Sakarya University Journal of Science, 22 (5), 1466-1476, 2018.

SAKARYA UNIVERSITY
JOURNAL OF SCIENCE

e-ISSN: 2147-835X

http://www.saujs.sakarya.edu.tr

Recieved	Accepted	DOI
2018-03-06	2018-07-05	10.16984/saufenbilder.

Agent-based Dynamically Reconfigurable Missile Modelling

Mehmet Fatih Hocaoğlu*[1], İlyas Kandemir[2]

ABSTRACT

In this study, dynamic model of an air platform (missiles and aircrafts) that has the ability to maneuver in all axes (displacement and rotation) and to follow another aircraft is achieved. Rigid aircraft mathematical models are adopted for the model aircraft and the calculations are done in a generic way excluding the effects of aeroelasticity. While considering the geodesic geometry of earth, the gravity model is also designed accordingly, and the atmospheric conditions at a given altitude are included in the calculations by using an atmospheric model. Depending on analysis purposes, it is possible to change the air platform degrees of freedom. Including degrees of freedom, for missile models, the user-defined definitions such as phase definitions, engine selections and customization of engine properties, seeker activation schedule and fuse type are configurable through an interface. Phase transition controls, motor controls, sensor activation/deactivation decisions are designed reasoning-based. To be able to enhance modeling flexibility and inject a reasoning capability, the models are designed as intelligent agents and they are modeled in Agent driven Simulation Framework (AdSiF). How a missile configuration is defined and how it is dynamically changed based on user interactions and user defined conditions are conducted using an agent-based modeling language. Furthermore, a novel approach for scripting and execution of the dynamic time frequency update is put forward in the study.

Keywords: AdSiF, Agent based simulation, Air Defense, Aircraft Modeling, Missile Modeling, Six Axis Motion

1. INTRODUCTION

Ammunition modeling and simulation are much more important in terms of defense strategies, especially nowadays. The degree of freedom of an ammunition is important in terms of success in catching the target it follows and for the sensors that detect it. The target tracking maneuverability that is followed is important, as it differs in radar cross section and thermal signature appearing from the orientations for the detecting sensors. Although positioning with three (3) degrees of freedom (displacement) simulations of ammunition and follow-up are possible, it is important to add rotational movements into calculations in order to accurately describe maneuvering predictions and abilities. While designing a scenario, ammunition models have been developed so as to be differentiated at design and operation time, while the sensor systems included in the simulation are differentiated in terms of objectives and scenario analysis purpose. During scenario design, the degree of freedom can be selected using ammunition model design interface. Whenever user defined conditions are met, and/or phase duration is elapsed, and/or user selected event interaction occurs, the degree of freedom is changed in simulation run time. In this study, for the simulation design, a general flight mechanics model was used for an ammunition [1], [2]. Axes transformations used in the design are available in the open literature [2]. Although all of these studies have been dealt with flexible

* Corresponding Author
[1] Istanbul Medeniyet University, Faculty of Engineering and Natural Sciences, Department of Industrial Engineering, İstanbul, Turkey – mfatih.hocaoglu@medeniyet.edu.tr
[2] Gebze Technical University, Faculty of Engineering, Department of Mechanical Engineering, Gebze, Kocaeli, Turkey - kandemir@gtu.edu.tr

designs due to the tactical objectives and the thrust mechanism, all air vehicles are represented by a general model in this study [3]. In the created model, the forces acting on the aircraft are based on the translational and rotational acceleration feedback calculated in the control system. There are different control and feedback approaches to the control system in the literature [3]. In this study, a cascade PD control system is used for deviation and yaw control, altitude and pitch control, which is sufficient to describe the core of the problem. The control parameters required for this approach are determined and verified by computer simulations developed. In the validation scenarios implemented, turning maneuvers to catch fixed or moving targets are provided by a follow-up calculation model integrated into the simulation. The differences between the ammunitions developed (bullets, rockets, bombs and missiles) are defined by phases, fuse types and engine options characterized in the ammunition configuration definition. The phase definitions for missile models, the motor selection and motor properties, the seeker heading options, the fuse selections and the selection of the guidance algorithm are done by the user selections from the defined interfaces and they are executed based on the user-defined rules and the results of the reasoning based on the defined predicates. The missile movement model has been developed as time increment based and designed to implement the maneuvers necessary to achieve the goal that is assigned to them. The aerodynamic and thrust forces and moments are calculated at each movement according to the control applied in the direction of the determined control algorithm in the progress step Δt. According to these forces and moments, the platform updates the angular and axial velocities at the next time step and proceeds with the calculated velocities. The progression step is dynamically calculated based on the target and missile dynamics, and each entity may use different progress steps in the scenario. This is defined as dynamic frequency execution. Synchronization of the target and missile time steps is achieved by providing defined conditions and/or predicates, thus achieving faster and higher resolution execution. Synchronization is mapping the highest time advance resolution to the entities that do not request time at the same resolution to get them update their object states for this resolution. In a scenario, the synchronization in the sense of updating their object states for the highest resolution does not need covering all the entities. This is a feature that increases operating efficiency.

Simulation design and operation is developed based on Agent Based Simulation Framework (AdSiF) [4]–[6]. AdSiF provides a dynamic frequency time management, run-time synchronization, and dynamically configuration change for any entity in the scenario at run time, and a scripting modeling language.

The article is organized as follows. In Part 2, brief information about AdSiF is presented. In Sections 3 and 4, the missile flight dynamics modeling background information and the flight dynamics model are presented. In Section 5, phase changes, time synchronization and degrees of freedom change behaviors of the missile are presented. Operational results are presented with the example presented in Section 6. Conclusions and discussions are presented in Section 7.

2. ADSIF: AN AGENT-BASED SIMULATION SYSTEM

AdSiF is an agent and simulation development environment that enables continuous and discrete event simulation execution in an agent architecture. AdSiF combines multiple programming paradigms into state-based programming paradigm as an integrated paradigm. These paradigms have been defined as object-oriented programming, logic programming, and aspect oriented programming [7], [8]. State-based programming diagrams provide a scripting language based on which all semantic structures of each entity developed are modeled by this scripting language. The atomic functions that the behaviors access are defined as functions developed in the C ++ language. There is no semantic flow in the C ++ class structure of the entity as whole semantic flow is kept in behavior declaration script. The simulation realizes the simulation execution by interpreting the kernel interpreter model scripts. Because of the agent architecture and event driven architecture that the entities developed in AdSiF are based on, the interactions between entities are provided as event transmissions.

3. BACKGROUND INFORMATION FOR FLIGHT DYNAMICS

This section provides information that is considered to be necessary to follow the study better. These are grouped under the headings of Axes Set, Target Coordinates, Variable Notation and Atmospheric Models.

3.1. Axes Set

Simulations tracking a target air vehicle quite often employ different coordinate systems and transformations between. Ground, air, tracker, target and body coordinate systems are most commonly used. In this study besides the geodetic coordinates per ground, NED (north-east-down) coordinates on the body and Euler angles (θ_x , θ_y , θ_z) for the orientation are used.

In body cartesian coordinate system; x axis points to the nose of the aircraft, perpendicular right wing direction is the y axis and the perpendicular bottom is directed in the positive z axis. Angular velocities are named correspondingly (ω_x , ω_y , ω_z). Angular position according to the target is shown as rise (θ_h) and deviation (φ_h). In Figure 1 these axes set are shown. The orientation of the air platform is determined by the latitude, longitude and altitude according to the geodesic coordinates of the 3D platform, while the orientation is expressed by Euler angles with respect to the NED axis.

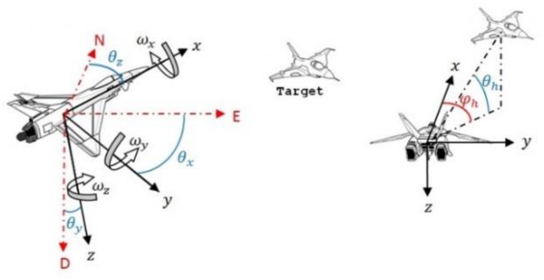


Figure 1. Axis and angle descriptions for the air platform

3.2. Target Coordinates

One can think that angular position of the target may be expressed easily for the close targets as depicted in Figure 1, however, considering the roundness of the Earth, this is not so for far targets. In the altitude control there are two options: Fixed altitude or fast-catch. In the fixed altitude approach, aircraft first reaches the target altitude with a proper ascent/descent velocity per onset parameters and travels in this altitude. Fast-catch methodology is for an ascend/descend path that minimizes the intercept time.

3.3. Variable Notation

In the study, a time marching simulation is used, and the previous state of the variables are denoted by index “0”.

Since different coordinate systems are used in the study, for the transformation between the coordinates a transformation operator notation is required. For instance, the transformation of $\vec{c}$ vector in NED coordinates to geodetic coordinates with origin $\vec{X}$ is expressed as $\{NED \rightarrow Geo\}(\vec{X},\vec{c})$. Added to this, when a coordinate system name is used as an index of a variable, that variable is considered in that coordinates. A vector $\vec{b}$ in cartesian coordinates is depicted with its components as $\vec{b} = [b_x \quad b_y \quad b_z]^T$.

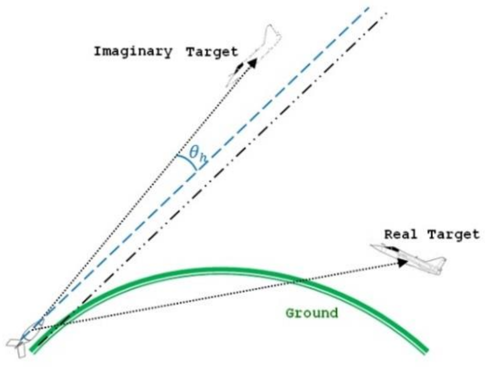


Figure 2. Revision of the direction of target ascension for the world's curvature

3.4. Atmospheric model

Regional wind effect is included in the problem along with the standard atmospheric model. Temperatures on different altitudes are easily calculated depending on the sea level temperature. Absolute air temperature (T) is important for the calculation of the local speed of sound: (a_{sound}):$a_{sound} = \sqrt{\gamma RT}$. Gas constant for air is $R = 287\ J/kgK$ and the specific heats ratio is γ=7/5. Considered this, for an aircraft of air speed of $|\vec{V}_{air,body}|$ in the body coordinates, Mach Number is equal to $M = |\vec{V}_{air,body}|/a_{sound}$. Mach number is required in the calculation of aerodynamic magnitudes such as lift and drag, and of propulsion as well.

4. FLIGHT DYNAMICS MODEL

Even though the control and navigational model of the simulated problem is rather complicated, here, a general solution method of a 6 degrees of freedom (6DOF) aircraft simulation is presented. The position of the platform in 3D space is stated by the center of the gravity. All rotations are defined about the perpendicular body axes originated at this point in order to avoid unnecessary complexities in the motion of equations. Inertial, geometric and aerodynamic design characteristics and maneuver restrictions of the aircraft are required and considered to be known.

4.1. Initial conditions

At the beginning of the simulation, the state of the aircraft is: Geodetic position ($\vec{X}_0$), NED velocity ($\vec{V}_{NED,0}$), orientation ($\vec{\theta}_0$), target position ($\vec{x}_{h,0}$) and propulsion ($\vec{T}_0$).

Before any time increment of the simulation, the angular velocity ($\vec{\omega}_0$) about the body (x , y , z) axes, Euler angular velocity ($\dot{\vec{\theta}}_0$), instantaneous fuel quantity

(Y_0), total mass (m_0) and the angular and translational accelerations $(\vec{\omega}, \vec{a})$ to be applied in the step and its duration, Δt, has to be already determined.

4.2. Time Increment

Rotational and translational velocities are accepted to be changed linearly during the interval Δt and their values are calculated as the average of previous and new values. Thus, new angular velocities are calculated as

$$\vec{\omega}=\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}=\vec{\dot{\omega}}\,\Delta t+\vec{\omega}_0 \qquad (1)$$

and new Euler angular velocities are

$$\vec{\dot{\theta}}=\begin{bmatrix}\theta_x\\ \theta_y\\ \theta_z\end{bmatrix} \qquad (2)$$

Where

$$\theta_x=\omega_x+\left(\omega_y\sin\theta_{x,0}+\omega_z\cos\theta_{x,0}\right)\tan\theta_{y,0}$$

$$\theta_y=\left(\omega_y\cos\theta_{x,0}-\omega_z\sin\theta_{x,0}\right)$$

$$\theta_z=\left(\omega_y\sin\theta_{x,0}+\omega_z\cos\theta_{x,0}\right)\sec\theta_{y,0}$$

By the integration in time of this value, new orientation is calculated:

$$\vec{\theta}=\Delta t\left(\vec{\dot{\theta}}+\vec{\dot{\theta}}_0\right)/2\ +\vec{\theta}_0 \qquad (3)$$

Instantaneous fuel consumption of an aircraft $\left(\frac{dY}{dt}\right)=f\left(|\vec{T}|,M,z_{geo}\right)$ should be modeled as a function of propulsion, Mach number, and altitude. Thus, the instantaneous aircraft mass is calculated as

$$m=m_0-\left(\frac{dY}{dt}\right)_0\Delta t \qquad (4)$$

Updated geodetic coordinate of the platform is

$$\vec{X}=\{NED\rightarrow Geo\}\left(\vec{X}_0,\vec{V}_{NED,0}\Delta t+\frac{1}{2}\vec{a}_{NED}(\Delta t)^2\right) \qquad (5)$$

For the new velocity, the integration in time is enough:

$$\vec{V}_{NED}=\vec{a}_{NED}\Delta t+\vec{V}_{NED,0} \qquad (6)$$

Values of position, orientation, rotational and translational velocities are updated for the next time step of the aircraft.

4.3. Determination of new angular accelerations and time increments

New accelerations and corresponding aerodynamic and propulsion forces should be calculated for the maneuver requirement corresponding to the new state of aircraft and relative position of the target. In the critical damping region a cascade PD control is employed.

4.3.1. Calculation of the Critical Angles

For the calculation of aerodynamic effects, aircraft's air speed is calculated as

$$\vec{V}_{air,body}=\{NED\rightarrow Body\}(\vec{V}_{NED}-\vec{V}_{wind,NED}) \qquad (7)$$

where

$$\{NED\rightarrow Body\}=\{Body\rightarrow NED\}^T=\begin{bmatrix}\cos\theta_y\cos\theta_z & a & b\\ \cos\theta_y\sin\theta_z & c & d\\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y\end{bmatrix}^T \qquad (8)$$

$$a=\sin\theta_x\sin\theta_y\cos\theta_z-\cos\theta_x\sin\theta_z,$$

$$b=\cos\theta_x\sin\theta_y\cos\theta_z+\sin\theta_x\sin\theta_z,$$

$$c=\sin\theta_x\sin\theta_y\sin\theta_z+\cos\theta_x\cos\theta_z,$$

$$d=\cos\theta_x\sin\theta_y\sin\theta_z-\sin\theta_x\cos\theta_z$$

Pitch and yaw angles are:

$$\alpha_1=Arctan\left(\frac{V_{air,body,z}}{V_{air,body,x}}\right) \qquad (9)$$

$$\alpha_2=Arctan\left(\frac{V_{air,body,y}}{V_{air,body,x}}\right) \qquad (10)$$

Relative target vector is

$$\vec{r}=\{Geo\rightarrow\ NED\}(\vec{X},\vec{X}_h) \qquad (11)$$

for which z-component is revised as described in Section 3.2 as

$$r_z=X_z-X_{h,z} \qquad (12)$$

and in body coordinates, relative target position is calculated as

$$\vec{r}_{body}=\{NED\rightarrow Body\}(\vec{r}) \qquad (13)$$

Now it is possible to calculate the required elevation and yaw angles as independent from the 3-axes orientation in 3D space:

$$\theta_t=Arctan\left(-\frac{r_{body,z}}{\sqrt{r_{body,x}^2+r_{body,y}^2}}\right) \qquad (14)$$

$$\varphi_t=Arctan\left(\frac{r_{body,y}}{r_{body,x}}\right) \qquad (15)$$

4.3.2. Control model

Target relative angular position $(\theta_t,\ \varphi_t)$ calculated above is the most important input for the calculation of desired angular maneuver command $(\vec{\gamma}=[\gamma_x\quad\gamma_y\quad\gamma_z]^T)$. The purpose of the control is to minimize these target angles. However, since commonly the target is far from the aircraft, reaching the target altitude and maintaining ($\theta_t \to 0$) is the method used in this study.

Yaw-control is applied in order to decrease the φ_t angle that shows how much left or right (or even behind) the target is. It requires the calculation of yaw-angular acceleration that is driven by tail rudder or thrust vectoring. Applied feedback command is simply $\gamma_z = \varphi_t$.

In the fast-catch control algorithm however, there is no need for special altitude treatment simply because the control for the elevation angle is adequate. Altitude control, on the other hand, is done by calculation of required $a_{NED,z}$ acceleration for nullification of the r_z value that means the target altitude has been matched. For the purpose, using a critically damped mass–spring–damper system analogy, a PD control model is enforced yielding a larger acceleration to reach a far target point and a retarder negative acceleration when it is closer:

$$a_{NED,z} = r_z {c_a}^2 - 2c_a V_{NED,z} \qquad (16)$$

In the approach, $2c_a$ damping constant depends upon the characteristics of the aircraft and the sharpness of the requested maneuver. Acceleration $a_{NED,z}$ (which is updated for each Δt) is applied until r_z and $V_{NED,z}$ approaches to zero. Considering this, it is required to consider the added force in plumb direction for the calculation of command for the applied pitch moment:

$$\gamma_y = -\frac{m\, a_{NED,z}}{\frac{1}{2}\rho \left|\vec{V}_{air,body}\right|^2 \left(\frac{dC_L}{d\alpha}\right) c_c} \qquad (17)$$

c_c coefficient is for the calculation of vertical component of the added lift:

$$c_c = cos(\theta_y - cos\alpha_1) cos\theta_x \qquad (18)$$

Air density (ρ) and the lift-curve slope (ratio of lift coefficient to angle of attack, $dC_L/d\alpha$) is assumed to be known for the flight conditions and aircraft.

Calculated γ_y-control represents the elevator effect and pitch-thrust. In this approach, the altitude maintaining angle of attack naturally develops during a turning maneuver with bank.

Roll moment has to represent a bank coordinated with the required turning maneuver. Required roll control is

$$\gamma_x = \frac{\varphi_T}{2} - \theta_x \qquad (19)$$

that corresponds to the aileron control.

These are all components of the $\vec{\gamma}$ control. In order to include the steepness of the maneuver a coefficient is used: $0 < k_M \leq 1$. These controls are also performed using a nearly critically damped mass–spring–damper system analogy, just like in elevation control:

$$\vec{\dot{\omega}} = k_M {c_\omega}^2 \vec{\gamma} - 2c_\omega \vec{\omega} \qquad (20)$$

Results are the angular acceleration values for the following time step.

4.3.3. Time Increment Calculation

Time step mentioned above for the time marching simulation has to be determined. Even though constant time increment is much simpler for the simulation, for the case of almost constant translational speed and orientation, it is important to increase the duration for achieving higher computational efficiency. For the purpose of optimizing time step Δt, the following procedure is developed:

Next step angular differences in the body coordinates are

$$\vec{\delta} = \vec{\omega}(\Delta t)_{new} + \vec{\dot{\omega}}\, (\Delta t)^2_{new}/2 \qquad (21)$$

These differences (increments) are required to be held within small tolerances ($\vec{\epsilon}$), so that the value of $(\Delta t)_{new}$ corresponding to the largest δ_i angular difference bounded as

$$|\delta_i| \leq \epsilon_i, \qquad i = x, y, z \qquad (22)$$

and satisfying

$$(\Delta t)_{min} \leq (\Delta t)_{new} \leq (\Delta t)_{max} \qquad (23)$$

for predefined $(\Delta t)_{min}$ and $(\Delta t)_{max}$ values is assigned as the next step's Δt.

Restrictions for largest angles of attack and roll angle should also be included in the control and calculations.

4.4. Force, moment and acceleration calculations

Aerodynamic forces ($\vec{F}_{aero,body}$) are calculated for the aircraft model depending on the angle of attack, geometry, dynamic pressure, Mach number, and altitude.

Magnitude of the thrust for the missiles is defined as $\left|\vec{T}\right| = T_{max} k_{throttle}$. Its value is zero for the no-thrust missiles and bullets. Thrust vector can be calculated as

$$\vec{T} = [T_x \quad T_y \quad T_z]^T \qquad (24)$$

$$T_x = \sqrt{\left|\vec{T}\right|^2 - T_y^2 - T_z^2}\ ,$$

$$T_y = \frac{\dot{\omega}_z}{\tau_{p,x}\, I_{zz}},$$

$$T_z = \frac{\dot{\omega}_y}{\tau_{p,x}\, I_{yy}}$$

where $\vec{\tau}_p$ is the thrust center based on the center of the gravity of the body and [I] is the moment of inertia matrix.

Moments on the body are:

$$\vec{M}_{body} = [I]\vec{\omega} + \begin{bmatrix} 0 \\ T_z\tau_{p,x} \ + T_x\tau_{p,z} \\ -T_y\tau_{p,x} \ + T_x\tau_{p,y} \end{bmatrix} \quad (25)$$

Total acceleration is calculated via Newton's 2nd Law with the addition of gravitational acceleration to the contributions from the calculated forces:

$$\vec{a}_{NED} = \{Body \rightarrow NED\}\left(\frac{\vec{F}_{aero,body}+\vec{T}}{m}\right) + \vec{g} \ (26)$$

5. MISSILE AGENT MODEL BEHAVIORS

The ammunition models are described in an agent architecture. There is a reasoning engine that is capable of operating predicates and a knowledge base. The property factor allows us to define our agent model as an intelligent agent [9], [10]. Agent behaviors are defined in the AdSiF behavioral framework, and the predicates are designed in accordance with the AdSiF logic programming structure.

The missile model life cycle begins with an ignition order and is completed as an explosion either catching the target or failing. Phases of a missile are planned as successive options, using 1) unthrusted-unguided, 2) unthrusted -guided, 3) thrusted-unguided (rocket), and 4) thrusted-guided as successive options. The transition condition between consecutive phase definitions is user defined. Phase transition condition, engine release condition, fuse type and seeker head on/off condition; 1) reaching a defined coordinate, 2) reaching a defined elevation, 3) approaching to the defined distance to the target, 4) truth value of a defined predicate; for example, deciding whether the target is missed, deciding if the target is falling, etc., and 5) expiring the duration defined for the phase. User defined functions and logical expressions can be used to encompass all of these condition definitions. In this sense, it can be defined as a condition in which the return value of a function belonging to a model is compared with a model attribute, and the parameter value verifying a defined predicate is compared with another function return value.

Missile behaviors are covered under five headings. These are defined as flight movement, phase change, explosion, target information update, and degree of freedom change. In the subsections, these behaviors are examined with the AdSiF behavioral design.

5.1. Flight Behavior

The command and control system that manages the firing weapon system decides to engage a target and sends a fire order to the weapon system. The weapon system that receives the fire command sends fire order to the missile by *"fire"* event that consists of target information. Since missile is in *Ready* state (the behavior with the Ready state is defined as the initial behavior) *Bh_Fly* behavior and *Bh_FlyDurationControl* behaviors are initiated. Flight behavior is a cyclical behavior and *Fly* state *Af_GetTimeincrement* function is used to calculate the duration the missile stay in the state. Output function at state exit *Af_Fly* function is executed for the duration exactly as long as it requests. If less duration than it requests is granted, then the same function is executed as *external* transition function for the duration specified and as a result in both cases the missile changes its position by the given duration. With the function, as basically seen in Section 4.2, Section 4.3 and Section 4.4 the new position calculation is executed with the angular accelerations, forces, moments, and distance traveled. The behavior is cancelled if the fuel is depleted (CancelCond: NoFuel). Whenever the behavior *Bh_Fly* is cancelled the behavior *Bh_explosion* (Activate::Canceled) is activated. As seen from the Figure, the event fire triggers the behavior *Bh_FlydurationControl* as well. The behavior Bh_FlyDurationControl is activated at the same time with the behavior *Bh_Fly*. It controls how long the behavior Bh_Fly is kept active, in other words, how long the missile is allowed to fly. The behavior *Bh_FlyDurationControl* is kept active in the state *Expired* for duration determined by an attribute named as *expireDuration.* On the exit phase, the state *Expired* cancels *Bh_Fly* (note that when the flight behavior is canceled *Bh_Explosion* is activated). The same state activates the behavior *Bh_Protect* in entry phase. When this behavior is activated, it turns the missile Explosion Protection on by entry phase function Af_SetExplosionOff and turns again off at the end of the duration that is assigned by the attribute named *CollisionProtectionDuration*. This prevents any explosion if there be any collision during this time period to save the weapon system firing the missile.

Bh_Explosion behavior broadcasts an event named *effect* at exit phase of the state *Explosion*. Depending on the point where the explosion occurs and the type of warhead, an effect is propagated by the event. This effect is determined as temperature, pressure or clusters spanning around the effect point and the effect level on each physical entity is calculated based on distance from the effect point using a theoretical distribution function or an experimental discrete distribution or any other user defined function. An exploded missile remains in *Exploded* state for infinite time (Figure 3).

5.2. Phase Transition

After the missile is launched, it passes to its first phase that is defined in its phase set, it headings towards its target. A missile warhead is either thrusted or un-thrusted, or free-fall, and either guided or unguided. Phase transition to the next one depends on the phase transition condition. Figure 4 defines configuration definition interface. As seen from the figure, using the interface, missile is defined as 3, 4, and 6 DOF or a table based simplified model. Similarly, phase series that is prepared by phase selection guidance algorithm selection, transition condition selections between phases, engine selection and customization the engine is seen in phase list interface element on the configuration interface.

The example is given in Section 6 consists of a phase series with three phases and an engine. In the example, the first phase is a non-guided, thrusted phase and the phase duration is determined by the normal distribution which the average and the standard deviation are equal to 5 and 1, respectively. The second phase in the phase of shutting down the engine and using the freefall phase of CLOS guidance algorithm. It comes to an end and passes to the next phase, when the defined altitude is below 3000 m. At the last phase that has guidance and propulsion, the engine is turned on again, the target continues tracking and it is only be ended by explosion. On the fuse selection tab of the missile configuration interface, the convergence with approximation distance, hit, elevation, position, and user defined conditions are presented as fuse activation selection options. In fact, user defined fuse activation condition covers all types of fuse activation conditions. On the engine selection tab, time and conditions to drop any of the missile engine is defined based on conditions similar to the phase transition conditions. Because dropping an engine changes the missile total weight, it directly affects missile dynamics. On the sensor tab there are timelines for being active and inactive for the seeker head.

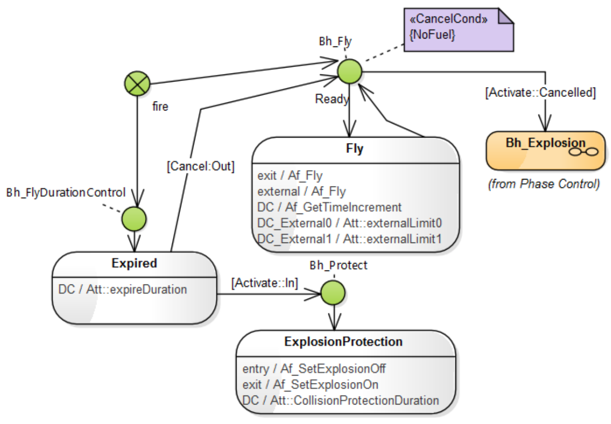
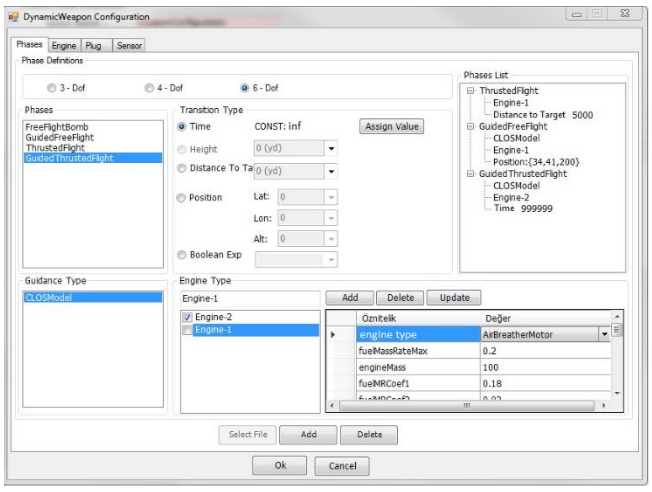


Figure 3. Flight Behaviors

After the missile ignition, phase transitions and fuse controls are handled by model behavior scripts. Phase transition behaviors are shown in Figure 3**Hata! Başvuru kaynağı bulunamadı.**. The flight behavior started by fire event makes the behavior of phase management started, as well. The activation function assigns the first phase of the behavior and in the entry phase of *"Next Phase"* state the behavior *Bh_PhaseTransition* is activated. The behavior *Bh_PhaseTransition* is cancelled by the cancel condition (*CancelCondition:: EvaluatePhaseExp*) when the next phase transition condition is satisfied and this activates *Bh_NextPhase*. The last phase is expected to be defined as infinite time by the user. After all phases are completed properly, the behavior is cancelled and the missile is exploded because the behavior *Bh_Explosion* is activated when the behavior is canceled.



( ) 3 - Dof ( ) 4 - Dof (●) 6 - Dof

Phases:
FreeFlightBomb
GuidedFreeFlight
ThrustedFlight
GuidedThrustedFlight

Transition Type:
(●) Time CONST: inf
( ) Height 0 (yd)
( ) Distance To Ta 0 (yd)
( ) Position Lat: 0 Lon: 0 Alt: 0
( ) Boolean Exp

Phases List:
ThrustedFlight
- Engine-1
- Distance to Target 5000
GuidedFreeFlight
- CLOSModel
- Engine-1
- Position:{34,41,200}
GuidedThrustedFlight
- CLOSModel
- Engine-2
- Time 999999

Guidance Type:
CLOSModel

Engine Type:
Engine-1
[✓] Engine-2
[ ] Engine-1

Öznitelik	Değer
engine type	AirBreatherMotor
fuelMassRateMax	0.2
engineMass	100
fuelMRCoef1	0.18

Figure 4. Missile Configuration Interface

All definitions done on the interface has a behavioral counterpart. That means it is possible to make all these definitions in run time designing a set of behaviors configuring the missile.

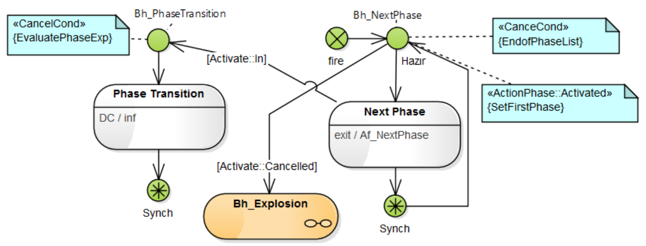


Figure 5. Phase Transitions Management

5.3. Explosion Behavior

The missile explodes in four ways. The first is to trigger the behavior of *Bh_FlyDurationControl* which triggers the explosion behavior by limiting the duration of the missile on the air. The behavior activates *Bh_Explosion* behavior at the exit phase of the state "*Expired*" (Figure 3). The explosion behavior is activated in general way, any time the fly behavior is cancelled, the explosion behavior is activated. The second activation method is the conditional activation. Anytime the activation condition Targetstalling is satisfied or it is decided the target is dead (TargetDead), the missile explodes itself. In this sense, the explosion behavior depends on satisfaction of an activation condition. The predicate detection is defined in the form as given below;

detection (Sensor, Target, Target X Pos, Target Y Pos, Target altitude, Time)

With the detection information from the missile seeker, if present, the ground-lighting radar, the current detection information is added to the knowledge base. The information to be kept in the knowledge base in the structure given below through the declaration period that is defined as 20 seconds and the information that is older than 20 seconds is deleted. The predicate definition behavior and the explosion behavior of the missile are seen in Figure 6**Hata! Başvuru kaynağı bulunamadı.** and Figure 7**Hata! Başvuru kaynağı bulunamadı.**, respectively.

```
stalling(Target,true):-detection(Snsr,Trgt,_,_,PosZ0,
T0),
        detection (Snsr, Trgt, _, _, PosZ1, T1),
        detection (Snsr, Trgt, _, _, PosZ2, T2),
        F0 is PosZ2-PosZ1,
        F1 is PosZ1-PosZ0,
        T2 > T1, T1 > T0,
        F0 < 0, F1 < 0.
stalling (_, false).
targetFalling(Predicate::In::FieldID=1==true ||
                              Af_TargetDead ())
```

Figure 6. Proposition Structure of the Missile Self Destruction Decision

The predicate decides that the target being tracked is falling, when the target decreases its altitude in last three detection time points. If the target changes its altitude intentionally, this will be interpreted as a deceptive maneuver. Thus, the target is seen as a deliberative agent that has a goal to achieve and a plan defining how to do that.

During simulation execution, it is possible to enhance a new predicate to make decision making more powerful. In this sense, it can be said that the model has learning capability in addition to reasoning capacity [6].

The detection flow is interrupted for the duration given as a duration of the state "TargetUpdate" by the attribute *expireUpdateDuration* (seen in Figure), the explosion behavior is activated (*Dv_DetectionTime* behavior).

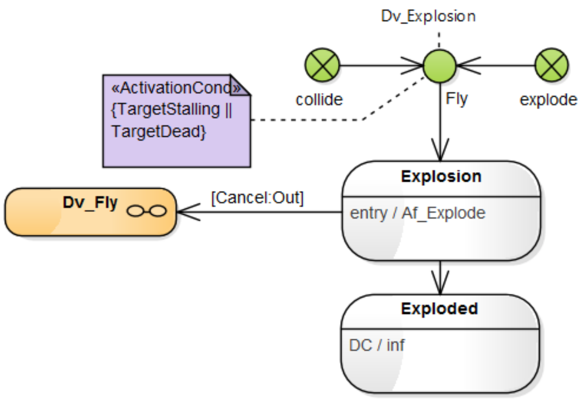


Figure 7. Explosion Behavior Diagram

Finally, during the flight (*Fly* is the first state), the behavior is activated any time in the state Fly the collide event or the explode event are received. While the "*collide*" event is broadcasted in the case of a physical collision with any other objects, the event *explode* is forwarded by the user interaction or any other entity.

5.4. Target Update Behavior

Having up-to-date target information is important for an accurate target tracking. With the tracking algorithm, the direction is channeled towards the up-to-date target position. The event *detection* received from sensors activates the behavior *Bh_UpdateTarget* if the missile is in the state *Fly*. Whenever the behavior is activated, it cancels the time counter behavior that triggers the explosion behavior when the target update is interrupted for a defined period of time, and restarts it at the end of the update (*Bh_DetectionDuration*). Thus, after every target information update, until the next update, the counter is started. The behavior diagram is shown in Figure 8**Hata! Başvuru kaynağı bulunamadı.**. At *UpdateTargetInformation* state entry phase, the position and the orientation information that are belongs to the target assigned to the missile is updated by the event detection received. Duration of the state UpdateTargetInformation is determined by the attribute *expireUpdateDuration.* During simulation execution, any change on the attribute value affects the state duration directly.

5.5. Changing the Degree of Freedom

During the flight of the missile, there is a flexibility of changing the degree of freedom. There are two purposes in providing this flexibility. The first is to provide different resolution levels in different scenarios. The second is to provide detailed enhancement in the desired phase of the simulation execution to have finer trace records by switching to higher degrees of freedom. As seen in the flight behavior, the time step is determined by the missile at the moment of execution and for each model and every missile object it is produced as different values. This is described as dynamic frequency execution. Degree of freedom change behavior is activated by any other model in the scenario, or by the *changeDOF* event sent by the user interaction, or by the provision of a user-defined condition. Changing the degree of freedom can only be activated by the event if the missile is in *Fly* state. There is no requirement for this in conditional activation (Figure 9).

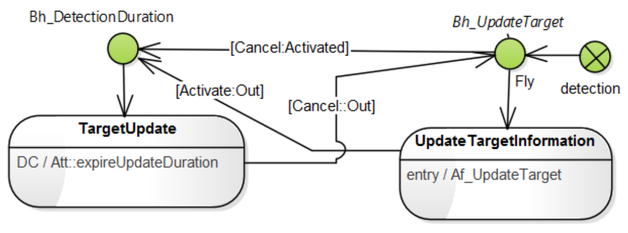


Figure 8. Target Update Behavior

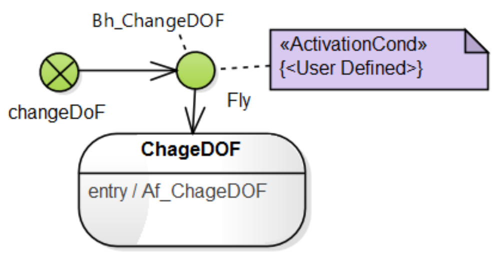


Figure 9. Change Freedom Behavior

5.6. Intelligent Target Tracking Behavior

During target following, the time synchronization of target with the missile is provided according to the user-defined conditions. Due to the use of dynamic frequency, target and the missile can demand different time steps. Often, the missile requests smaller time steps than targets and this differentiates the resolutions. As an example, while a missile updates its position at steps of 300 ms, an aircraft may move by 5 seconds steps. In this case, the events occurring within the time steps of both models are processed by the external transition feature (activity scanning). The synchronous execution is provided by changing the external transition interrupted time step, which allows the missile to approach the target at a defined distance or at the end of the defined time, so that both behaviors can proceed at the same time step. *DC_External0* that is seen at the state Fly determines the usage lower limit for the time step granted and, in the example, it is set by the attribute *externalLimit0*. As an example, when at the target model, the *externalLimit0* attribute is set to one (1), the time step such as 300 ms granted via the missile time request is consumed by the target and the corresponding position step updates are performed. Because these changes are performed on an object-by-object basis, update operations do not need to be executed for all entities in these narrower steps. The approach is more efficient than standard time slicing approach. Errors encountered in the simulation such as double precision, and the fourier transformation are thus reduced [6], [11].

6. EXAMPLE: TARGET ENGAGEMENT SCENARIO

In the scenario, an execution is performed in which time is followed by the same resolution, after a condition in which a missile is fired on an air platform and the corresponding condition is met, where both entities proceed independently from one another until a certain distance. An air platform was detected by the radar and an order of engagement was sent by the commander following its detection (Figure 10**Hata! Başvuru kaynağı bulunamadı.**). The missile was fired towards the target and flew for 5 seconds with 3 degrees of freedom. After the 5th minute of the movement, it passes 6 degrees of freedom, synchronizing its time with the target aircraft after approaching the target 15000 m. As seen in Figure 11**Hata! Başvuru kaynağı bulunamadı.**, the target time steps have shrunk since this point and have begun to progress with the same precise time steps as the missile.

Until at the time of 530.514th second of the scenario execution, the time requests and advances of the air platform and the missile are independent from each other (Figure 11). Once the synchronization initiates, even though the time step is still independent, the entity at the request of the great time step, here is the presence of the airplane, completes its time by operating the smaller time advances. Approach Operation provides faster execution and higher accuracy.

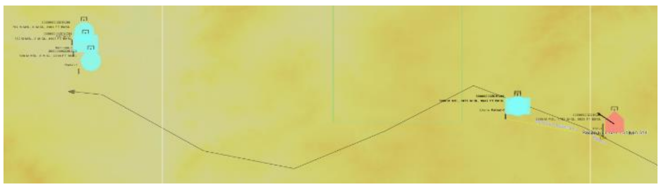


Figure 10. Scenario Screenshot

With the presented aerial model, 6 DOF simulation of a general aircraft is performed and the developed software is tested for different extreme conditions. In the sample test depicted in Figure 12, an aircraft at 1000 m altitude headed south, engages, chases and catches a target at 3000 m altitude in northeast. In the graph on the left, while bank angle is restricted by 30º, rolling and elevation are adjusted by the control model in order to reach the target altitude. In the graph on the right the aircraft initially cruising at N33-E33.4 makes a maneuver and reaches the target coordinates, N34-E34.4.

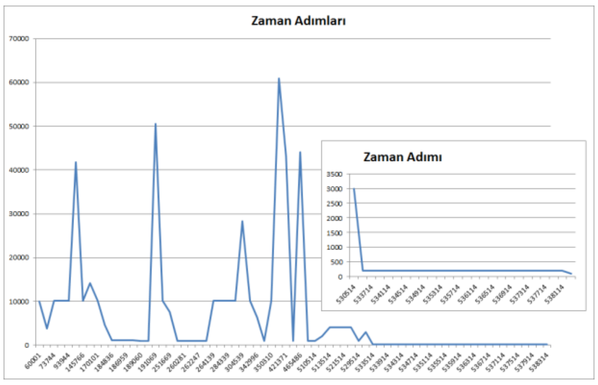


Figure 11. Time Steps

7. CONCLUSIONS

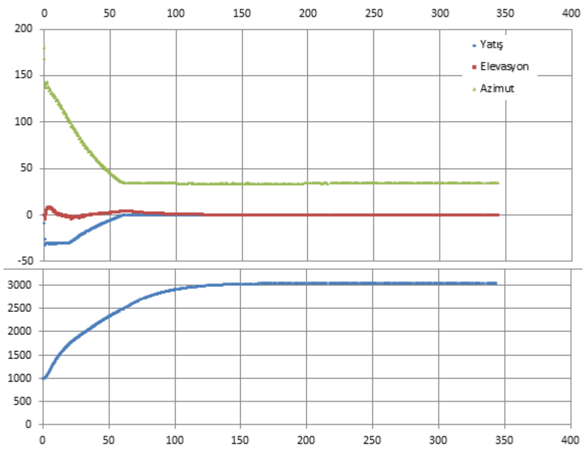
Altitude and elevation controls are conducted in a cascade structure, which is why the correct definition of system parameters is very important. In performed verification scenarios, required maneuvers to catch the constant and moving targets are obtained by the tracking model integrated in the simulation. Aircraft model performs the coordinated maneuver controls required to head towards to and reach the target while dynamically adjusting the time steps in order to dramatically decrease the computational load while slightly increasing the allocated memory.

The dynamic frequency operation support provided by AdSiF in simulation provides a solution for the variable step-wise integration algorithm [11], which is often needed in continuous event simulations. With the presented solution, variable time steps (dynamic frequency) in the operation of the simulation are determined by the specific dynamics (mathematical model) of the continuous event model. Therefore, instead of the use of a globally predefined constant time step, utilization of variable time steps calculated based on the control and simulation model minimizes the computational cost. Every continuous event model in the simulation determines its own time step being independent of other models' dynamics.

AdSiF executes simulation scenarios by a scripting language. Since the language is an interpreted language, it allows to add new behaviors and predicates for decisions in run time and being based on scripting provides flexibility in modeling. For our example, the explosion predicate structure for different missions can be defined differently for different scenario objects, and the fact that missile phase sequences are identifiable in the structure provides the flexibility to produce different ammunition configurations without coding.

Missiles and aircrafts are modeled as agents with reasoning capability. How a missile is configured in simulation run time is showed, especially, changing its degrees of freedom. An opportunity is given to define conditions and configuration items using logical predicates. Since logic programming and script parts of the simulation model are not required compiling, it's possible to enrich algorithms in simulation run time.

Being dynamically configured provides very flexible simulation opportunity for modelers. It supports modelers to configure a dynamic missile model in run time depending on scenario conditions even the constraints determined based on other components in the scenario. For example, the missile configuration can be changed depending on the status of a radar located on the land or any other platforms that there is no direct connection with the missile. Beyond conditionally configuration change, user interactions are also supported. Users can change missile configurations in run time by event interactions.



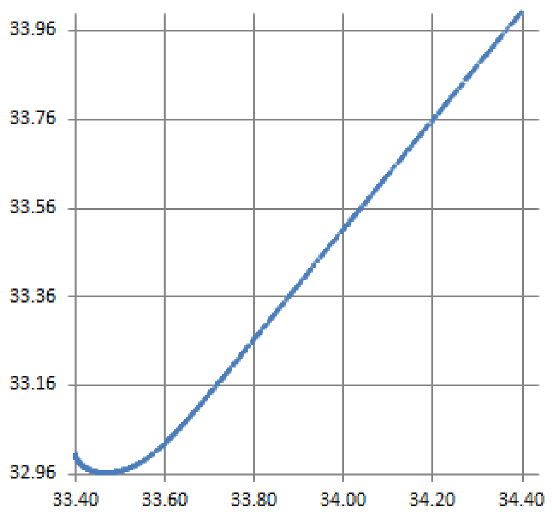


Figure 12. Simulation of an Air Platform

In addition to missile behavior configuration, turning on and turning off times for seeker headings, fuse types, engine dump conditions, and time schedules for all these definitions can be changed in run time by conditionally and by user interactions.

While the solution provides a strong and flexible modeling capability, decision making algorithms need high computational effort.

ACKNOWLEDGMENTS

This study is supported by Agena Information and Defense Technologies Ltd. (www.agenabst.com, https://hocaoglumf-projects.weebly.com/).

8. RESOURCES

[1] D. G. Hull, *Fundamentals of Airplane Flight Mechanics*. Heidelberg, Germany: Springer-Verlag Berlin, 2007.

[2] “Military Handbook Missile Part I Simulation One Surface-To-Air,” 1995.

[3] J. Roskam, “Airplane Flight Dynamics and Automatic Flight Controls – Part II,” USA, 2003.

[4] M. F. Hocaoglu, “AdSiF : Agent Driven Simulation Framework,” in *Hunstville Simulation Conference -HSC2005*, 2005.

[5] M. F. Hocaoğlu, “AdSiF : Agent driven Simulation Framework,” in *USMOS 2011- National Defense Application and Modeling & Simulation Conference -(in Turkish)*, 2011.

[6] M. F. Hocaoğlu, “AdSiF: Developer Guide.” Agena Information & Defense System Ltd. www.agenabst.com, Istanbul, 2013.

[7] M. F. Hocaoğlu, “Aspect Oriented Programming Perspective in Agent Programming,” *USMOS 2013- Natl. Def. Appl. Model. Simul. Conf. -(in Turkish)*, vol. 8, no. 3, 2013.

[8] M. F. Hocaoglu, “Aspect Oriented Programming Perspective in Software Agents and Simulation,” *Int. J. Adv. Technol.*, vol. 8, no. 3, 2017.

[9] Wooldridge, M., *An introduction to MultiAgent Systems*. John Wiley & Sons, Ltd, 2002.

[10] R. J. Schalkoff, *Artificial Intelligence: An Engineering Approach*. McGraww-Hill, New York, 1990.

[11] D. J. Murray-Smith, *Continuous System Simulation*. University og Glasgow, Glasgow, UK: Chapman & Hall, 1995.