Modeling and Simulation of Inertial Navigation Systems with Automated Interface Checking and Post-Processing Support

Yiannis Papelis and Menion Croll
Virginia Modeling, Analysis and Simulation Center
Old Dominion University
[ypapelis, mcroll]@odu.edu

James Leathrum, Roland Mielke and Jesse Caldwell
Modeling, Simulation and Visualization Engineering
Old Dominion University
[jleathru, rmielke, jcald013]@odu.edu

Robert Greer and William Labelle
Spawar System Center Atlantic
[robert.a.greer, william.labelle]@navy.mil

ABSTRACT

Even though the architecture of Inertial Navigation Systems (INS) remains relatively constant, a designer is faced with numerous choices when it comes to identifying the appropriate sensors, selecting among similar sensors and configuring the appropriate filtering. Software tools that can simulate an INS are frequently used so support a specific INS model but there are relatively few general purpose tools that support arbitrary INS topologies. In this paper we describe a framework that supports modeling and simulation of INS. The framework is not tied to a specific INS topology and allows modeling at an arbitrary level of detail. Some novel features of the framework include a data dictionary that contains the system interface specification and which allows automated checking of the interfaces between modules, a scenario generator that can automatically generate Monte Carlo simulations based on specific input parameters, and a data-aware post-processor that can compactly display simulation results. The paper describes the framework and provides examples of its usage.

ABOUT THE AUTHORS

Yiannis Papelis is a Research Professor at the Virginia Modeling, Analysis & Simulation Center at ODU. His research interests include autonomous unmanned systems, NextGen, semi-autonomous behavior modeling, and immersive virtual reality. Dr. Papelis received a BS degree from Southern Illinois University, an MS degree from Purdue, and a Ph.D. degree from University of Iowa, all in electrical & computer engineering.

James Leathrum, Jr. is an Associate Professor in the Department of Modeling, Simulation and Visualization Engineering at ODU. His research interests include hardware and software constructs for simulation, applications of discrete event simulation, computer architectures, and distributed simulation. Dr. Leathrum received the BS degree in computer engineering from Lehigh University, and MS and PhD degrees in electrical engineering from Duke University.

Roland Mielke is a University Professor in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. His research interests include systems theory, applications of modeling and simulation, and modeling and simulation education. Dr. Mielke received the BS, MS, and PhD degrees in electrical engineering from the University of Wisconsin – Madison.

Menion Croll is a Senior Project Scientist at the Virginia Modeling Analysis & Simulation Center at Old Dominion University. His research interests include visualization and serious gaming. Mr. Croll received a BS from Virginia Tech and the MS from Old Dominion University, both in computer science.

Jesse Caldwell is a senior in the Department of Modeling, Simulation and Visualization Engineering at Old Dominion University. His interests include serious gaming, formal methods, and applications of modeling and simulation. He will receive the BS degree in Modeling and Simulation Engineering in May 2014.

Robert Greer is an Engineer and Manager with the Space and Naval Warfare Systems Center Atlantic. His technical interests include emerging geospatial data standards and visualization software. Mr. Greer received a ME in Computer Engineering at Old Dominion University and a BS in Electrical & Computer Engineering from the University of South Carolina.

William Labelle is an Engineer with the Space and Naval Warfare Systems Center Atlantic. He is the leader for the INS-R M&S effort for the Navy. His technical interests include navigation M&S and enhanced inertial sensor design. Mr. LaBelle received the MS & BS degrees in Computer Engineering from Old Dominion University.

Modeling and Simulation of Inertial Navigation Systems with Automated Interface Checking and Post-Processing Support

Yiannis Papelis and Menion Croll
Virginia Modeling, Analysis and Simulation Center
Old Dominion University
[ypapelis, mcroll]@odu.edu

James Leathrum, Roland Mielke and Jesse Caldwell
Modeling, Simulation and Visualization Engineering
Old Dominion University
[jleathru, rmielke, jcald013]@odu.edu

Robert Greer and William Labelle
Spawar System Center Atlantic
[robert.a.greer, william.labelle]@navy.mil

INTRODUCTION

Navigation is the process of establishing a current location and then determining how best to proceed to a desired destination. In early times, sailors were forced to maintain sight of land to facilitate navigation. Today, very complex navigation systems use multiple sensor technologies and high-speed computers to guide vehicles across vast distances. Due to the complexity of navigation systems and the need for sophisticated environments for test and evaluation, modeling and simulation frequently is used as the primary design and development tool. However, except for the development of several special-purpose libraries of common navigation components for use with general-purpose system analysis tools, little has been done to develop a comprehensive framework for navigation system design and development.

The purpose of this paper is to describe a framework that supports modeling and simulation of navigation systems. The framework is not tied to a specific system topology or component technology, but rather presents the engineer with a convenient and flexible environment in which to explore new design approaches. The framework supports hierarchical design at an arbitrary level of detail. Some additional novel features of the framework include a data dictionary that maintains signal consistency at device interfaces, a scenario generator that automatically facilitates Monte Carlo simulations, and a data-aware post-processor that supports appropriate statistical analysis and helpful data visualization capabilities.

The paper is organized as follows. In the second section, a brief description of current navigation technology is presented and the need for better simulation design and development tools is explained. Desirable features of a simulation-based design framework are identified. In the third section, the new simulation framework is presented. The overall framework architecture is described. Then our approach to data management and post-processing support is presented. Progress made in implementing a framework prototype is described in the fourth section. In addition, a preliminary case-study is presented to demonstrate how the framework is used. A conclusion is presented in the fifth section and includes an explanation of our vision for future enhancements of the framework.

BACKGROUND AND MOTIVATION

In this section, the present state of navigation system design is described and the reasons motivating the need for improved design environments are identified. Then, desirable features for an enhanced simulation-based framework for navigation system design are presented.

Current Navigation System Design

One of the mature technologies for navigation is the inertial navigation system (INS). An INS utilizes inertial sensors, accelerometers to measure accelerations along orthogonal axes and gyroscopes to measure angular rotation rates about these axes, to compute location with respect to a known starting position. The main advantage of an INS is that it is self-contained; it does not require the sensing of external quantities in order to function. The main disadvantage of an INS is that the inertial sensors exhibit small errors that result in a calculated position error that grows with elapsed time. The use of very expensive inertial sensors, such as ring laser gyroscopes, can reduce the sensor error, but even these systems eventually exhibit unacceptable error over time. Newer MEMS inertial sensor

technology has dramatically reduced cost, size, and power requirements of inertial sensors but at the expense of larger sensor errors.

Another mature technology for navigation is the global positioning system (GPS). A GPS utilizes timing information of signals received from satellite-based transmitters to triangulate position. The main advantage of a GPS is that it provides highly accurate and time-independent position information. The main disadvantage of GPS is that the system fails to work with interruption of received signals. This can occur naturally due to geographic obstacles or inclement weather conditions, or because of man-made obstacles such as buildings or intentional signal jamming and blocking. Many modern navigation systems utilize multiple sensor technologies. A common approach is to utilize inertial sensors aided by GPS and perhaps other sensor technologies (Gade. 2009). Depending on the level of integration among the components, such systems are referred to as loosely or tightly coupled INS, or deeply integrated GPS/INS units (the term ultra-tight coupling is also used) (Edwards, Clark, Bevly, 2010). In such systems, the INS provides highly accurate short term position information while the aiding sensors are used to maintain the long term error of the overall system within acceptable bounds.

The design and development process for integrated systems usually relies heavily on modeling and simulation. INS system components often are highly nonlinear and the system signals are represented as stochastic processes, making an analytical analysis difficult or impossible. In addition, design testing and evaluation requires a sophisticated test environment that is capable of synthesizing system inputs and then analyzing and visualizing system outputs. Both of these issues suggest extensive use of modeling and simulation, especially in the early stages of prototype design and development. One of the early descriptions of a GPS-aided INS is presented in (Orozco, Ruiperez, de la Cruz, and Aranda, 1995). A set of Matlab library components is defined and used to develop a simulation model for the system. In (Eck, Chapuis, and Geering, 2001), the design of an INS using low-cost inertial sensor components is presented. The concept of a supporting simulation environment is introduced in this context. One of the first detailed presentations of how to utilize an extended Kalman filter to integrate GPS data with an INS is given in (Giroux, Gourdeau, and Landry, 2005). A rapid prototyping environment, implemented in Matlab-Simulink, is described. Perhaps the most extensive effort to develop a generic simulation environment to support INS design is described in (Gade, 2004). Called NavLab, the system simulator supports hierarchical design with a variety of aiding sensor technologies. The simulator also includes provision for conducting prototype system test and evaluation. However, the navigation system architecture is fixed and support for varying system topology and adding new system components is somewhat limited.

Changing Design Environment

We believe that two emerging factors will influence extensively the environment in which the next generation of INS will be designed and developed. First, present-day navigation systems generally are acquired from a manufacturer as a total system solution. These systems would be described as self-contained, black-box, proprietary solutions. System modifications and upgrades can be made only by returning to the original manufacturer to initiate another purchase agreement. Next generation navigation systems are likely to employ a more modular or component-based architecture. Customers will have the flexibility to mix and match different system components, such as sensors, filters, and computing platforms, obtained through different vendors to realize the overall system. Second, the use of autonomous unmanned vehicles to address a rapidly expanding set of military and consumer applications will create a significant new market demand for navigations systems that are smaller, lighter, cheaper, and that are more capable than current systems. These systems will utilize new sensor types and technologies designed especially to address emerging applications. They will need to be more fault-tolerant and reliable. These requirements again suggest a flexible and modular design and development approach where components from different manufacturers are easily integrated.

This changing design environment requires development of more flexible and capable modeling and simulation design tools. We propose a simulation-based design and development framework to support this new design environment. The framework would host system component models from different manufacturers; where desired, component models could be defined at the input-output level to preserve proprietary design details. Components could be interconnected in a variety of system topologies; automated interface checking would be used to ensure component interoperability. In addition, the framework would provide a configurable and flexible test harness to test and evaluate system performance. A more detailed description of the requirements for this framework is presented in the following section.

Requirements for Design Framework

The proposed framework should support both the design process and the test and evaluation process associated with the development of a new navigation system. Therefore, the framework must provide libraries of INS components and a workspace where these components can be interconnected to form a complete system. In addition, the framework must provide the test harness necessary to stimulate the system under test and to analyze and visualize the system response. The proposed conceptual configuration of the framework is shown in Figure 1. The test harness includes components to generate input signals to stimulate the navigation system as well as analysis and visualization components to investigate the system output. The test harness also includes the framework controller. The navigation system model represents the system under test. The system under test may include the entire navigation system or just some subset of system components designated for testing. In that case, essential system components not included as part of the system under test are included as part of the test harness. The requirements for the framework are presented in this section categorized by framework functionality.

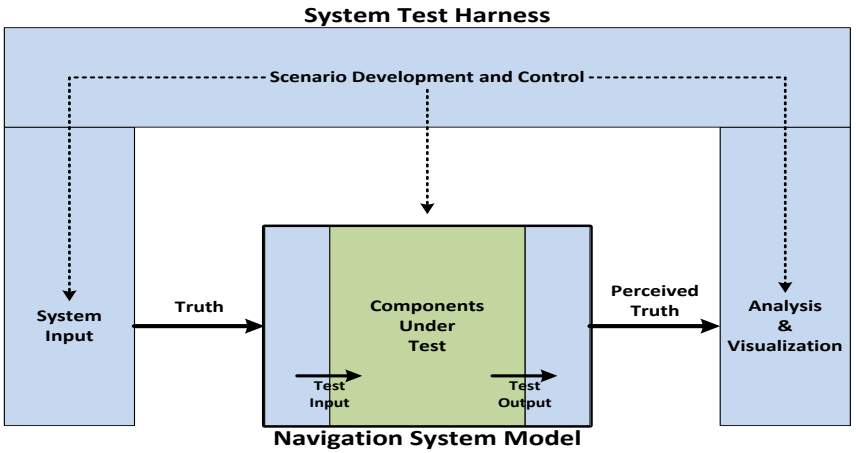

Figure 1. Framework Conceptual Configuration.

The framework should provide the following capabilities in a flexible and user-friendly user environment.

Expected Users. Framework users are expected to have an engineering background and familiarity with the following: navigation system terminology; system modeling and analysis; use and limitations of modeling and simulation of dynamical systems; and Monte Carlo analysis of stochastic systems.

Scenario Development and Control. The framework should support the notion of a "scenario" that includes: system and test harness configuration; input and output data file management; input trajectory generation; specification of test parameters and performance metrics; and output analysis and data visualization. System control and operation should be available to the user through a simple and flexible graphical user interface. The user should have the capability to select or define a scenario, configure and execute desired experiments, and analyze and display experimental results via this user interface.

System Input. The system input component should be able to generate a vehicle trajectory from a set of desired waypoints and descriptions of vehicle dynamics and control strategies. Alternately, the system should accept data files representing actual vehicle trajectories.

Simulation Core. The framework should support graphical block diagram development of various navigation system architectures and the execution of these system models using real-time or best-effort-time strategies. The simulation core consists of model libraries organized in logical groupings by function. The simulation core libraries also are used to implement components comprising the test harness.

System Execution. The framework should support sequential and concurrent execution of the primary model components: Input Generation; Model Execution; and Output Analysis. In sequential execution, each primary

model component is executed separately with an output data file from one component being used as the input data file for the next component. In concurrent execution, all three primary components are executing simultaneously with data flowing across the modules during execution.

Analysis and Visualization. The framework should support analysis and visualization functions required to display and/or compare performance characteristics of navigation system models. Analysis capabilities should support single-run and multiple-run simulation experiments. Various two-dimensional and three-dimensional data visualizations should facilitate views of truth verses predicted performance for a single system or a comparison of system alternatives under identical input conditions.

SIMULATION FRAMEWORK

Architecture Description

The framework is partitioned into three primary components: trajectory generator, simulation core, and the analysis and visualization component. The primary components are supported by a set of user tools that facilitate selecting, importing, and editing of data files as well as tools to configure and control simulation execution. The overall framework architecture is shown in Figure 2.

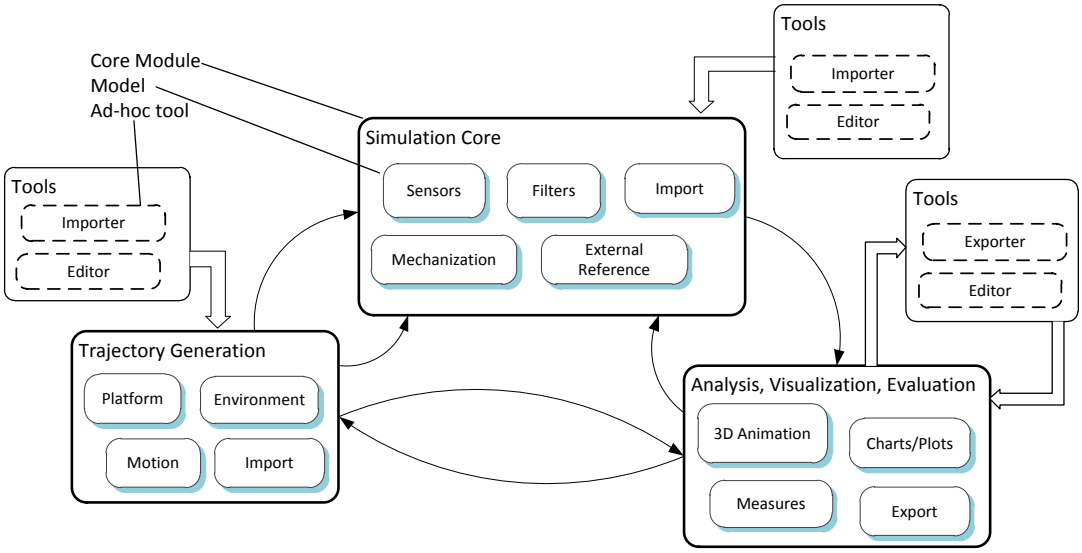

Figure 2. Framework Architecture.

The trajectory generator contains modules for modeling platform characteristics, environmental factors, platform control strategies, and trajectory generation. The simulation core contains modules for representing system sensors, mechanization strategies, and filter strategies. The analysis and visualization component contains modules to implement various analysis methods, 2D charting and data plots, 3D visualizations and animations, and management functions to configure system performance tests and system comparative evaluations.

We have selected Matlab-Simulink as the foundation upon which to construct the framework. Matlab is a high-level language that provided and interactive environment for numerical computation and simulation. Simulink is an environment that supports model-based design and simulation. It utilizes a graphical environment that facilitates specification of the navigation system and associated test harness through interconnected block. In addition, it includes a variety of solvers for simulating dynamical systems. Matlab-Simulink is a modeling and simulation environment that already is familiar to many engineers, and that supports development of the support capabilities we desire in the framework.

Interface Specification Management

A key aspect of the simulation framework is the use of a Data Dictionary (DD), a user-defined table that enumerates all possible input/output connections among blocks in the framework. The DD is a KxK matrix, where K is the

number of user-defined data types. Each entry in the DD specifies if connections between the respective data types are allowed:

$$\forall\, i,j\ \ DD[i][j] = \begin{cases} 1 \rightarrow i\ can\_connect\_to\ \ j \\ 0 \rightarrow i\ cannot\_connect\_to\ \ j' \end{cases} \qquad i < K, j < K$$

A strict compatibility DD would be a diagonal matrix, effectively indicating that only identical data types are compatible, but the designer is free to define the DD as required. For example, if a given input is compatible with more than one type of outputs, compatibility classes can be created by setting additional entries of the DD to 1.

The DD is an integral part of the framework because one of the requirements for a block to be usable in the framework is that all inputs and outputs be tagged with one of the entries in the DD. During the design process, any time a connection is made between blocks, the DD is consulted as to the validity of the connection; if the connection is not allowed an error message is displayed, along with a list of compatible outputs and the connection is rejected.

Using a DD provides several advantages when composing large systems by combining existing and newly developed modules. At a basic level, the DD constraints ensure that only appropriate data types are linked. For example, a module tagged as outputting an acceleration value is not allowed to connect to a module tagged as expecting a velocity value. In addition to basic compatibility check, the DD can also be used to ensure that input/output linkages obey appropriate axis conventions, utilize the same units, or have compatible time-steps. Doing so requires that the designer explicitly incorporate this functionality in the DD, which can be done by encoding such information in the tag itself. As an example, if a module expects as input an inertial acceleration three element vector represented in SI units, the tag could be labeled as ACC_3_INRT_SI, whereas the same acceleration represented in the chassis coordinate system would be labeled as ACC_3_CHAS_SI. Any number of variations can be used, as there is no inherent limitation to the size of the DD. Whereas this approach appears to be shifting the complexity to the user, in reality it exposes the existing need to properly document all input/output parameters associated with modular blocks that can be used in the overall design. In addition to simplifying the construction of a navigation system and providing confidence about the structure of the overall model, a carefully designed DD can act as an interface specification document by providing all required information about every input/output connection among modules.

For a simulation model that is designed from scratch, a prudent design strategy would be to begin by defining the entries of the DD before defining the model's structure. On the other hand, most activities for which the framework is envisioned to support do not begin with a blank model; instead utilize partially constructed models into which new modules need to be incorporated. To accommodate this, an existing DD (associated with an existing model) can be extended by merging a new DD (associated with an imported module). This merging operation involves extending the size of the DD to accommodate the new tags and the subsequent definition of the new cells. The two dictionaries may share tags, provided that the identical tags convey semantically identical meanings. If that is not the case, the incoming tags have to be renamed, something that can trivially be done before merging. If the two dictionaries do not share tags, or share tags that are semantically identical, then, there are three possible situations for each cell in the merged DD:

1) The new cell conveys compatibility between two tags both of which are defined in either of the two DDs
2) The new cell conveys compatibility between two tags both of which are defined in both of the DDs
3) The new cell conveys compatibility between two tags that belong to different DDs.

This operation is shown below, where the existing DD containing tags (a, b, c) is merged with a new dictionary containing tags (b,d), and for this example, tags b in the original and merging dictionaries are semantically identical.

$$\begin{array}{c} a \\ b \\ c \end{array}\begin{bmatrix} D_{a,a} & D_{a,b} & D_{a,c} \\ D_{b,a} & D_{b,b} & D_{c,b} \\ D_{c,a} & D_{c,b} & D_{c,c} \end{bmatrix} \begin{array}{c} \\ \\ \\ a \end{array} \begin{array}{c} \\ \\ \\ b \end{array} \begin{array}{c} \\ \\ \\ c \end{array} \quad MERGE \quad \begin{array}{c} b \\ d \end{array}\begin{bmatrix} D'_{b,b} & D'_{b,d} \\ D'_{b,d} & D'_{d,d} \end{bmatrix} \begin{array}{c} b \\ d \end{array} \quad = \quad \begin{array}{c} a \\ b \\ c \\ d \end{array}\begin{bmatrix} D_{a,a} & D_{a,b} & D_{a,c} & \overline{D_{a,d}} \\ D_{b,a} & D_{b,b} & D_{b,c} & D'_{b,d} \\ D_{c,a} & D_{c,b} & D_{c,c} & \overline{D_{c,d}} \\ \overline{D_{d,a}} & D'_{d,b} & \overline{D_{d,c}} & D'_{d,d} \end{bmatrix} \begin{array}{c} \\ \\ \\ a \end{array} \begin{array}{c} \\ \\ \\ b \end{array} \begin{array}{c} \\ \\ \\ c \end{array} \begin{array}{c} \\ \\ \\ d \end{array}$$

In case (1), the value from the original DD is copied. In the example above, the entry for $D_{c,c}$ only appears on the main dictionary (D) and can thus be copied to the merged output.

In case (2), which only occurs if the existing and new DDs share tags, if the entries are the same (i.e., either both 0 or both 1), then that value is copied to the merged DD. If the entries are different, a conflict has been identified; such a conflict is an indication that there are semantic differences between tags in the two DDs, something that must be handled manually. In the example above, this would happen if the value of $D_{b,b}$ is different than $D'_{b,b}$.

In case (3), the designer must assess the compatibility of the new relationship. In the example above, the value of the entry $\overline{D_{d,a}}$ would have to be defined after inspecting the relationship between tag a (in the existing DD) and tag d (in the incoming DD). A reasonable default value would be 0, as that indicates that unrelated tags are not compatible.

The value of aforementioned process for managing and merging DDs partially lies in the fact that it can be automated but most importantly because it forces the designer to evaluate and assess the correctness of the interfaces among all module input/output. By evaluating the relationship among tags, subtle differences in interfaces or conventions that would otherwise remain hidden can be identified and handled. The net result is that defining a well-documented interface becomes an intrinsic part of the design process, as opposed to an afterthought.

Parameter Variation

The model of a navigation system is rarely deterministic in nature because of the presence of stochastic noise and the use of probabilistic approaches for modeling system components. Implementing such models invariably leverages pseudo-random number generators; should the same initial seed be used, running the model multiple types will generate the same output, something that is desirable while debugging the simulation system itself. However, from the standpoint of the navigation system designer, varying the random seed is desirable as a means to better understand the operation of the system. The framework allows management of the random generation seeds thus providing flexibility to the designer.

In addition to managing random number seeding, the framework provides the ability to perform Monte-Carlo simulations while varying input parameters. By leveraging tags in the DD, the system is aware of which signals are considered inputs/outputs or model parameters. Once a model is completed, the system allows the designer to select a set of model parameters, specify sweep ranges and simulation sample sizes. A batch-mode simulation then takes place where model parameters are perturbed as specified and multiple simulation runs execute each with different pseudo-random seeds. The resultant outputs accumulate to an output database, and each instance is linked to the specific set of parameters associated with the simulation run.

Post-Processing Support

As the simulation is executed, estimated positioning data is accrued. In order to process these data, we can attach a post processing block. Post-processing blocks can perform any number of tasks depending on the purpose of a particular simulation run. It could add the results of the simulation run to an archive of previous simulation results, launch a visualization of some aspect, transform the data to another format or specification, or some combination of these options. The DD ensures that there will be no problems with data types or structure to impede whatever post processing method is chosen. In cases that the output consists of multiple execution runs, data can be shown in isolation or cumulatively.

The initial prototype chooses to implement a visualization of the estimated trajectory, along with a reference true trajectory, as the post-processing block. When the simulation has finished creating an estimated trajectory, the Common Geospatial Extensible Navigation Toolkit (COGENT) is automatically launched. The estimated trajectory is saved to a file, which is then sent to COGENT for visualization. Additional runs with altered random seeds can be performed, and each time the result will be added to the visualization. This enables to the user to easily visualize the results of a Monte Carlo experiment of the possible trajectories created by the navigation simulation, because the results of each trial are archived and added to COGENT.

Figure 3 shows the results of two simulation runs, with the solid red line depicting the true trajectory of the ship and the two dotted blue lines representing the estimated trajectory results of the INS simulation. For simplicity, only two output runs are shown, each with a different random seed. In cases where multiple simulation runs are available

after a Monte-Carlo simulation, they can all be displayed concurrently, including the possibility of grouping output runs according to which parameter was swept to generate the outputs.

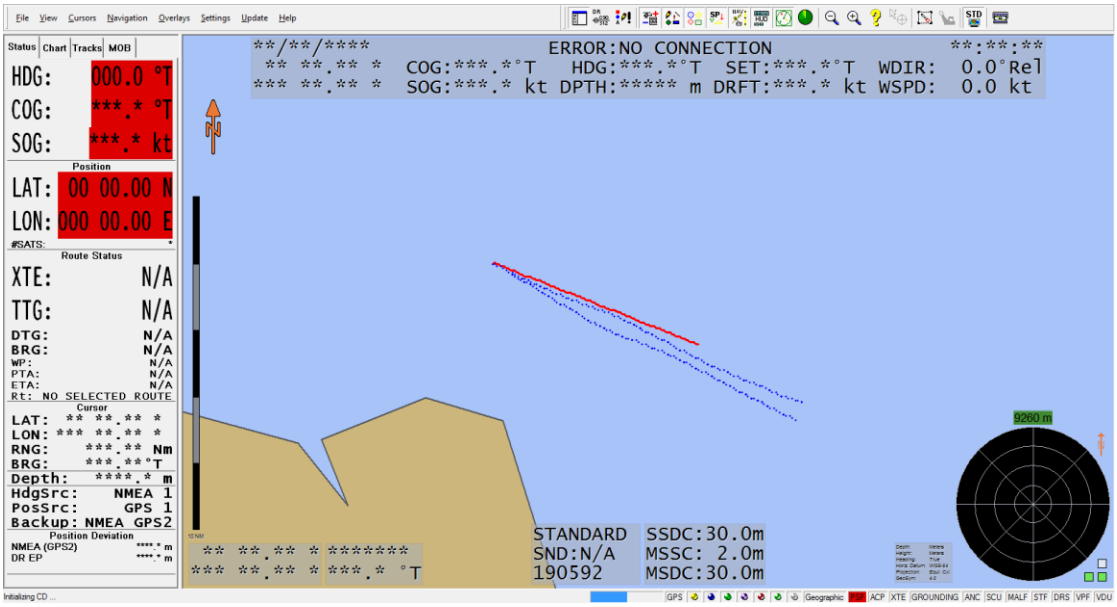

Figure 3. Post-Processing Visualization in COGENT.

PROTOTYPE IMPLEMENTATION

The simulation was implemented in Simulink, with the addition of various MatLab scripts. The environment was chosen to help integrate various model components, which may come from multiple parties. The simulation is comprised of a series of scripts and Simulink model blocks. The DD is defined in a set of Excel spreadsheets, which define the valid connections and port tags of each block in the simulation as shown in Figure 4.

8	position_m_ecef	velocity_m/s_ecef	acceleration_m/s2_ecef	orientation_rad	angularVelocity_rad	angularAcceleration_rad	position_m_lla	velocity_knots
position_m_ecef	1	0	0	0	0	0	0	0
velocity_m/s_ecef	0	1	0	0	0	0	0	1
acceleration_m/s2_ecef	0	0	1	0	0	0	0	0
orientation_rad	0	0	0	1	0	0	0	0
angularVelocity_rad	0	0	0	0	1	0	0	0
angularAcceleration_rad	0	0	0	0	0	1	0	0
position_m_lla	0	0	0	0	0	0	1	0
velocity_knots	0	1	0	0	0	0	0	1

Figure 4. Example connection matrix for the DD

The DD is constructed using a series of scripts and the two Excel spreadsheets described above. When the toolbox is opened, the DD goes through the spreadsheets and creates a set of maps of the block names and port types. Once a block is added to the model, a script is activated that checks if that block is defined in the DD and creates an instance of that block in the DD for the model if it is defined. The DD is used to ensure two blocks can interact by calling an action script when a user attempts to connect two blocks. The script looks at the port tags of the blocks and if they are compatible the DD allows the connection to occur. Otherwise, the DD generates an error message that is displayed to the user, warning them the two blocks cannot connect as shown in Figure 5.

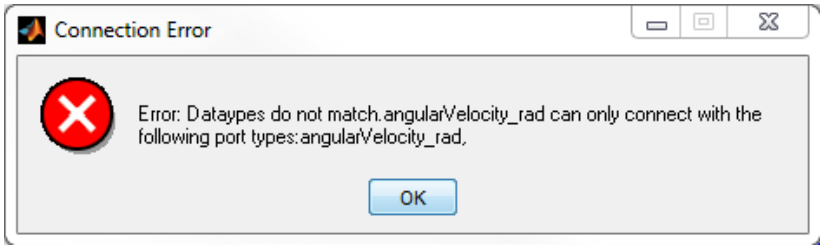

Figure 5. Sample Simulation Error Message.

Along with the scripts and spreadsheets that comprise the DD, a series of Simulink blocks have been developed and compiled into a Simulink toolbox. The toolbox has been divided into two main sections, the test bed and the system under test. The test bed section makes up the test harness for the simulation, supplying system inputs and analyzing system output. The system under test section contains all the components the user wishes to insert into the system. These blocks include INS models, sensor models, and various filters as seen in Figure 6. One of the key features of this framework is the ability of outside users to develop their own models and filters that can be easily inserted into the simulation, so long as they adhere to the DD.

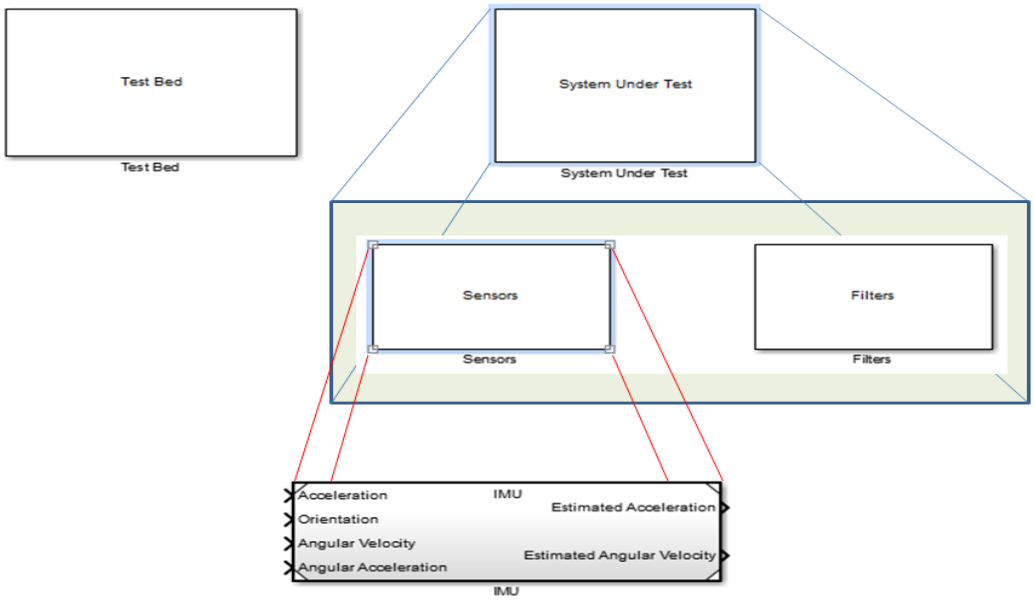

Figure 6. Simulink toolbox structure.

CONCLUSION

The paper described a framework that can be used to build navigation system models by combining model blocks of arbitrary granularity. The goal of this framework is to leverage modeling and simulation to allow users to experiment with alternative system structures, equipment choices, and filter designs. Key contributions of the framework include a flexible architecture that allows the designer to delineate the model under test versus the test fixture, the use of a data dictionary that helps maintain appropriate interface connections among system modules, an experiment control module that automates parameter sweeping and Monte-Carlo-type simulations as well as flexible input/output hooks to trajectory generation and visualization and post-processing tools.

At the time of the writing of this paper, a prototype of the framework has been built and the majority of the features have been incorporated into the prototype, with future work focusing on refining capabilities and construction of a larger example that will help validate the overall approach.

ACKNOWLEDGEMENTS

The authors wish to acknowledge support for this work from SPAWAR System Center Atlantic, Navigation and Geospatial Information & Services IPT, with funding administered through WR Systems, Ltd., Norfolk, VA.

The authors wish to acknowledge that research and development of some detailed navigation sensor models to test and operate with the framework are being developed by the Pennsylvania State University Applied Research Laboratory (PSU ARL) Navigation Research and Development Center (NRDC), Warminster, PA.

REFERENCES

Gade, K. (2009). Introduction to Inertial Navigation and Kalman Filtering. IAIN World Congress, Stockholm. Retrieved February 20, 2014, from http://www.e-bookspdf.org/download/kenneth-gade.html

Orozco, J., Ruiperez, P., de la Cruz, J. and Aranda, J. (1995). Modeling and Simulation of Navigation Systems: An INS Simulation Matlab Toolbox. Proceedings of the 1995 EUROSIM Conference, Vienna, Austria.

Eck, C., Chapuis, J., and Geering, H. (2001). Software-Supported Design and Evaluation of Low-Cost Navigation Units. Proceedings of the 8th Saint Petersburg International Conference on Integrated Navigation Systems, St. Petersburg, Russia.

Giroux, R., Gourdeau, R., and Landry, R. (2005). Inertial Navigation System/Global Positioning System Fusion Algorithm Design in a Fast Prototyping Environment: Towards a Real-Time Implementation. Canadian Aeronautics and Space Journal, Volume 51, Number 3.

Gade, K. (2004). NavLab, a Generic Simulation and Post-Processing Tool for Navigation. European Journal of Navigation, Volume 2, Number 4.

Edwards, W. L., Clark, B. J., Bevly, D. M., Implementation Details of a Deeply Integrated GPS/INS Software Receiver. Position Location and Navigation Symposium (PLANS), 2010, IEEE/ION, pp. 1137-1146, May 4-6 2010.

Common Geospatial Extensible Navigation Toolkit (Version 2.4.2.0) [Software]. (2012). SPAWAR Systems Center Atlantic.