

The Evasive Monkey: An Environment For Evaluating Adaptive Learning Algorithms

Mike Roberts * Mark F. DeHerrera † Armando A. Rodriguez ‡

Department of Electrical Engineering
Center for System Science and Engineering
Arizona State University
Tempe, AZ 85287-7606

October 1, 1996

Keywords: Graphical interface, control, adaptive algorithm, performance analysis.

Abstract

This paper describes a Windows/C++-based PC environment for facilitating the development of complex “learning” algorithms. The environment, based on a mobile monkey evading projectile fire, consists of four modules: (i) a program user-interface module, (ii) a simulation module, (iii) a graphics/animation module, and (iv) an education module. The program-user interface module allows the user to interact with the program. More specifically, this module permits the user to select parametric equations describing the 2-dimensional monkey motion. It also permits a user to select cannon dynamics, projectile dynamics, a radar model, motion learning or estimation algorithms, targeting algorithms which compute a firing solution, and algorithm parameters. The simulation module is responsible for the numerical solution of the equations which govern the monkey’s motion, the cannon and projectile dynamics, the radar model, and the algorithms used for motion estimation and targeting. The graphics/animation module updates plots and animation on the screen using data generated by the simulation module. The education module will provide the user with interactive lessons designed to

teach fundamental concepts pertaining to adaptive algorithms, systems, and controls. Designed to communicate with MATLAB, it is shown how this environment may be used to analyze many scenarios by observing system dynamics both graphically and through animation, making the environment a very useful tool for building physical intuition, and teaching adaptive algorithm design concepts.

1 Introduction

Recent technological advances in computational hardware and software have fueled research and development efforts in the area of adaptive systems. Although many theoretical contributions have been seen in this area within the past decade, few such systems are seen in our daily lives. One might point to the cruise control in our cars as an example of an adaptive system. In this system, speed measurements are fed back to a control system. The control system adjusts the throttle automatically in a manner that attempts to maintain a user-specified constant speed. The feedback system adapts to inclines and downslopes by comparing the desired or commanded speed with the actual speed and using the resulting error to compute appropriate increases or decreases in the throttle.

Perhaps a better example of an adaptive system is found in the military arena. Such an example is the missile guidance and control system found within a surface-to-air missile (SAM). In such a system, target information is gathered and used to “learn” or anticipate the motion of the target. A feedback algorithm processes this information and computes a missile heading error. On the basis of this error, vertical and lateral acceleration commands (i.e. desired accelerations) are computed and issued to the missile

*Currently completing a Bachelor of Science in EE.

†Currently completing a Bachelor of Science in EE.

‡This research has been supported, in part, by the National Science Foundation (NSF) through the Coalition to Increase Minority Degrees (CIMD), the ASU Center for Innovation in Engineering Education (CIEE), the Boeing A.D. Welliver Faculty Fellowship Program, and the Intel Corporation. For additional information please contact aar@asu.edu.

autopilot. The missile autopilot compares the desired accelerations with the actual acceleration and uses the error value to compute fin deflection commands. The fin deflection commands are then issued to the servos driving the missile’s control surfaces. The missile autopilot, we see, works very much like the cruise control system found in our cars. The above examples suggest that the components of an adaptive system are as follows:

1. Feedback of measured signals,
2. Calculation of decision variables, called *controls*, on the basis of the feedback information and some performance objective,
3. A changing or uncertain environment (e.g. changing objective, disturbances, etc.).

Given the above, the development of the Evasive Monkey environment has been driven by the need for an environment to facilitate the analysis, design, and performance evaluation of adaptive algorithms. Toward this goal, and inspired by missile guidance and control systems, the authors propose the Evasive Monkey environment. Within this environment the main objective is to hit a non-stationary target (i.e. evasive monkey) with a projectile. The critical issue then becomes learning, or predicting, the motion of the monkey and updating appropriate projectile launch parameters. This environment, accompanied by a user-interface with pull-down menus, dialogue boxes, and real-time graphics, provides an excellent testbed for analyzing, designing, and evaluating the performance of the latest adaptive algorithms. Visualizing projectile-monkey tactics on the screen as they evolve, rather than just plotted simulation data, can provide significant insight to engineers and scientists as well as students. Other such environments by the authors include [2], [3], [9], [12], [14],[15], [16].

2 Mathematical Models

In this section, the essential components of the evasive monkey environment are described. The environment consists of (1) an evasive monkey, (2) a radar, (3) a projectile launcher, (4) projectiles, and (5) learning algorithms. The flow of information between these objects may be visualized as shown in Figure 1. The mathematical model used to describe each is now discussed.

Monkey Kinematics. The monkey’s motion is assumed to be periodic and two dimensional. A user specifies the monkey’s motion by specifying parametric equations for the horizontal and vertical components of motion, $x_m(t)$ and $y_m(t)$. More specifically, a user specifies fundamental frequencies ω_x

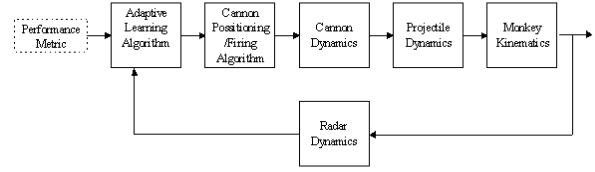


Figure 1: Information Flow Within the Evasive Monkey Environment

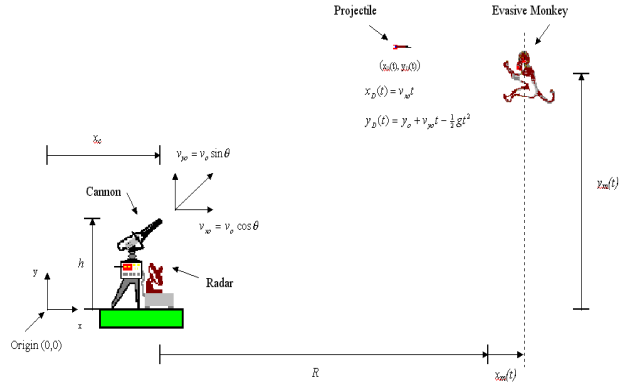


Figure 2: Monkey-Projectile-Launcher-Radar Environment

and ω_y for the horizontal and vertical motions, respectively. The user also specifies Fourier coefficients $a_{n_x}, b_{n_x}, a_{n_y}, b_{n_y}$, where

$$x_m(t) = a_{o_x} + \sum_{n=1}^{M_x} a_{n_x} \cos n\omega_x t + b_{n_x} \sin n\omega_x t \quad (1)$$

$$y_m(t) = a_{o_y} + \sum_{n=1}^{M_y} a_{n_y} \cos n\omega_y t + b_{n_y} \sin n\omega_y t \quad (2)$$

The Fourier orders M_x and M_y are determined by the number of Fourier coefficients specified by the user.

Radar Dynamics. There are two user-selectable modes for the radar. In the first mode, the exact coordinate of the monkey ($x_m(t), y_m(t)$) is known to the radar at discrete time samples $t = nT_s$ where T_s is the *sampling interval*. These samples are available to the adaptive learning algorithms which “estimate” the monkey’s motion. In the second mode, the radar mimics a true radar by incorporating a potential for inaccuracy, controlled by a probability distribution function. In the future, a user-defined “monkey stealth coefficient” will be used to implement a signal scattering model.

Projectile Launcher Dynamics. The projectile launcher is located at coordinate $(0,0)$. There are two user-selectable modes for the projectile launcher. The first mode assumes instantaneous muzzle elevation and exit velocity selection; i.e. negligible projectile launcher dynamics. The second mode uses a set of ordinary differential equations to govern the muzzle actuator-elevation dynamics. The critical cannon variables in our study, however, are muzzle elevation θ and muzzle velocity v_{p_o} .

Projectile (Dart) Dynamics. There are three user-selectable modes for the projectile dynamics: no aerodynamic resistance, linear resistance, and quadratic resistance. When no air resistance is selected, the dart kinematics are given as follows:

$$x_p(t) = v_{p_o} \cos \theta t \quad (3)$$

$$y_p(t) = v_{p_o} \sin \theta t - \frac{1}{2}gt^2 \quad (4)$$

where $g = 9.8 \frac{m}{sec^2}$ denotes the acceleration due to gravity.

Adaptive Algorithms: Motion Estimator. There are a number of methods currently available to “learn” or “estimate” the monkey’s motion. Each method produces a model which can be used to predict or estimate the monkey’s motion. Two methods are now discussed.

Fourier Coefficient Estimation Method. One method attempts to estimate the Fourier coefficients. This is done by gathering data, approximating ω_{0_x} and ω_{0_y} and then solving for the Fourier coefficients using Gaussian Elimination. Another method simply takes the gathered data and attempts to solve a non-linear set of equations for the fundamental frequencies and Fourier coefficients. These methods are partly addressed in prior papers [3].

Taylor Series Estimation Method. In this paper, the focus is on a Taylor Series approximation technique to estimate the monkey’s future position. The estimates for the monkey’s horizontal and vertical coordinates are denoted by the symbols \hat{x}_m and \hat{y}_m , respectively, and are given by:

$$\hat{x}_m(t) = \sum_{n=0}^{N_x} \frac{d^n \hat{x}}{dt^n} \frac{t^n}{n!} \quad (5)$$

$$\hat{y}_m(t) = \sum_{n=0}^{N_y} \frac{d^n \hat{y}}{dt^n} \frac{t^n}{n!} \quad (6)$$

where N_x and N_y are user-defined integers denoting the order of the Taylor Series approximations. The coefficients $\hat{x}^{(n)}$ and $\hat{y}^{(n)}$ are obtained on the basis of gathered data and simple forward difference derivative approximations. As such, this method is expected to “amplify noise”. Nevertheless, this method is a good starting point to illustrate basic concepts. The number of data samples used to estimate the needed derivatives is $N_x + N_y + 2$.

Targeting, Positioning, and Firing. The monkey motion models generated by the motion estimator are fed to a “targeting algorithm” which computes a “firing solution” and sets the angle of the cannon to hit the monkey. To do this, the following nonlinear equations

$$f_1(t, \theta) = x_p(t, \theta) - \hat{x}_m(t) = 0 \quad (7)$$

$$f_2(t, \theta) = y_p(t, \theta) - \hat{y}_m(t) = 0 \quad (8)$$

are solved, for t and θ , using Newton’s method [8, pg.201–202].

3 Description of Environment

The Evasive Monkey environment is written in Windows/C++ [1], [5],[7],[11],[17]. It consists of four modules: (1) a program user interface (PUI) module, (2) a simulation module, (3) a graphics/animation module, and (4) an education module. Each module is now described.

Program User Interface (PUI) Module. The PUI provides an interface between the user and the Evasive Monkey program. Written in Windows/C++, the interface provides pull-down menus which permit the user to select models, algorithms, and parameters. The user, for example, can (i) specify the parametric equations describing the 2-dimensional monkey motion, (ii) select a radar model, (iii) select an adaptive algorithm to estimate the monkey’s motion, (iv) select a targeting algorithm to determine cannon firing parameters, (v) select a cannon dynamics, etc. (vi) select projectile aerodynamic properties, etc.

The user may also choose to control the firing of the cannon manually, much like an arcade game. In *interactive* mode, the cannon is controlled using the mouse. Left-clicking the mouse anywhere on the screen will cause the cannon to align itself with the mouse pointer. Right-clicking the mouse causes the cannon to fire, with the velocity being proportional to the pointer’s horizontal distance from the cannon.

Other menu options for data storage and plotting also exist. Data storage routines automatically

format saved simulation data for use within other environment modules as well as external programs (e.g. MathWork's MATLAB and Microsoft's Excel). User-selected variables saved from an earlier simulation may be plotted against current simulation data. Multimedia lessons, which use live audio and video will also be accessible through the PUI.

Simulation Module. The simulation module is responsible for solving all of the equations associated with the monkey motion, radar, motion estimation algorithm, targeting and firing algorithm, cannon dynamics, and the projectile dynamics.

Graphics/Animation Module. The main purpose of the graphics module is to use data provided by the simulation module to update graphics (i.e. plots) and animations on the screen. Data and plots are displayed within *child windows* [6], [10]. Animation is created using high quality bitmaps. The bitmaps were generated with Corel's CoreDRAW and CorelPHOTO-PAINT¹.

Education Module. The education module will contain routines which implement interactive multimedia lessons. The lessons use text, audio, video, and animation to convey ideas. Users answer questions and control simulation parameters via interactive menus. Correct answers are supported with a multimedia explanation - including "supportive audio." Incorrect answers are followed up by hints, partial explanations, and additional chances.

4 Educational Utility

In this section, the utility of the Monkey Environment as an educational tool is demonstrated. We specifically examine the impact of monkey motion parameters, such as frequency of oscillation and distance to projectile launcher, on *miss distance*.

Monkey Motion. Throughout this section, it will be assumed that the monkey horizontal and vertical coordinates are given by the following trigonometric expressions:

$$x_m(t) = R - 125 \cos \omega t + 75 \sin \omega t \quad (9)$$

$$y_m(t) = 100 + 125 \cos \omega t - 100 \sin \omega t \quad (10)$$

where $\omega = \omega_x = \omega_y$ and R will be the focus of our study.

¹CorelDraw and CorelPHOTO-PAINT are trademarks of Corel Corp. CorelDraw is a general purpose drawing/graphics manipulation utility. CorelPHOTO-PAINT is a general purpose image processing utility.

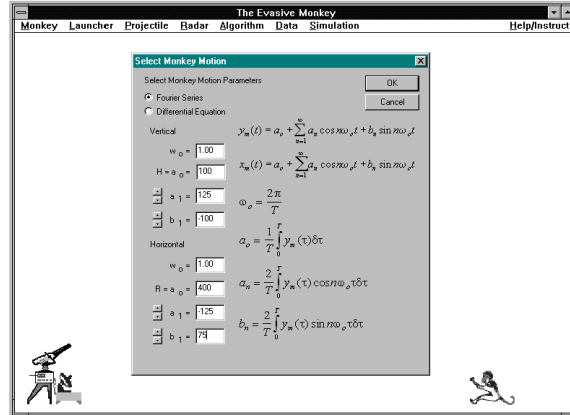


Figure 3: Menu to Define Monkey Motion

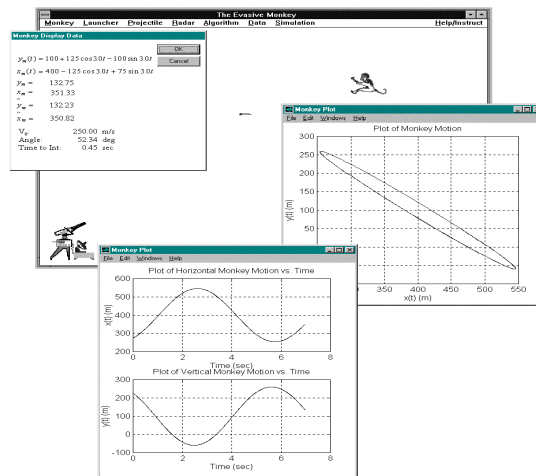


Figure 4: Monkey Motion

Figure 3 shows the menu used for defining the parametric equations describing the 2-dimensional monkey motion. The user selects the form of the equations and specifies the appropriate motion parameters. In this case, a Fourier series has been selected and the user has specified the frequencies, starting displacements, and Fourier coefficients for both the horizontal and vertical motion. The user may specify coefficients to desired arbitrary Fourier orders M_x and M_y .

Figure 4 is a screen capture of the Evasive Monkey environment during one of our simulations. A dialog box containing user-selectable display data is shown in the upper-left corner. As seen in the figure, the actual position of the monkey, (x_m, y_m) can be easily compared to the calculated position (\hat{x}_m, \hat{y}_m) . The figure also shows the horizontal and vertical positions of the monkey versus time and a plot of the elliptical path that the monkey traces on the screen as it moves.

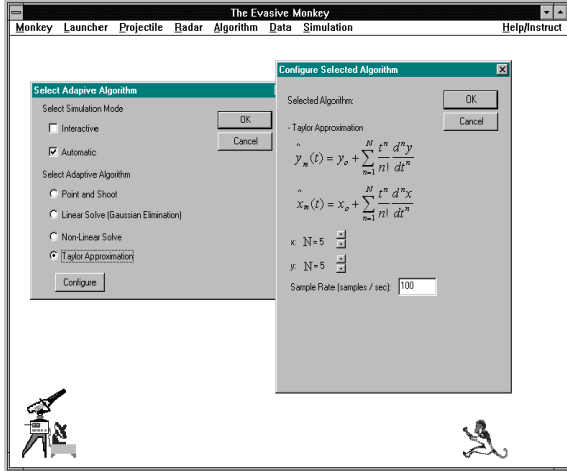


Figure 5: Adaptive Algorithm Set-up Menus

Adaptive Learning Algorithm: Taylor Series Estimation. Figure 5 shows the menus used to set up the adaptive learning algorithm that will be used to estimate the motion of the monkey. In the *Select Adaptive Algorithm* menu, the user first specifies whether the monkey will be targeted using the computer or by using the mouse interactively like an arcade game. If this is set to *automatic mode* then the user may select between a number of adaptive algorithms. In this case, the user has chosen to use a Taylor Series motion estimation algorithm. The user then configures the specific algorithm using the *Configure Selected Algorithm* menu. In this case, the user has opted to take both the horizontal and vertical Taylor approximations to the 5th order using a radar sampling rate of $f_s = \frac{1}{T_s} = 100Hz$.

Projectile Dynamics. In this study, the projectile aerodynamics were neglected; i.e. zero drag. The initial projectile speed was set to $v_{p_o} = 250 \frac{m}{sec}$.

Targeting Algorithm. Newton's method was used to solve Equations 7-8 for an approximate intercept time t_f and a launch angle θ . The initial conditions used were $t_f = \frac{x_m(0)}{v_{p_o} \cos \theta}$ and $\theta = 20$ degrees. This initial choice for t_f and θ resulted in few Newton iterations. It should be noted that initial conditions based on higher order monkey derivative information reduced even further, the number of Newton iterations.

Impact of Frequency on Miss Distance. For a given projectile firing, the *miss distance* is defined by the relationship

$$D = \sqrt{|x_p(t_f) - x_m(t_f)|^2 + |y_p(t_f) - y_m(t_f)|^2} \quad (11)$$

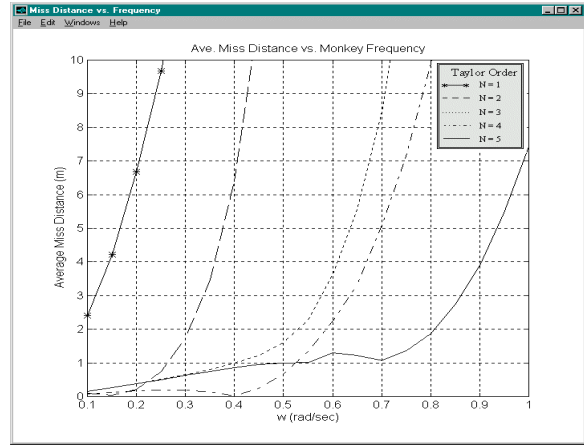


Figure 6: Impact of Monkey's Fundamental Frequency on Average Miss Distance for Various Taylor Orders

where t_f is determined by the Newton-based targeting algorithm.

Figure 6 shows simulation data of the average miss distance versus monkey frequency ω for $R = 400 m$ and five Taylor series approximations. For each value of frequency and Taylor order, the simulation was started at 10 different initial times, from $t_0 = 0$ to $t_0 = 1$ second in steps of 0.1 seconds. The miss distances for these runs were then averaged to find an average miss distance for each frequency-Taylor order pair. The figure shows that for a given Taylor series order, the miss distance increases with increasing monkey frequency ω . This is intuitively expected. The figure shows that the higher order Taylor estimates performed well over a larger range of ω than the lower ordered estimates.

Impact of Initial Position on Miss Distance.

Figure 7 shows a plot of average miss distance versus the initial monkey distance from the projectile launcher. This plot was generated by varying the parameter R with the monkey frequency fixed at $\omega = 0.5$ rad/sec. The plot shows that the average miss distance - computed as discussed above - increases as the initial distance is increased. This too is intuitive. Once again, it is shown that the higher Taylor approximations perform better than the lower ones.

5 Summary and Directions

This paper has described an environment for simulating and graphically visualizing learning algorithms and missile-target intercepts. The program may be used as a tool for demonstrating how parameter variations affect algorithm effectiveness and efficiency.

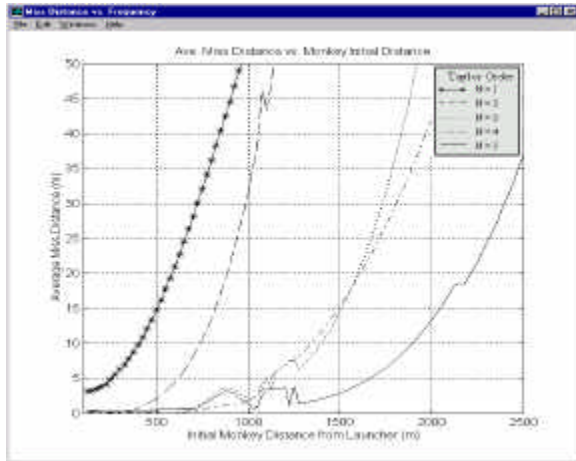


Figure 7: Impact of Initial Monkey Distance on Average Miss Distance for Various Taylor Orders

The environment is designed so that students can experiment and develop insight into adaptive algorithms and missile-target intercepts in general. The simplicity of the environment lends itself well to instruction on topics related to both controls and computer science. Currently, help and instruct data bases are being developed to help users master the program features and provide them with instruction on relevant concepts. Future work will include the development of interactive computer-aided-lessons (ICALS) to teach specific concepts [13], [14].

References

- [1] L. Adams, *Supercharged C++ Graphics*, McGraw Hill, 1992, pp. 207-223.
- [2] K.J. Elliott and A.A. Rodriguez, "A Graphical Dual Robot Simulation for the PC," *Proceedings of the 1995 International Conference on Simulation in Engineering Education*, Las Vegas, NV, January 15-18, 1995, pp. 77-81.
- [3] M.F. DeHerrera and A.A. Rodriguez, "Trying to 'Shoot' an Evasive Monkey: A Tool for Designing and Evaluating Adaptive Learning Algorithms," *Proceedings of the 1996 International Conference on Simulation in Engineering Education*, San Diego, CA, January 14-17, 1996, pp. 31-36.
- [4] L. Geppert, "The New Contenders," *IEEE SPECTRUM: Fast and Powerful*, December 1993.
- [5] L. Heing, *Advanced Graphics Programming Using C/C++*, John Wiley & Sons Inc., 1993, Chapter 1.
- [6] L. Heiny, *Windows Graphics Programming with Borland C++*, John Wiley & Sons Inc., New York, NY, 1992.
- [7] S. Holzner, *Borland C++ Windows Programming*, Brady Publishing, Indianapolis, IN, 3rd Edition, 1994.
- [8] D.G. Luenberger, *Linear and Nonlinear Programming*, Second Edition, Addison-Wesley Publishing Company, 1984.
- [9] R.P. Metzger, K.J. Elliott, and A.A. Rodriguez, "Modelling, Analysis, and Graphical Visualization of a Dual Robot Arm System: A PC Based Environment," *Proceedings of the 1996 International Conference on Simulation in Engineering Education*, San Diego, CA, January 14-17, 1996, pp. 175-180.
- [10] S. Oualline, *Windows Programming with Borland C++*, M&T Books, New York, NY, 1993.
- [11] S. Potts and T.S. Monk, *Borland C++ 4 By Example*, Que Corporation, Indianapolis, IN, 1994.
- [12] A.A. Rodriguez and R. Aguilar, "Graphical Visualization of Missile-Target Air-to-Air Engagements: An Educational Tool for Designing and Evaluating Missile Guidance and Control Systems," *Journal of Computer Applications in Engineering Education*, Vol. 3, No. 1, 1995, pp. 5-20.
- [13] A.A. Rodriguez, M.F. DeHerrera, and R.P. Metzger, "An Interactive MatLab-Based Tool for Teaching Classical Systems and Controls," to appear in the Proceedings of the 1996 Conference on *Frontiers In Education*, Salt Lake City, Utah, Nov 6-9, 1996.
- [14] A.A. Rodriguez and M.F. DeHerrera, "Modeling, Simulation, and Graphical Visualization of a Twin Lift Helicopter System Under Automatic Control: An Educational Tool," to appear in the Proceedings of the 1996 Conference on *Frontiers In Education*, Salt Lake City, Utah, Nov 6-9, 1996.
- [15] M. Sonne, A.A. Rodriguez, "A PC-based Graphics System for the Evaluation of Missile Guidance and Control Laws", Proceedings of the American Control Conference, Baltimore, MD. June 29-July 1, 1994, pp. 2726-2730.
- [16] M. Sonne, A.A. Rodriguez, "PC's in the Design and Evaluation of Guidance and Control Systems for Missiles," Proceedings of the 1994 ICSEE, Tempe, AZ, January 21-23, 1994, pp. 329-334.
- [17] P. Yao, *Borland C++ 4.0 Programming for Windows*, Random House Inc., New York, NY, 1994.