FORMAL VERIFICATION OF MANEUVERING TARGET TRACKING

Mark Moulin* Leonid Gluhovsky† Eli Bendersky‡
IBM Research Lab, Haifa 31905, Israel

Abstract

The paper introduces the application of a formal verification (model checking) technique for improvements in the performance of maneuvering target tracking. Formal methods is a well-established completely automatic approach to system verification which analyzes a system model behavior against desired properties. Formal methods provide a full-coverage verification of the system model behavior. In this paper, we look into the problem of tracking a maneuvering ballistic missile, based on bearings-only measurements. The interception parameter estimations are essential for the implementation of efficient guidance laws. Assuming knowledge of a target model, a standard Extended Kalman Filter was applied. This filter is known to produce biased estimations of the range and range rate. The applied estimation debiasing heuristic procedure consists of multiplication of the state covariance matrix by a scalar fudge factor at every sampling time, and the use of this modified state covariance matrix in the covariance update equation. The value of the scalar fudge factor was obtained by using the RuleBase formal verification engine - a formal verification tool developed by the IBM company. We also checked the consistency and capturability properties of the true proportional navigation guidance law. The numerical results of a realistic representative case are presented.

Introduction

In this work, we apply a formal methods (model checking) technique[1] to verify free maneuvering missile target tracking. A standard closed-loop tracking system with incomplete state measurements comprises a target acceleration model, a state estimator, and a tracking controller - guidance law.

The guidance law is the principal tracking function. It produces the command signal - missile acceleration. The control theory views the missile guidance system as a compensation network placed in series with engagement process in order to accomplish an interception. The guidance system determines the best trajectory for the missile, based on its knowledge of the missile's capability, the target's capability, and the desired objectives. The decision process consists of the target motion estimation, generation of the guidance command for optimal interception, and the control of the nonlinear uncertain dynamics of the interception process.

Most of the applied guidance laws belong to the Proportional Navigation guidance laws family,[2,3] which contains, among others, two large groups of Pure Proportional Navigation (PPN) and True Proportional Navigation (TPN) guidance laws. In PPN, the missile acceleration is applied normal to the missile velocity vector, whereas in TPN, it is applied normal to the missile-target line-of-sight (LOS).

The performance of guidance laws is usually examined under the realistic considerations on target maneuvers. The intelligent target tries to maneuver when a missile is on a collision course. Usually, these maneuvers consist of circular movements with target acceleration assumed to be constant and applied normal to the target velocity vector. Target acceleration also can be applied normal to the LOS with the time-variated acceleration magnitude. The possible profiles of such maneuvers are different, and the applied guidance law must be sufficiently robust to deal with all of these maneuvers.

The proportional navigation guidance laws are based on the estimated LOS and range rates. Traditionally, for modeling and analysis clarity, the state measurements are assumed to be incomplete. The only available data are the Gaussian noise corrupted bearing-angle measurements. The nonlinear tracking process usually employs an Extended Kalman Filter (EKF) as a state estimator.[4,5] This filter determines the system's present state by integrating estimation results with measurement noise parameters and system model knowledge. Usually, it performs a state estimation in the Cartesian coordinate frame, and updates the system in the spherical coordinates. The filter advantages from the linearity in both coordinate systems at the cost of the bias in

* research staff member
† research staff member
‡ student research staff member

range and range rate estimations.[5–7]

The closed-loop tracking system evolves in continuous time with discrete jumps at the EKF update instances. The methods developed for purely continuous, or purely discrete, systems are not directly applicable to the analysis of the entire behavior of such a hybrid system.[8] In the context of hybrid systems, one may expect to encounter properties both from the continuous and from the discrete domain, with some interaction between them. A considerable challenge is to verifying hybrid systems that involve a significant amount of nonlinear continuous dynamics.

An initial step in hybrid system verification is to make a reasonable approximation (discretization) of the nonlinear dynamics in order to reduce the possibly infinite state space system into a finite state space system. Depending on time constants of the given system, a practical approximation can be done, such that the reachable state of the approximated system also contains the reachable state of the given system. Then, model checking techniques can exhaustively search through all reachable states of this finite system. A weakness of the conservative approximation is that it may require a large number of samples, and both the memory requirements and the computation time of a formal verification tool may soon become impractical. The number of states to examine is huge in applications of practical interest, such as airspace control systems.

Usually, a hybrid system is examined through simulations. Still, the simulation tools can only show how the system behaves under a particular input vector. It is not always feasible to verify the complete behaviour of the system by the given set of simulations under all possible input vectors, and the full coverage of system behavior cannot be achieved in simulation mode.

Formal methods (model checking)[1,9] provide a full-coverage verification of the interception process performance and the robustness of the applied guidance law. Formal methods are able to verify a system using the same difference equation representation of the system that is used by simulation tools, such as Matlab/Simulink. Formal methods examine if all possible states of a logical model of the system satisfy or fail to satisfy the particular temporal properties.

Model checking verifies the specified temporal property using temporal logic.[1] Temporal logic is a standard Boolean logic with added special temporal operators to describe future events. Hence, temporal properties describe the relationships between Boolean expressions over time. Model checking deals with two types of properties: the safety property and the liveness property. The safety property states that something bad (undesirable) never happens. This property is suitable for checking that tracking errors are always lower than some upper boundary. The liveness property states that something good eventually happens. This property is useful to check the convergence of the system. All these properties successfully describe the complete behavior of the reactive system. They are verified automatically by special programs - model checkers. For this purpose, the system under test is represented as a logic circuit of the basic logic elements, such as logic gates and flip-flops. Then, the model checker processes this logic circuit in three phases: environment modeling, specification writing, and the verification itself.

An environment model is a description of the all legal input sequences to the system. While test suites used in simulation technique describe a subset of the legal input sequences one at a time, an environment model simultaneously describes all and only the expected behaviors of the system's inputs.

The specification writing process consists of translating the properties of the system and its performance requirements into a set of formal properties represented in a standard formal specification language.[10,11] For example,

- $$\textbf{always}(change\_in\_velocity \rightarrow \textbf{next}\ change\_in\_position) \quad (1)$$

  is a temporal property stating that whenever (**always**) the signal (state) *change_in_velocity* is asserted, then ($\rightarrow$) at the next cycle or time instant (**next**), the signal (state) *change_in_position* is asserted.

- $$\textbf{always}(change\_in\_velocity \rightarrow \textbf{next}[k]\ change\_in\_position) \quad (2)$$

  is a temporal property stating that whenever (**always**) the *change_in_velocity* is asserted, then ($\rightarrow$) after the k cycles (**next**[**k**]), the *change_in_position* is asserted.

- $$\textbf{always}(change\_in\_velocity_1, change\_in\_velocity_2 \rightarrow \textbf{next}[k]\ change\_in\_position) \quad (3)$$

is a temporal property stating that whenever (**always**) the *change_in_velocity*$_1$ is asserted, and on the next cycle the *change_in_velocity*$_2$ is asserted, then ($\rightarrow$) after the k cycles (**next**[**k**]) from the assertion of the *change_in_velocity*$_2$, the *change_in_position* is asserted.

Once the environment has been modelled and the specification coded, the model checker is invoked on the logic circuit. The check is performed as an exhaustive search, which is guaranteed to terminate because the model (circuit) is a finite state transition system. The model checker provides output information about the system's validity: demonstrating a witness if a property satisfied, or a counterexample if the property is unsatisfied. This information is very useful in finding sources of system instability, or defects in controller design.

The main limitation of model checking is that it needs to build a complete state graph for the model. In general, the number of the states in this graph increases exponentially with the size of this model. In order to avoid the exponential explosion of the state number, the Bounded Model Checking (BMC) technique[12] limits number of states through the explicit declaration of the time interval on which the system have been verified.

The BMC constructs a specific Boolean formula from the system under test. This formula can be satisfied if and only if the underlying state transition system can realize a finite k sequence of state transitions (cycles) that reaches certain states of interest. If such a path cannot be found at a given length k, then k can be increased to find the interest path. Note that when a check is performed for a specific path of length k, all paths of length k are examined. The Boolean formula formed is examined in automatic mode[12] and, if a satisfying assignment for the formula is found, that assignment represents a witness for the path segment of interest. Since the model is built only for a bounded time interval, the exponentially growing model is avoided, and a large system can be checked very quickly. It is worth noting that the applied method is generally incomplete, which means that an eventual true or false determination for every property is not always guaranteed. This is because the length of the propositional formula is subject to satisfiability solving that grows with each time step and greatly inhibits the capacity of to find a long witness (or counterexample) for a large k, and the ability to automatically check all possible paths.

In our work, we employ formal verification in the BMC mode to verify the properties of the free maneuvering target tracking. The described below system is converted from an analog to a digital system and then synthesized to a digital logic for verification.

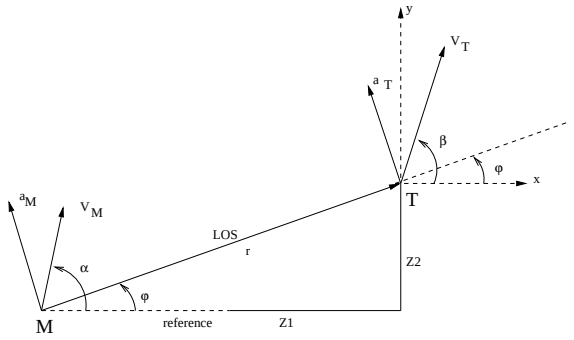

Figure 1: Geometry of the planar tracking problem.

Description of the Free Maneuvering Target Tracking Model

The geometry of the illustrative example of the maneuvering target pursuit is depicted in Figure 1. Both the target **T** and the pursuing missile **M** are assumed to be point masses moving in a plane with velocities V_T and V_M. The position of the target T is assumed to be the center of the relative coordinate system. The polar system equations of motion are given:

$$\begin{cases} \dot{r} &= V_T \cos(\beta - \varphi) - V_M \cos(\alpha - \varphi) \\ r\dot{\varphi} &= V_T \sin(\beta - \varphi) - V_M \sin(\alpha - \varphi) \end{cases} \tag{4}$$

where LOS is the instantaneous missile-target line-of-sight (LOS), a time-variant vector from the pursuer to target; r is the range i.e., the length of the LOS; φ is the bearing angle i.e., the angle between the LOS and the reference line; α is the missile heading angle; and β is the target heading angle. The missile a_M and target a_T accelerations are applied normal to LOS and govern the following input dynamics described by Eqs.(5-6)[13]

Target dynamic:

$$\begin{cases} a_T = \frac{b}{r\dot{\varphi}} \\ \dot{V}_T = a_T \sin(\beta - \varphi) \\ \dot{\beta} = \frac{a_T}{V_T} \cos(\beta - \varphi) \end{cases} \tag{5}$$

where $b > 0$ is a positive constant.

Missile dynamic:

$$\begin{cases} \dot{V}_M = a_M \sin(\alpha - \varphi) \\ \dot{\alpha} = \frac{a_M}{V_M} \cos(\alpha - \varphi) \end{cases} \tag{6}$$

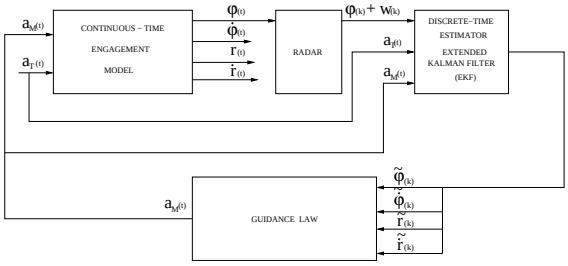

Figure 2: A block diagram of the interception process.

The employed guidance law for the engagement model given by Eq.(4) is a True Proportional Navigation guidance law[14]

$$a_M = -\lambda \dot{r} \dot{\varphi} \tag{7}$$

where $\lambda > 0$ is a navigation constant. This guidance law uses the EKF estimations of bearing-angle and range rates, since it is assumed that only $w(t)$ white noise corrupted bearing-angle φ seeker data are available:

$$y(t) = \varphi + w(t) \tag{8}$$

In order to apply the EKF as a state estimator, we describe the missile target relative motion Eq.(4) in a Cartesian inertial reference system (X, Y) by the following linear time invariant model:

$$\begin{cases} \dot{z}_1 = z_3 \\ \dot{z}_2 = z_4 \\ \dot{z}_3 = a_{T_x} - a_{M_x} \\ \dot{z}_4 = a_{T_y} - a_{M_y} \end{cases} \tag{9}$$

where a_{T_x} and a_{T_y} are the target acceleration vectors, and a_{M_x} and a_{M_y} are the missile acceleration vectors in the Cartesian inertial reference system (X, Y).

The associated nonlinear measurement equation

$$y(t) = \arctan(\frac{z_2}{z_1}) + w(t), \tag{10}$$

is identical to Eq.(8).

Thus, the closed-loop hybrid system tracking system is comprised of a continuous-time guidance law; a discrete events estimator - an Extended Kalman Filter (EKF); and a free system input - target acceleration. A block diagram of the interception process is shown in Figure 2 (the standard blocks of analog-to-digital and digital-to-analog converters are suppressed in the figure).

Formal Verification of the Tracking Algorithm Consistency

Guidance Law Capturability Consistency

Capturability characterizes the ability of guidance law to ensure the capture or interception of a target. Usually, for ease of analysis, the target maneuver is considered to be a bounded piecewise continuous function. Meanwhile, in reality, actual avoidance and escape maneuvers performed by missiles involves a combination of sharp turns and straight line trajectories which are executed by switching between positive, zero, and negative lateral accelerations. Thus, it is very important to test the derived guidance law on the ensemble of different continuous and discontinuous realistic target maneuvers.

A set of all initial conditions from which a missile can intercept a target is called a capture region. It is a measure of the capturability performance of the guidance law. Usually, the qualitative analysis technique is used to evaluate the bounds of the capture region for the engagement model Eqs.(4-7).[13] The Lyapunov function analysis is used as an alternative approach.[15] In both cases, the obtained capture regions are too restrictive, and the main boundary is a maximum limit on the initial range. The standard capture equations[13] are not solvable in closed form, i.e., it is not possible to obtain expressions for r in terms of time. These inequalities ensure the capture under the right choice of the b - target maneuver parameter, and interception scenario initial conditions $\dot{\varphi}(0)$, $\dot{r}(0)$, $r(0)$.

The qualitative analysis shows that the capture region may vanish when the initial conditions or target maneuver are high. The other factors that can change the capture region shape are high perturbation in the LOS rate, saturation of command acceleration, high density process and measurement noises, and so on.[13] For example, the capture region expands with increases in navigation constant, i.e. the command acceleration gain. In the opposite situation, the increase of target acceleration decreases the capture region.

The standard employment of capturability conditions is to find the initial conditions $\dot{\varphi}(0)$, $\dot{r}(0)$, $r(0)$, that are sufficient for interception under the chosen target maneuver and final time (time-to-hit). This approach constrains the capturability and the guidance law to be relevant to some specific target maneuver.

The target maneuver considered in Eq.(5) is restricted to the application of the lateral acceleration normal to the target velocity under assumption that

the target is expected to perform evasive maneuvers to increase the probability of its escape.[13] Still the perturbations in the target acceleration parameters allow the target to perform different maneuvers in order to decrease the capture region and to mislead the designed guidance law. The restriction on target maneuvers degrades the universality of analysis, and the actual capture region can be different for perturbed parameter b.

The formal verification can examine the capturability properties more efficiently than qualitative analysis. It is able either to certify in one run the qualitaty analysis results under a bounded set of target maneuvers, or to find target maneuver that could lead to the escape of the target under the defined particular final time and initial conditions. This information can help to design robust controllers, which can take into account numerous realistic target maneuvers. Such result is hard to obtain in other way except by the long time numerous simulations.

Extended Kalman Filter Consistency

The effectiveness of the tracking system depends directly on the consistency of EKF estimations. The guidance law Eq.(7) parameters, i.e. bearing-angle and range rates, are obtained through EKF estimation. These EKF output estimations are frequently biased. The bias in EKF estimation can seriously degrade the tracking performance. The reason for the bias in EKF output estimations is the insufficient correlation between the gain and innovation sequences against all possible target maneuvers. The EKF introduces errors when the higher order terms of nonlinear state expansions have been neglected, or when the Jacobians (and Hessians) are evaluated on predicted or estimated state values when the exact values are not available. There are several ways to compensate for these errors. The first is to add a pseudo-noise term to compensate for the errors in the state prediction by using a larger modified process noise covariance in the covariance prediction equation.[5] This completely heuristic increase of the state covariance cause a state gain to be larger, thus giving more weight to the recent data. Formal verification engine easily finds the required pseudo-noise covariance matrix.[9]

Another way to decrease the EKF output estimations bias is to multiply the state covariance by a scalar fudge factor ϕ at every sampling time

$$P_\phi(k+1|k) = \phi P(k+1|k) \qquad (11)$$

The obtained $P_\phi(k+1|k)$ matrix is used in the covariance update equation. The filter has fading memory when ϕ is larger than unity.[5] Usually, the fudge factor is only slightly larger than unity. Hence, the choice of this factor turns to be a delicate, system parameter sensitive, process.

The formal methods check the following property to find a required fudge factor: consider the set of all possible $\phi \in [1 \ldots 1+\delta]$ and prove that for all these ϕ the bias takes place. The provided counterexample would give the required ϕ that reduces the bias to acceptable values.

Formal Verification of the Illustrative Realistic Free Maneuvering Target Tracking Example

Formal methods analyze only the discrete systems. To satisfy this inherited constraint, we transformed the continuous-time system Eqs.(4-9) into a periodically updated system. We described the resulting overall closed-loop sampled system in the Verilog - hardware description language.[16] This hardware description language has a powerful compiler to synthesize the descriptive model of the system into a circuit of basic logical gates. All arithmetic data were represented by 32-bit vectors.

The tracking algorithm was examined in a realistic complicate interception scenario,[7] presented in Table 1. The system properties were verified with the help of RuleBase, a formal verification tool developed by the IBM Haifa Research Laboratory. RuleBase supports the Chaff SAT solver,[17] of a BMC family solvers and made various optimizations of logic circuits in order to achieve a better performance of the SAT solver. As a hardware verification platform we used an IBM Cascades PC with a Pentium III 700 MHz microprocessor with Red Hat Linux Advanced Server Release 2.1 AS (Pensacola).

Table 1. The initial conditions of the scenario and system's parameters.

LOS angle rate	$\dot{\varphi}$	$\frac{rad}{sec}$	0.0051
LOS angle	φ	rad	-0.9600
range rate	$\dot{r}$	$\frac{m}{sec}$	-1787
range	r	m	30000
missile velocity	V_M	$\frac{m}{sec}$	1500
target velocity	V_T	$\frac{m}{sec}$	500
missile heading angle	α	rad	-0.80
target heading angle	β	rad	-0.75
navigation constant	λ		3.85

Formal verification of the interception dynamic properties

To check the performance of the interception process, we considered that the target acceleration Eq.(5) parameter b varies in the interval $b \in [500..2500]$ with granularity of 100. Then, we supposed that interception must occur after 30 seconds of the interception process. This explicit time constraint allows us to work in formal verification BMC mode with a samples number parameter $k_{fin} = 10$. The interception condition was considered to occur when the distance r between the missile and target is less than 3000 meters. The consistently chosen gains of controller (navigation constant λ) must ensure this for all perturbations of b. The usual simulation procedure to check this is to launch at least 500 Monte Carlo trials. Meanwhile, RuleBase found a counterexample in one run after verifying the following property relative to all possible perturbations of the target acceleration parameter $b \in [500..2500]$ with granularity of 100:

Property 1:

$$\{\mathbf{always}(\mathbf{for\_all}(b(k) \in [500..2500]) \wedge \wedge r(0) = 30000 \rightarrow \mathbf{next}[k_{fin}](r(k_{fin}) < 3000))\} \quad (12)$$

The counterexample shows the target acceleration profile (Figure 3) that caused the target and missile to come within 3100 meters of each other after 23 seconds of the interception process. From this run, we obtain the vector $B(0..10)$ that contains values of parameter b in every sample.

Since the designed controller has not met the requirements of the final range value, we must tune the navigation constant of the guidance law Eq.(7) according to the target maneuver obtained from the first property. The second property finds the navigation constant λ that ensures the condition $r(10) < 3000$ meters for a given target maneuver

$$a_T(k) = \frac{B(k)}{r(k)\dot{\varphi}(k)} \quad (13)$$

In this property, $\lambda \in [3..5]$ variates with granularity of 0.1, and constant $b_k = B(k)$ is fixed, according to the previous property verification results.

Property 2:

$$\{\mathbf{always}(\mathbf{for\_all}(\lambda \in [3..5]) \wedge r(0) = 30000 \wedge \wedge B(0), B(1), ..., B(k_{fin}))\} \rightarrow (r(k_{fin}) > 3000) \quad (14)$$

Figure 3 shows by the dashdot line that Rulebase found out that $\lambda = 4.5$ ensures the convergence of the range below 3000 meters.

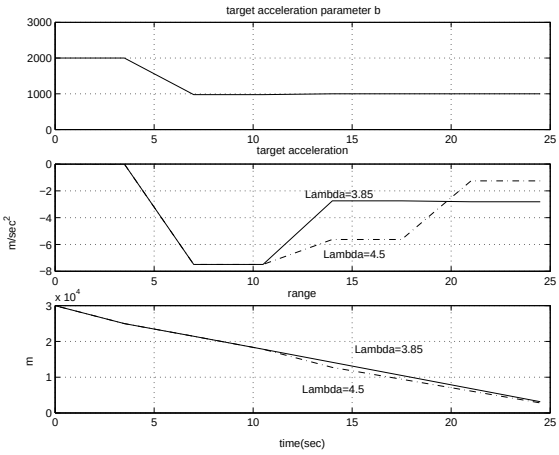

Figure 3: Properties 1 and 2: parameter b, target acceleration and range profiles.

Formal verification of the capture region properties

The capture region properties were formulated according to the qualitative analysis capture conditions.[13] In these inequalities, the initial conditions r_{bound} and $\dot{r}_{bound}$ define the capture region bounds according to time-to-hit t_{fin} value and parameter b. The task is to find the empirical capture region bounds for the varied parameter b. For this purpose, we consider the initial conditions $r(0)$ and $\dot{r}(0)$ to be higher than these initial conditions bounds. On the next stage, RuleBase tries to find target maneuvers that lead to escape of the target from this capture region after 25 seconds of interception. Similarly, to the Property 1 case, $k_{fin} = 10$ and the target acceleration parameter b varies in the interval $b \in [500..2500]$ with granularity of 100.

Property 3:

$$\{\mathbf{always}(\mathbf{for\_all}(b \in [500..2500]) \wedge r(0) \geq r_{bound} \wedge \dot{r}(0) \geq \dot{r}_{bound} \rightarrow \mathbf{next}[k_{fin}](r(k_{fin}) < 3000)\} \quad (15)$$

The counterexample finds the target acceleration profile that causes an escape of the target from the missile after 25 seconds of the interception process. The parameter $b(k)$ values in every sample have been stored in the vector $B(0..10)$. Finally, the following property

Property 4:

$$\{\mathbf{always}(\mathbf{for\_all}(r(0) \in [20000..25000]) \wedge B(0), B(1), ..., B(k_{fin}))\} \rightarrow \mathbf{next}(r(k_{fin}) > 3000)\} \quad (16)$$

finds the initial conditions from the interval $r(0) \in [20000..25000]$ with granularity of 1000 that ensure

the capture for a given target maneuver $B(0..10)$ and $k_{fin} = 10$. The obtained initial conditions and target maneuver are completely coincides with qualitative analysis capture conditions,[13] thus certifying these analytical results.

Formal verification of the EKF consistency properties

The last group of properties we checked dealt with a bias in EKF range estimations. At first, we ran the property that detects all particular target maneuvers that cause the bias. We considered the maximal accepted value of the range bias *range_bias_max* to be 3000 meters (i.e. 10% of the initial range).[6]

Property 5:

$$\{\textbf{always } (\textbf{for\_all}(b(k) \in [500..2500]) \rightarrow range\_bias\_max < 3000)\} \quad (17)$$

RuleBase easily finds a problematic target maneuver. The interception parameters are shown in Figure 4: the target acceleration and range are depicted by a solid line; biased range estimations are depicted by a dashed line.

Next, the EKF can be tuned to improve the correlation between the gain and innovation sequences by using the fading memory, or a fudge factor from Eq.(11). We reformulate the property Eq.(17) in the following way: consider the fudge factor $\phi \in [1..1.5]$ with granularity of 0.05, and state that it cannot debias the EKF output estimations according to Eq.(11).

Property 6:

$$\{\textbf{always } (\textbf{for\_all}(\phi \in [1..1.5]) \; range\_bias\_max > 3000)\} \quad (18)$$

The failure of the property Eq.(18) provides us with the fudging factor $\phi = 1.15$ that debiases the EKF estimations up to 2900 meters. The results are depicted in Figure 4 by the dashedot line. The verification protocols of all six properties are summarized in Table 2.

Table 2. The summary on the verification protocols.

#	Property verifies	Number of gates in logic circuit	CPU time(sec)
1	guidance law	128820	94
2	guidance law	126559	121
3	capturability	127119	152
4	capturability	129498	168
5	EKF	289609	283
6	EKF	293731	291

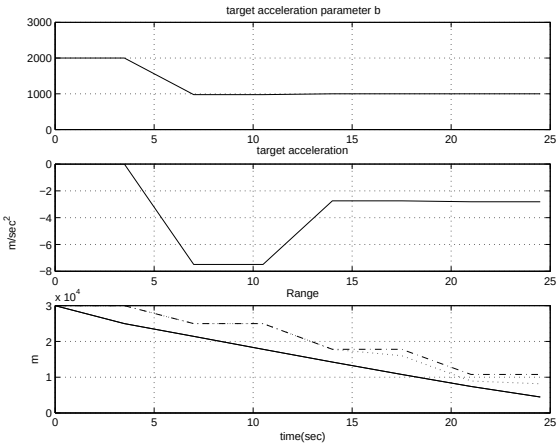

Figure 4: Properties 4 and 5: parameter b, target acceleration, range and range estimations profiles (bias range estimation are depicted by a dashed line; unbiased range estimations are depicted by dashedot line).

Table 2 shows the two most important parameters of the verification performance: the number of gates and the CPU time. The number of gates characterizes the size of the logical circuit. The length of the BMC Boolean formula depends exponentially on this number. In its turn, CPU time nearly linearly depends on the length of the checked formula. For example, the CPU time for EKF consistency checking is larger than for capturability checking. The same relations are true for the number of gates.

The dependencies between the gate number and the CPU time are not strictly linear because the BMC search[7] uses heuristic algorithms. The performance of these algorithms depends also on the complexity of the property. Thus, the Property 1 circuit has more gates than the Property 2 circuit, but the second property needs more CPU time. The second property is complicated for a BMC engine, because we required the property Eq.(14) to satisfy the concrete conditions on every cycle

$$... \wedge B(0), B(1), ..., B(k_{fin}))\}$$

As a result, RuleBase spent more time verifying Property 2 than Property 1. Such a situation is typical for BMC engines. The way to facilitate the search is to use the verification data of the relative property. RuleBase uses this mechanism during verification of Properties 3 and 4 where it needs similar CPU times in both cases. The engine successfully used the results and data from Property 3 verification to verify the Property 4.

RuleBase also uses verification data sharing for checking Properties 5 and 6. Here the CPU time grows together with number of gates. The circuit size for Properties 5 and 6 show how the inclusion of EKF expands the model. This is an inherited drawback of the formal verification approach - the model complexity increases with every inclusion of a new arithmetic operation. Even the mere matrix multiplication can lead to a size problem, i.e. the computational explosion of the verification process, and engine trouble to verify a property in adequate time.

Herein lies the cost of full-coverage verification - the formal engine capability to verify million-gates size models is smaller than that of simulation tools. The quality requirement says that formal verification needs much more selective modeling than simulation. This requirement is emphasized in the ability of RuleBase to optimize out the redundant parts of the model (logic circuit), if the modeling is not optimal. Still, this procedure needs additional computational power, and it is better to avoid it through the right modelling methodology.[10,11]

Summary

The main contribution of this work is the application of formal methods to maneuvering target estimation algorithms under a realistic air-to-air planar scenario. A novel and powerful technique is introduced to analyze the interception process behavior, and the EKF range estimation bias in particular. The verification was made in coupled properties. The first property finds the parameters that lead to inconsistency of the tracker. The second property tunes the control parameters in order to ensure the interception. The RuleBase verification engine steadily verifies these properties in BMC mode. The system implementation in Verilog customizes a floating-point arithmetic processing. At the same time, it must be admitted, that magnitudes of the interception parameters perturbations are still short of being comparable to the benchmark values of industrial control applications.[3]

In our opinion, the proposed methods are efficient for consistent checking of the hybrid and switching control algorithms, fuzzy-logic-based control, and intelligent controllers in general.

References

1. Clarke, E.M., Grumberg, O., and Peled, D.A., *Model Checking*. The MIT Press, 1999, pp. 27-87.
2. Zarchan P., *Tactical and Strategical Missile Guidance*, Progress in Astronautics and Aeronautics., Vol.199, AIAA Inc., Washington, DC, 2002.
3. Lin C.-F., *Modern Navigation, Guidance and Control Processing.*, Prentice Hall, NJ, 1991, pp.414-485.
4. Maybeck P.S. *Stochastic Models, Estimation and Control*. Academic Press Inc., 1982, pp.224-230.
5. Bar-Shalom, Y., Kirubarajan, T., and Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001, pp.371-420.
6. Moorman M.J., Bullock T.E., "A Stochastic Perturbation Analysis of Bias in the Extended Kalman Filter as Applied to Bearings-Only Estimation," *Proc. of the* 31^{st} *Conference on Decision and Control*, Tucson, Arizona, 1992, pp. 393-398.
7. Moulin, M., Kreindler, E., and Guelman, M., "Ballistic Missile Interception with Bearings-Only Measurements." *Proc. of the* 36^{th} *Israel Conference on Aeronautics and Astronautics*, 1996, pp.203-210.
8. Henzinger T.A., and Sastry, S., *Hybrid systems:Computation and Control*. LNCS 1386, Springer-Verlag, 1998.
9. Moulin, M., Bendersky, E., and Gluhovsky, L., "Application of Formal Verification Methods to the Analysis of Bearings-only Ballistic Missile Interception Algorithms." *Proc. of the* 43^{rd} *Israel Annual Conference on Aeronautics and Astronautics*, 2003.
10. Beer, I., Ben-David, S., Eisner, C., Geist, D., Gluhovsky, L., Heyman, T., Landver, A., Paanah, P., Rodeh, Y., Ronin, G., and Wolfsthal. Y., "RuleBase: Model Checking at IBM," *Proc. of* 9^{th} *International Conference on Computer Aided Verification (CAV)*, LNCS 1254. Springer-Verlag, 1997, pp.279-290.
11. RuleBase - Formal Verification (FV) Tool, developed by the IBM Haifa Research Laboratory: Homepage "http://www.haifa.il.ibm.com/projects/verification/".
12. Clarke, E.M., Biere, A., Raimi, R., and Zhu Y, "Bounded Model Checking Using Satisfiability Solving". *Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99)*, LNCS 1579 , Springer-Verlag, 1999, pp.193-207.
13. Ghose. D., "True Proportional Navigation with Maneuvering Target," *IEEE Trans. on Aerospace and Electronic Systems*, Vol.30, No.1, 1990, pp.713-721.
14. Guelman M., "The Closed-Form Solution of True Proportional Navigation," *IEEE Transactions of Aerospace and Electronic Systems*, Vol.12, No.4, 1976, pp.472-482.
15. Ha, I.-J., Hur, J.-S., and Song, T.-L., "Performance Analysis of PNG Laws for Randomly Maneuvering Targets," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 26 No.5, 1994, pp.229-237.
16. Palnitkar. S., *Verilog HDL*. Prentice Hall PTR, 1996.
17. Moskewics, M., Madigam, C., Zhao, Y., Zhang, L., and Malik, M., "Chaff: Engineering an Efficient SAT Solver" In Proc. of Design Automation Conference (DAC), July 2001, pp.530-535.