2006-444: RADAR PERFORMANCE ANALYSIS SYSTEM: A SOFTWARE SIMULATION PACKAGE IN MATLAB

Roger Lee, Coherent Systems International

Rafic Bachnak, Texas A&M University-Corpus Christi

Radar Performance Analysis System: A Software Simulation Package in MATLAB

Ru-Ying Roger Lee
Titan, an L3-Communications Company
22290 Exploration Drive,
Lexington Park, MD 20653

Rafic Bachnak
Department of Computing and Math Sciences
Texas A&M-Corpus Christi
Corpus Christi, TX 78412

Abstract

After World War II radar has been increasingly used as an integral part of complex applications such as air-traffic control, reconnaissance and surveillance, navigation and weather, and missile targeting and defense. Analysis of radars systems for such applications requires representing the radar operation and performance in the context of the overall system and the external environment. The radar performance must be evaluated, and the impact of radar operation on system performance must be quantified. The major objective of this paper is to present the design/development and implementation of the automated end-to-end functional radar simulation software package. Using the database driven approach in the software development, this paper also discusses how the software package can be changed and used to solve problems in any other engineering and technology curricula. Finally, this paper describes how this software package developed in MATLAB can be converted into a C/C++ source code and then compiled the source code into a stand-alone executable system.

1. Introduction

In the signal-processing environment, computer simulation is used extensively in developing complex systems, including radar performance systems [1]. To analyze the performance of these radars at a system level, it is practical to rely on basic radar principles to model their performance, using the top-level parameters to characterize the radar design. Computer simulation is the most practical method of supporting system analysis due to the complex system interactions between radars and parameters relating to the environment [2].

Radar simulations, like most real time signal processing tasks, require significant quantities of data. This data usually represents a collection of different signals. The organization of the data implies some constraints on the way the data is processed by different functions [3]. Radar system technology has evolved tremendously during the last decade. The existing PC simulation software, although lower in cost, often falls short of meeting the high-powered simulation requirements needed to test complex radar

equations [4]. In this paper, we discuss the design and development of the Radar Performance Analysis System (RPAS). RPAS analyzes, predicts, estimates, and evaluates the performance of a radar system in a simulation environment. Upon providing the data such as the radar hardware parameters, antenna model, target geometry, terrain model, weather condition, jamming inference, and the desired applications, RPAS performs radar system design trade studies, sensitivity analysis, and performance predictions. RPAS is a stand-alone system running on a PC with the windows operating system and with a removable hard drive to accommodate the classified input data and output results.

MATLAB is selected as the programming language for RPAS to reduce overhead and speed up the development. This high level language provides abundant mathematical and engineering in-line functions and comprehensive graphic routines for the development of scientific and engineering system [5]. MATLAB also offers graphical user interface (GUI) tools that allow the designer/developer to use MATLAB in an application development environment. This combination of programming features and GUI tools made MATLAB an extremely attractive choice for the development of RPAS. The Radar function library database was designed/developed for six different categories: Radar Equation, Radar Detection, Radar Search, Radar Measurement, Environment and Mitigation Techniques, and Radar Countermeasures and Counter-Countermeasures. This paper documents the RPAS system design and implementation. The paper also shows how to convert the RPAS MATLAB program into C/C++ program and compile it to an executable program so that it will run faster and can be executed without the presence of MATLAB.

2. System Design and Radar Simulation

System Design

RPAS is designed as an open architecture system which results in a generic performance analysis/evaluation tool. Currently, the RPAS is a static system, it analyzes/evaluates the radar performance at the system level. Figure 1 shows the information flow among various components in RPAS. For easy maintenance and expansion of the simulation program, a modular approach was adopted for the implementation.

Several pushbuttons such as Figure, Result, Clear, Cancel were created by using the GUIDE (Graphical User Interface Development Environment) facility of MATLAB [5]. In the block diagram, shaded rectangles are menu/submenu, Edit Box, and Check Box. Shaded ellipses are the switches, and the underlined names are the software modules. The modules ending with CB are call back functions. For instance the ResultCB is invoked when the Result button is pressed. The ResultCB function in turn invokes the GenResult function. The GenResult retrieves a specific function from the Radar Library. The retrieved function, derived from menu/submenu, uses parameter inputs, and selected data units stored in Global Data by ParInputCB and SelectBoxCB to compute results. Solid lines represent control flow, dotted lines represent data flow, and dot-dash lines indicate how the GUI is enabled.

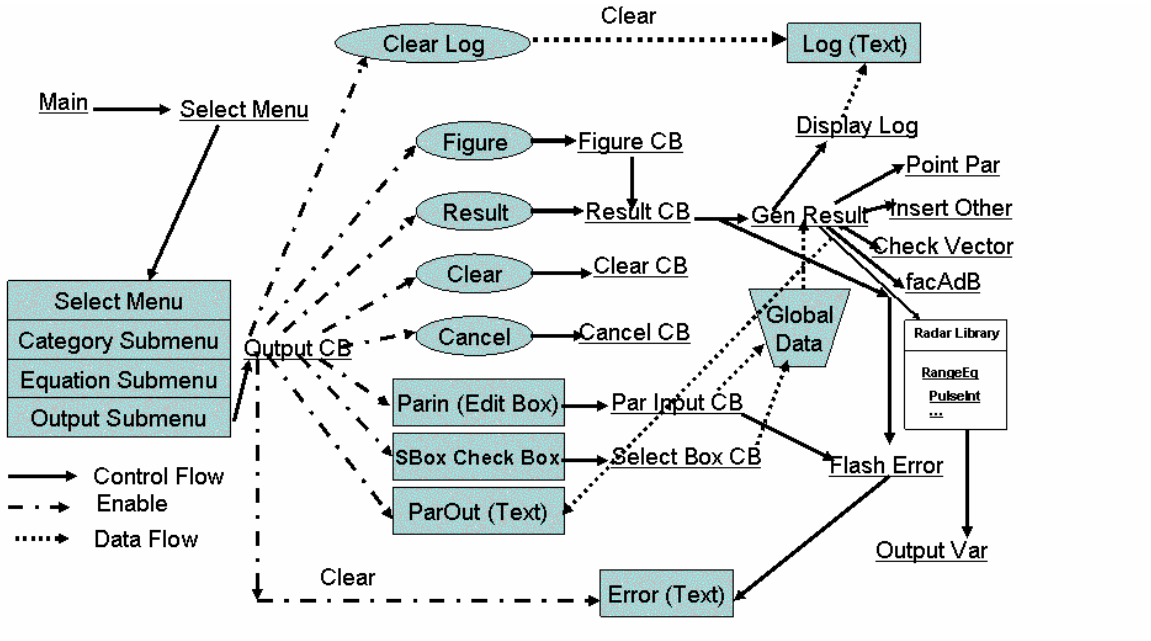

Figure 1. RPAS/Function Block Diagram

Radar Simulation

This section describes the features of RPAS. As an example, the basic form of the radar range equation gives the ratio of signal power from the target to the background noise power at the radar receiver, which includes both noise received from the external environment and noise added in the radar [6]. Consider the classical Radar Range Equation,

$$S/N = (P_p \tau G_t \sigma A_r)/ [(4\pi)^2 R^4 k T_s L] \qquad (1)$$

where

S/N = radar signal-to-noise ratio (factor)
P_p = radar transmitted peak RF power (Watts)
τ = radar pulse duration (seconds)
G_t = radar transmit antenna gain (factor)
σ = target radar cross section (square meters)
A_r = radar receive antenna effective aperture area (square meters)
R = range from the radar to the target (meters)
T_s = radar system noise temperature (Kelvin)
L = radar system losses (factor)
k = Boltzmann's constant (1.38e-23 Joule/Kelvin)

As an example, let us perform the following radar simulation. A radar antenna transmits 1kW of peak power at a frequency of 3 GHz with duration of 1 msec. The receiver has a gain of 32dB, with the effective aperture area of 14 square feet. The system has a loss factor of 10, and the effective noise temperature is 800K. A target with cross- section of 92.903 square meters is detected by the radar. Find the signal-to-noise ratio at a range of 20,40, and 100 nautical miles. We will use this example to guide you through the following simulation.

RPAS simulation models a parameterizable radar system, where the geometry of the antenna, and other radar parameters can be easily changed. RPAS provides the following special features and capabilities:

- Database design to achieve generic system
- Selection of output parameter of an equation
- User Input with Unit Selections
- Display of numerical or figure outputs
- Execution records/history saved in the LOG area

Database Design to Achieve a Generic System

The RPAS system is designed to be application independent. Therefore, to define a radar function we see it as an elementary operation applied to generic elements. The text of pull-down menus for category, equation, output selections, the mathematics expression of equation, input parameters and unit selections are all read in from a master database file. In that, the system is a generic system not only for the radar application but also for general application. Selecting another master file, the system can become a "civil engineering", a "mechanical engineering", or an "electronics engineering" Performance Analysis System. Of course, every bottom-line equation needs to be coded for that specific application. In the current RPAS system, all equations for radar applications are implemented into a package called the "Radar Library". This approach presents a simple and concise way to define a function in a manner equivalent to functional languages using higher-order function [7].

Selection of Output Parameter of an Equation

The RPAS groups all radar equations into 6 categories: Radar Equation, Radar Detection, Radar Search, Radar Measurement, Environment and Mitigation Techniques, and Radar Countermeasures and Counter-Countermeasures. The user selects a particular equation via the "SELECT -> CATEGORY -> EQUATION pull-down menus. Using our example, the above radar range equation is obtained from the selection of "SELECT -> RadarEquation -> RangeEq" as shown in figure 2.

Equation (1) is Radar Range Equation having S/N as output and the other parameters as inputs. When the user has selected this equation with this output as shown in figure 2 the system provides the mathematical expression and prompts the user for input as shown in figure 3. The system also provides the user with the capabilities of using the same Radar Range Equation but selecting another output parameter, such as R, the Range. In this case the system provide the mathematics expression of the Radar Range Equation with R as output and again prompt user for the input parameters. Likewise, other parameters, such as PulseWidth, in the Radar Range Equation can be selected as output. If it is for sure that some parameters, for example $T_{s,}$ will never be an output, we can disable the output selection of this particular parameter as shown in figure 2. It is noted that in figure 2 the allowable parameters for output are S/N, pulse width, and range. Finally, the switch "CANCEL" shown in figure 3 can be used for canceling the previous selection and selecting a new equation and output.

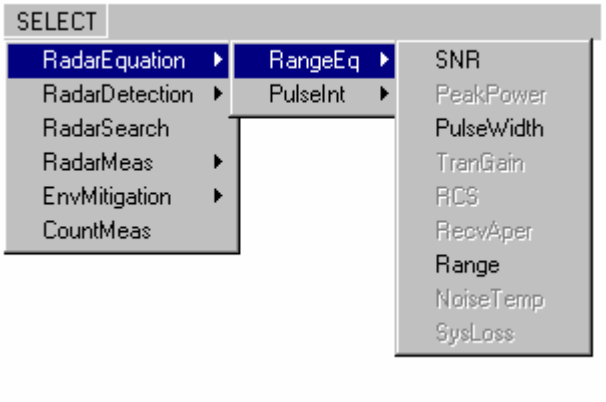

Figure 2 SELECT-> RadarEq ->Range

User Input with Unit Selections

After the system responds to user's selection of equation and output parameter as shown in figure 3, the user should enter data to the prompted input parameters. The valid inputs are: (a) single value x, (b) a sequence of values separated by commas (,) as x,y,z, (c) beginning and ending values with step value 1 as x:z, and (d) beginning, step, and ending values as x:y:z. If the input is other invalid data, the system will flash an error of "illegal input at n-th input" in the "system error box" at the left bottom corner of the screen. Also, shown in figure 3, to be a user friendly system, the system provides the user with the capability of selecting various commonly used units for inputs and output, and once a unit is selected the other units of the same parameter will be automatically unselected. A user can also start with a clean "sheet of input" by clicking the "CLEAR" switch and then entering a new set of data. Using our example, the values and units of input parameters of the radar range equation are selected as shown in Figure 3.

Display of Numerical and Figure Outputs

After the parameter inputs and data unit selections are entered, the user can click the "RESULT" switch for numerical output or the "FIGURE" switch for an output plot. If some of the input data or unit selection is not entered, the system will flash an error "Some input parameters are not entered" in the system error box. The data formats (b), (c), and (d) described in the last paragraph provide a row vector for the input. If more than 2 inputs are vectors, the system will flash an error "More than two vectors in input, STOP processing" and no further processing will occur. If every input is a singleton, then the "RESULT" switch will yield a singleton output. In this case, a click of the "FIGURE" switch will cause the system to flash error "Cannot plot a figure because no input vector". If only one input is a vector then a click of "RESULT" switch will yield a vector result and a click of "FIGURE" switch will plot a 2-dimension figure of input vector vs. output vector. If two inputs are vectors, then these two inputs will form a mesh and a click of "RESULT" switch will yield a 2-dimensional output and a click of "FIGURE" switch will yield a 3-D plot with the two inputs vector as x- and y-axes and output as a z-surface. When a 3-D plot is displayed, the user can further leverage the MATLAB figure utility to manipulate the figure such as 3-D rotation. Figure 3 shows the result of our example after the "RESULT" switch is clicked. Figure 4 shows the 3D figure of our example after the "FIGURE" switch is clicked and after a rotation is applied to the figure.

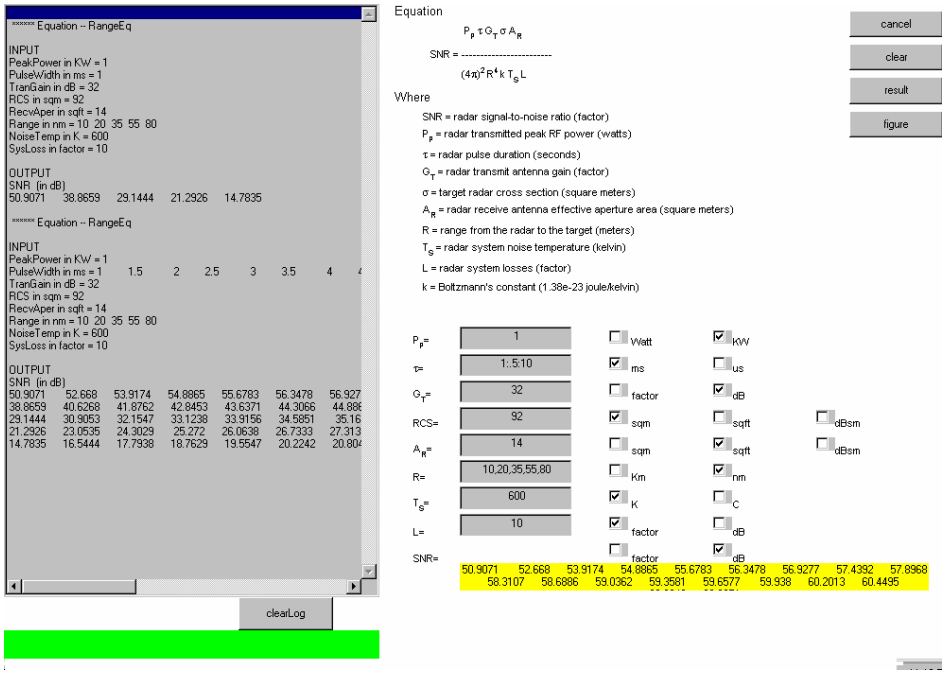

Figure 3. Execution of the Range Equation

Execution Records/history in the LOG Area

After an equation (with user's input) is executed (either the "RESULT" or the "FIGURE" switch is clicked), the equation, the input and the output will also be displayed in the LOG area in the left side of the screen, as shown in figure 3, for review or for printing out a hardcopy. The LOG text will be concatenated from one run to the next run. A "ClearLog" switch is also available to clear the text in the LOG area.

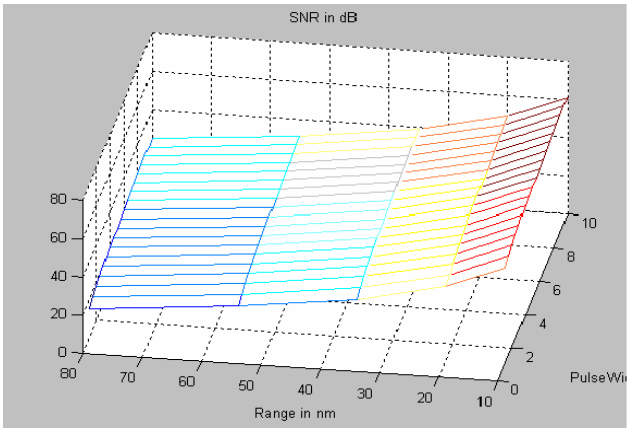

Figure 4 3D plot of Radar Range Equation

3. Building a Stand-Alone RPAS Executable System

This section describes how the RPAS MATLAB program in m-files can be converted into RPAS C/C++ program and then compiled into a stand-alone RPAS executable system. The advantages of building a PRAS executable system are: (1) it is a stand-alone system that can be executed in and PC without the presence of the MATLAB, and (2) it

runs faster. Depending on the complexity of functions and computational intensity, the compiled version can run up to 5 times faster [8]. A set of tools that automatically converts MATLAB files into C/C++ source code, offers programmers the ability to create stand-alone applications using a very extensive library of predefined functions [8]. The RPAS MATLAB program was successfully converted to a stand-alone executable file. The tools required for generating stand-alone applications are: MATLAB, the MATLAB Compiler, the MATLAB C/C++ Math Library, and a C/C++ compiler. The MATLAB C/C++ Graphics Library is also required in order to create stand-alone applications that make use of Handle Graphics® functions.

A MathWorks utility, mbuild, provides an easy way to specify an options file that can be used to set the compiler and linker settings, change compilers or compiler settings, switch between C and C++ development, and build an application. The Compiler (mcc) automatically invokes mbuild under certain conditions. In particular, mcc -m or mcc –p invokes mbuild to perform compilation and linking. To prevent mcc from invoking mbuild automatically, the -c option can be used; for example, mcc -mc filename. On systems where there is exactly one C or C++ compiler, the mbuild utility automatically configures itself for the appropriate compiler. On systems where there is more than one C or C++ compiler, the mbuild utility lets the user select which compiler the user want to use. Once a C or C++ compiler is selected, that compiler becomes the default compiler. The user may later specify another compiler by following the same procedure.

The Compiler, when invoked with the -m macro option, translates input MATLAB files into C source code that is usable in any of the supported executable types. The Compiler also produces the required wrapper file suitable for a stand-alone application. Then, an ANSI C compiler compiles these C source code files and the resulting object files are linked against the C/C++ Math Library. Similarly, the Compiler (mcc), when invoked with the -p macro option, translates input M-files into C++ source code that is usable in any of the executable types except MEX. The source code for a stand-alone C or C++ application consists either entirely of M-files or some combination of M-files, MEX-files, and C or C++ source code files. After compiling the C or C++ source code, the resulting object file is linked with the object libraries.

4. Summary

We have described the design and implementation of RPAS. With examples presented in this paper we have shown how RPAS could be applied to model and simulate problems in different applications. Due to the predominance of MATLAB as a major tool of engineering and scientific research it was naturally advantageous to develop RPAS entirely within MATLAB. Furthermore, MATLAB offers graphical user interface tools, which provides a high level of flexibility in design and implementation. RPAS can be fully integrated into any course with minimal effort and can provide a robust method for solving engineering problems with a computer. In the future, we will expand RPAS into a dynamic system to analyze/evaluate the radar performance at operational level in an environment where the radar equation, target geometry, terrain model, weather condition, and jamming can change in time. To achieve that, the target and environment will be modeled and then the radar performance will be assessed by simulation to simulate the real operation with the desired parameters.

References

1. G. Desodt, D. Muller, D. Puzenat, "OSCAR, a simulation environment dedicated to the design and performance assessment of RADAR system, "International Conference on Radar, Paris, 1994.
2. Barton, D. K., Modern Radar System Analysis, Norwood, MA; Artech, 1988.
3. F. H. M Fransen, F. Balasa, M. F. X. B. Van Swaaij, F.V. M. Cathoor, and H. J. De Man "Modeling Multidimensional Data and Control Flow", IEEE Trans on VLSI Systems, 1993.
4. Radar Systems Analysis and Design using MATLAB, Bassem R. Mahafza, 2000.
5. Mastering MATLAB 6, A Comprehensive Tutorial and Reference, Duane Hanselman, and Bruce Littlefield, 2001.
6. Radar System Performance Modeling, G. Richard Curry, 2001.
7. Modeling Radar Systems Using Hierarchical Dataflow, Karim P. Khair, and Edward A. Lee, Proceedings of ICASSP, May 1995.
8. R. Bachnak and R. Lee, " The MATLAB Compiler Suite: M-Files to C/C++ Executable Programs," Proceedings of the 2002 ASEE Annual Conference and Exposition, June 16-19, 2002 Montreal, CA.

RU-YING ROGER LEE

Dr. Ru-Ying Roger Lee is a retired senior scientist from the Radar and Antenna Division, Avionics Department of Naval Air Warfare Center in Patuxent River, Maryland. He currently works as a Senior Scientist Engineer for Titan, an L3-Communications Company. He received his BS degree in Mathematics from National Taiwan University, MS degree in EE from the Pennsylvania State University, and Ph.D. in Mathematics from University of Pennsylvania. His area of research is in simulation, signal processing and image processing; specifically for the applications of the Radar Systems in general and Synthetic Aperture Radar in particular.

RAFIC BACHNAK

Rafic (Ray) Bachnak is Professor and Coordinator of Engineering Technology at Texas A&M University-Corpus Christi (A&M-CC). He received his B.S., M.S., and Ph.D. degrees in Electrical and Computer Engineering from Ohio University in 1983, 1984, and 1989, respectively. Dr. Bachnak was previously on the faculty of Franklin University and Northwestern State University.