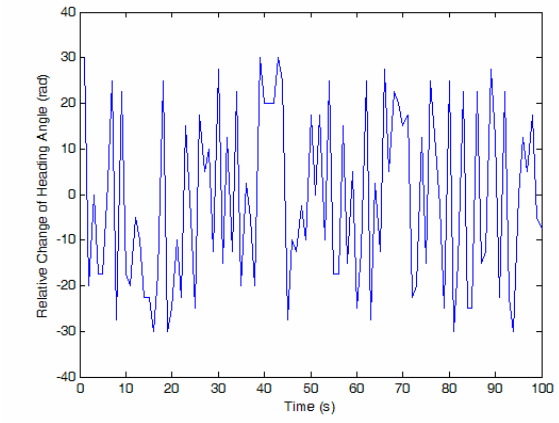

Figure 5: Relative heading change of the evader

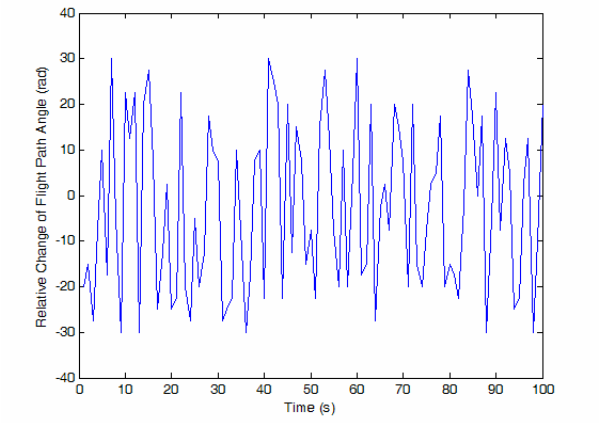

Figure 6: Relative flight path angle change of the evader.

Figure 7 shows the miss distance between both players during the simulation. The initial miss distance is 2000 meters. The faster pursuer closes its distance up to 130 m (at t = 19 s). At time t=19s, the evader makes a sharp turn to the right and the pursuer tries to follow but is out-manoeuvred by the evader because its turning rate is higher. The correct timing to do the "out-of-danger" manoeuvre is crucial. Such timing can only be achieved with perfect or near-perfect information and knowledge of the pursuer's guidance system is important.

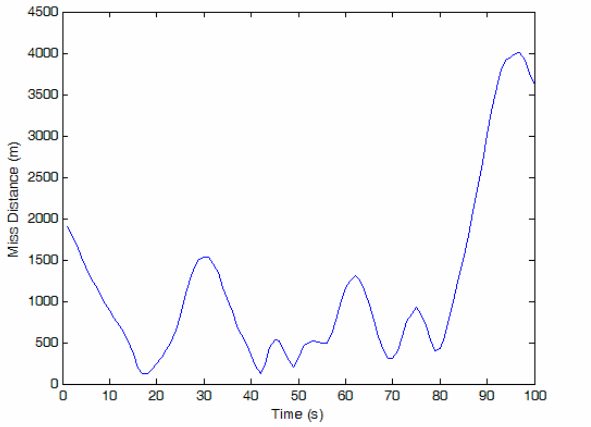

Figure 7: Distance between the pursuer and the evader.

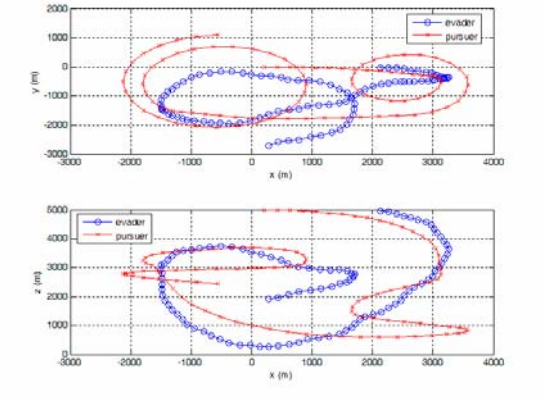

Figure 8: Two dimensional plot in plan view and side view.

The game can also be visualised in two-dimensions as shown in Figure 9. The evader dives right after t=0s in to a common plane. After that the evader makes a vertical half loop to out manoeuvre the pursuer and another horizontal-S to out-manoeuvre the pursuer for the second time. Figure 10 shows distance of x, y and z as a function of time. The velocity of the player is calculated based from Equation 1. Figure 11 shows the velocity profile of the evader during the game. The pursuer increases velocity as the evader is in a dive and a slight decrease in velocity when the evader climbs.

The maximum available thrust is a function of altitude as given in Equation 9. The power setting is kept constant in this simulation. This translates into the amount of available thrust according to Equation 9. Knowing the amount of thrust

produced, we could calculate the amount of fuel consumed. The pursuer must intercept the evader in a certain time period before it running out of fuel.

The weight reduction would translate into lighter aircraft and higher acceleration. However, higher velocity will mean higher drag, especially during turning.

The interdependence of variables makes the problem highly non linear and very difficult to solve analytically. The analysis would not be completed if the non-optimal trajectories were not discussed. It was shown that 90 percent of the strategies evaluated were not optimal. The main cause is that the trajectories could not avoid interception. Others are hitting the ground, over the ceiling limit, exceeding the maximum turning rate, exceeding the load limit and etc.

There are many factors that ensure the success of the algorithms. The number of populations should be sufficient to generate enough strategies in order to find the optimal solution. Too few populations would mean the optimal strategy may not be found, too many will only make the computation time longer. The algorithm should be kept moderate so that it is suitable for real time use.

The actual computation time for 100-seconds simulation to complete is about 25 seconds with the number of strategies evaluated is 4000 and using 1.5 GHz Intel Pentium-M CPU. The short computing time means the algorithm has a good prospect to be use for real time application.

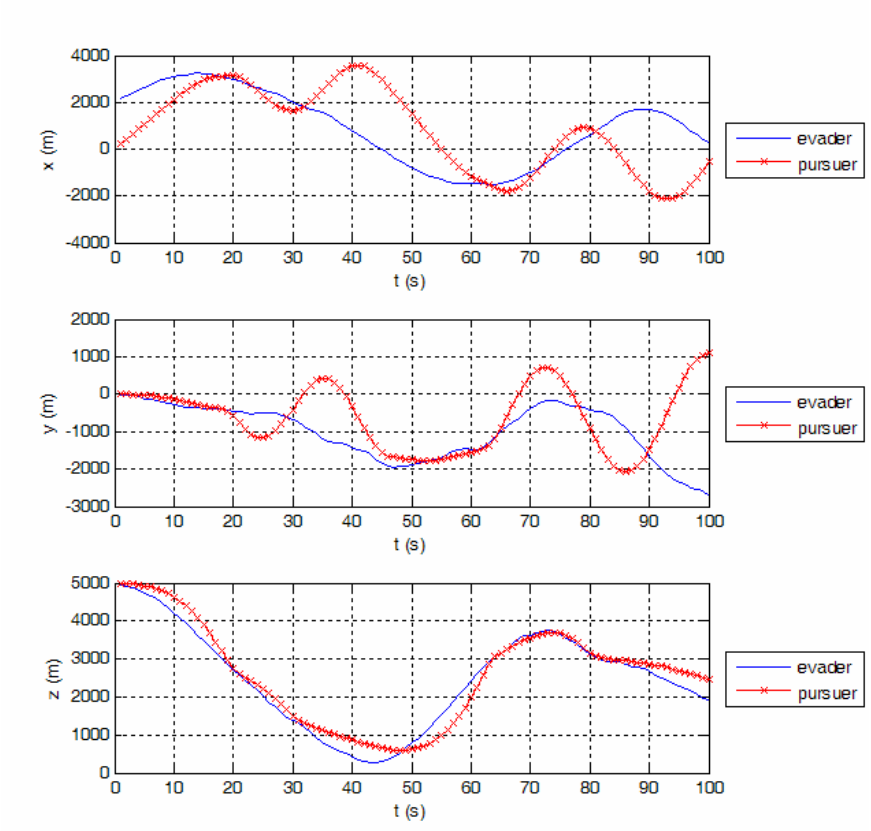

Figure 9: The x,y and z positions versus time.

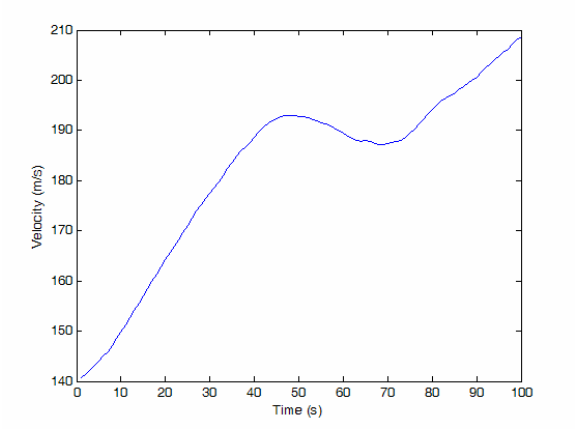

Figure 10: Velocity profile of the evader.

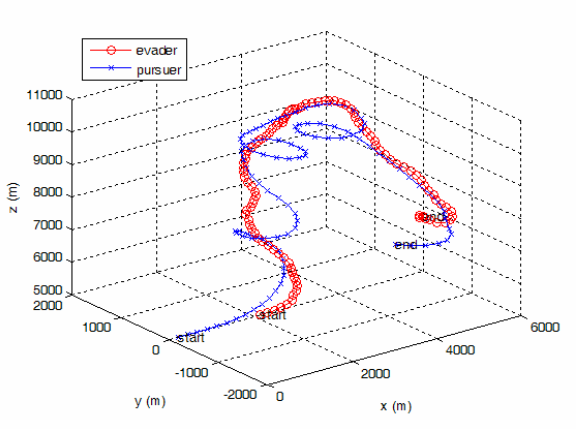

Figure 11: Scenario 2 in three dimensions.

4.2. Scenario 2

Table 2: Initial Parameters

Parameters	Value	
	Evader	Pursuer
v_t (m/s)	140	210
x-position (m)	2000	0
y-position (m)	0	0
z-position (m)	5000	5000
Heading Angle (deg)	0	0
Flight path angle (deg)	0	0
Thrust (N)	1.8×10^6	1.6×10^6
Interception Radius(m)	-	50
Max Turning Rate (deg/s)	30	20
Load factor limits	-4,+9	-4,+9
Mass (kg)	8500	6875
$C_{L,max}$	2.3	1.9
C_{Do}	0.045	0.0412
k	0.117	0.132

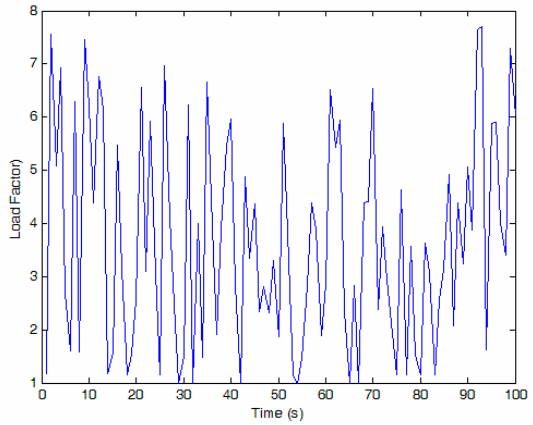

Figure 12: The evader's load factor time history.

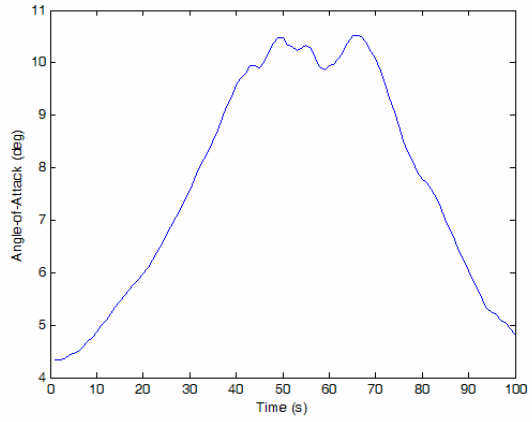

Figure 13: The evader's angle-of-attack time history.

In scenario 2, the thrust of the pursuer is changed to 1.6 x 10^6 N and the capture radius is reduced to 50.0 m. The algorithm is able to find an optimal trajectory for the evader as shown in Figure 11. Figure 12 shows the time history of the load factor acting on the evader. It is shown from Figure 12 that the load factor never exceeds 9.0 limit at any time during the simulation. The time history of the evader's angle-of-attack is given in Figure 13. As seen in Figure 11, many hard turns were observed between 40 and 80 seconds of the game. This explains why the evader experienced high angle-of-attack during that period of time. It was observed that the evader used well known manoeuvres such as high-g barrel roll, vertical-S and horizontal-S to avoid interception.

The optimal trajectories found in Figure 11 and 4 are different. This is because of the non linearity of the problem and interactions between many variables during the optimisation process. A slight change of a variable has a significant effect on the others, resulting a completely different optimal trajectory.

5. Future Work

This algorithm in future will be expanded to use parallel computing to increase its accuracy in finding the optimal solutions. It is expected that a parallelised version of the methodology will significantly reduce the computing time and improve the quality of the results.

6. References

[1] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization", Journal of Guidance, Control and Dynamics, Vol. 21, No.2, 1998, pp 193-207.

[2] Chauvin, J., Sinegre, L., and Murray, R.M., "Nonlinear Trajectory Generation for the Caltech Multi-vehicle Wireless Test bed," European Control Conf., 2003.

[3] Dever, C. et. al , "Nonlinear Trajectory Generation for Autonomous Vehicles via Parameterized Manoeuvre Class", Journal of Guidance, Control and Dynamics, Vol. 29, No. 2, 2006, pp 289-302

[4] Faiz, N., Agrawal, S. K., and Murray, R. M., "Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities," Journal of Guidance, Control, and Dynamics, Vol. 24, No. 2, 2001, pp. 219–227.

[5] Seywald, H., "Trajectory Optimization Based on Differential Inclusion," Journal of Guidance, Control, and Dynamics, Vol. 17, No. 3, 1994, pp. 480–487.

[6] Vinh, N. X., "Flight Mechanics of High-Performance Aircraft", Cambridge Aerospace Series 4, Cambridge University Press, New York. 1993

[7] Shinar,J.;Turetsky V., "Towards Reduced Miss Distance", Final Technical Report No AFOSR Contract No F61775-01-WE018, Technion Research and Development Foundation LTD., 2002

[8] Coulter, Craig R., "Implementation of the Pure Pursuit Path Tracking Problem", CMU-RI-TR-92-01, Technical Report, The Robotics Institute, Carnegie Mellon University, 1992.