REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
19-06-2019	Final	09/30/2016-02/28/2019

4. TITLE AND SUBTITLE
Distributed Verification and Validation Methods for Power System Dynamics and Control

5a. CONTRACT NUMBER
N000141613171

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kredo, Kurtis, II
Crosbie, Roy
Bednar, Richard
Mustafa, Hadil
Alavi, Zahrasadat

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The CSU, Chico Research Foundation
25 Main Street, Suite 203
Chico, CA 95928

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The research program proposed under this grant aimed to develop improved real-time simulation techniques that would be part of a distributed combination of simulated and hardware components.
Using a model provided by partners as a basis, significant progress was made in the development of techniques for high-level synthesis of FPGA code. This promises a convenient route to FPGA implementation without the need for FPGA coding expertise at the cost of some loss of execution speed. The model was implemented in the new Virtex 7 FPGAs and a preliminary analysis of timing and FPGA resource allocations was made.
With regard to verification and validation, the team followed its usual practice of using multiple runs of the simulation using different tools and methods to ensure close agreement within expected limits. Unfortunately, little progress could be made on model validation because of the very limited possibility of collaboration with project partners. Validation requires the availability of performance data from real hardware and this was not directly available to the Chico team.
Foundational work was completed on integrating the existing simulation system into a larger distributed simulation system. The original project goal was to integrate simulations into a distributed system involving interaction with remotely located hardware. Initial steps have been made in preparation for this kind of scenario. Improvements in the ability of one or more FPGAs performing simulations to communicate efficiently with each other and with external hardware is important in both meeting the original aim and with increasing the scalability of simulations.

INSTRUCTIONS FOR COMPLETING SF 298

15. SUBJECT TERMS simulation, FPGA, distributed system, high-level synthesis					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 42	19a. NAME OF RESPONSIBLE PERSON Kurtis Kredo II
a. REPORT UU	b. ABSTRACT UU	c. THIS PAGE UU			19b. TELEPONE NUMBER (*Include area code*) 530-898-4414

Distributed Verification and Validation Methods for Power System Dynamics and Control

ONR Grant: #N00014-16-1-3171

FINAL REPORT

Submitted by
The CSU, Chico Research Foundation, on behalf of
California State University, Chico

Technical Contacts
Dr. Kurtis Kredo II, PI; Dr. Roy Crosbie, co-PI
McLeod Institute of Simulation Sciences
California State University, Chico
400 West First Street; Chico, CA 95929-0003
Tel.: 530-898-4414 Fax: 530-898-4956 E-mail: kkredo@csuchico.edu

Administrative Contacts
David M. Hassenzahl, Ph.D.
Office of Research and Sponsored Programs
The CSU, Chico Research Foundation
CSU, Chico, 25 Main St.; Chico, CA 95929-0870
Tel.: 530-898-5700 Fax: 530-898-6804
E-mail: dhassenzahl@csuchico.edu

Table of Contents

1. Background and Research Aims

Researchers in the McLeod Institute of Simulation Sciences at California State University, Chico (CSU, Chico) have a history of setting benchmarks in the development of low-cost, high-speed, real-time (HSRT) simulation techniques [1, 2]. Starting in 1999 with a study based on arrays of digital signal processors (DSPs) that achieved frame times as low as 2 μs for a typical power electronic benchmark, the research has involved the development of new algorithms, methods for multi-rate real-time simulation, development of new high-speed interfaces for distributed hardware-in-the-loop (HIL) simulations, development of software support tools based on math analysis of model equations, and the use of field-programmable gate arrays (FPGAs) with frame times as low as 400 ns [3, 4, 5, 6]. This research has been supported since 2001 by a series of ONR Awards. Previous ONR support was received directly through ONR Awards N00014-11-1-0902 (Advanced Simulation Techniques Using FPGAs) and N00014-12-1-0376 (Technology Transfer of ONR-Funded Research to Support Simulation Activities at NSWC). The grant built on this experience to develop distributed verification and validation methods for power systems dynamics and control. The goal was to develop techniques and implementations within FPGAs to support the real-time execution of system models at small frame times and to interface the simulator to external systems. This effort required basic research into analytical methods, new algorithms, software engineering, and system architectures.

The Chico research program began with the limited goal of reducing the achievable frame times for real-time simulation of power-electronic systems by as much as an order of magnitude from the conventional 50 μs using only affordable, off-the-shelf components. When this had been achieved attention turned to the need to incorporate these high-speed simulations into simulations of more complete power systems having a broad dynamic range. This led to the development of techniques for multi-rate, real-time simulations. Further improvement in these techniques followed a decision to switch from DSPs to FPGAs. As experience has been gained with using FPGAs for simulation, the enormous potential of these devices to support advanced simulations became more evident. This potential is not limited to real-time applications. The highly parallel, enormously flexible architecture of the FPGA permits the development of processor architectures that can be customized to the target applications with huge implications for improving cost effectiveness in a range of application areas that involve the solution of large sets of differential and algebraic equations including finite elements, hydrodynamics, digital filtering and others.

The basic research continued to address improvements in techniques for high-speed, real-time simulation in applications of importance to the design of Navy ships. A major focus of the grant was to investigate techniques for high-speed, real-time simulation using small frame times through the implementation and verification of useful models selected by research partners. A goal of the research was to provide real-time operation of complex models at reduced frame times in the support of future simulation of high frequency (100 kHz) switching systems. Improved methods for handling non-linear model elements are also required.

Study of power system dynamics and control also benefits by the development of high-speed, real-time simulation systems. As part of a larger distributed system, the high-speed, real-time simulation system can support power system research by solving models of interest while interacting with hardware components, other simulated models, or a combination of the two. Distributed implementations also enable the study of how real-world delays and communication

overhead affect dynamics and control of power systems. To provide the greatest flexibility in overall system arrangement and location, Ethernet and TCP/IP were used. These common networking technologies enable the simulation system to connect to components in the same laboratory or at a great distance through the Internet. A major focus of the grant was to develop methods for integrating the simulation system to remote components over common networking technologies.

The research builds on the Chico team's experience with applying FPGAs to simulation problems. Considerable expertise has been developed in understanding and implementing methods that take maximum advantage of the unique properties of FPGAs. FPGAs provide users with an unmatched ability to design a processing structure that corresponds to the structure of the math model underlying the simulation. A sound basis has been established that provides experience in making decisions on key topics such as the balance of parallel and sequential execution that optimizes performance and the design of interface architecture and data transfer protocols that can so often be the critical factors affecting performance.

The research also builds on the many years of experience of key team members in developing simulation software, in a wide range of simulation applications, in computer architecture and interface design, in analytical methods in support of simulation, and in embedded system design.

The team continued to design and construct prototype systems in their laboratory at California State University, Chico using Xilinx Virtex FPGA products and development systems along with the companion software products.

The research leveraged the expertise of research partners at Drexel University and NAVSSES through collaboration in model selection. While additional collaboration was planned when the grant was awarded, the research partners did not participate beyond early discussions and had a limited role in the grant work. Research partners provided the System Model used throughout the grant for implementation and development. The System Model, shown in Figure 1, is based on a diesel generator backup for an electrical power system and contains an induction motor, an alternator, a diesel engine with controller, a transformer, power source, and loads.

The program of work was divided into three main areas of basic research in the methodology, design, and techniques to advance the capabilities and cost-effectiveness of simulation systems:

1. high-speed simulation techniques,
2. model verification,
3. and distributed system integration.

The following subsections further expand on each of these research areas and subsequent sections detail the results achieved through the grant.

The research program under this grant did not achieve all of the tasks outlined in the proposal due to limited research partner involvement and a reduction in resources allocated to the grant as the program officer redirected the research team to work on other efforts (ONR Grants N00014-19-1-2055 and N00014-19-1-2056). Of the $597,860 initially awarded when the grant started, only $212,301 (35.5%) was allocated to the grant. Additionally, the grant was proposed to cover a period of performance of 3 years, but was ended after 2 years 5 months. However, the program achieved many of the objectives and leveraged the knowledge gained through the grant in the new grants requested by the program officer.

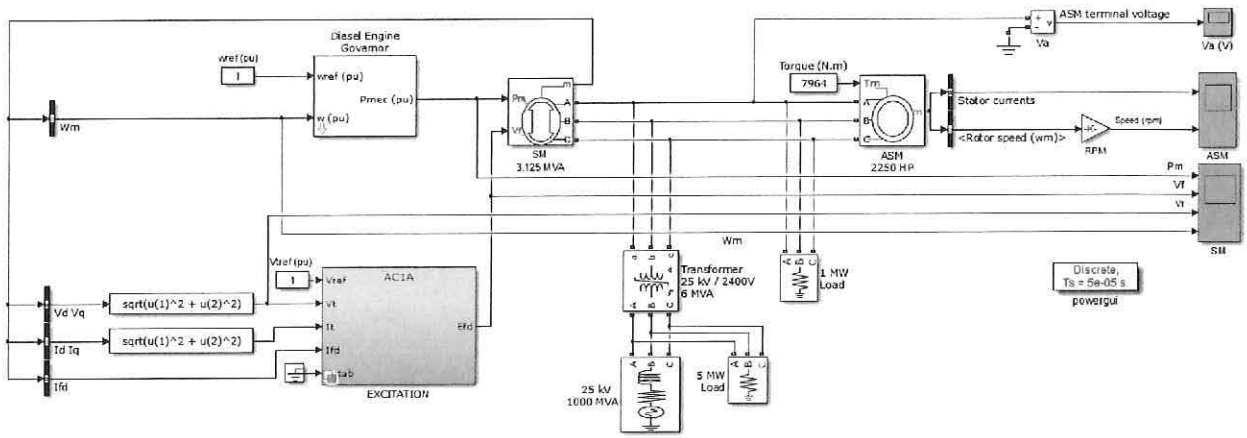

Figure 1: Complete System Model Diagram

1.1. High-Speed Simulation Techniques

The System Model in Figure 1 was used as the basis for implementation in FPGAs with the focus on high-speed simulation. Work on the System Model has followed three parallel paths: development of a mathematical representation of the model, an implementation in our existing simulation system, and an implementation using high-level synthesis techniques. In addition to providing broader resources for investigation of high-speed simulation techniques, the various paths provide independent sources for verification purposes.

A mathematical representation of the model is required for implementation within our existing simulation system. With models on previous projects we have been able to achieve frame rates below 1 µs, so the ability to implement the current model in our system is desirable. Additionally, a mathematical model enables us to adapt the hardware implementation to achieve the greatest performance and to easily adjust the simulation to changes in system parameters. However, this requires that we extract the underlying mathematical representation, primarily differential equations, of the model. The process of deriving and verifying the mathematical model takes considerable time given the sparse and sometimes incorrect documentation provided with Simulink and SimPowerSystems.

High-level synthesis design enables implementation in a shorter period of time at the disadvantage of a potentially slower final system compared to an implementation using our existing simulation system. High-level synthesis is a design methodology that enables users to design hardware for FPGAs using a high-level language, such as C or C++, instead of a traditional hardware description language, such as VHDL or Verilog. We leveraged high-level synthesis techniques by generating a C program version of the System Model using the Simulink Coder toolbox and, after significant manual modifications to meet requirements of high-level synthesis tools, implemented the model in an FPGA using the Vivado and Vivado HLS programs from Xilinx. Figure 2 shows the process used for this project.

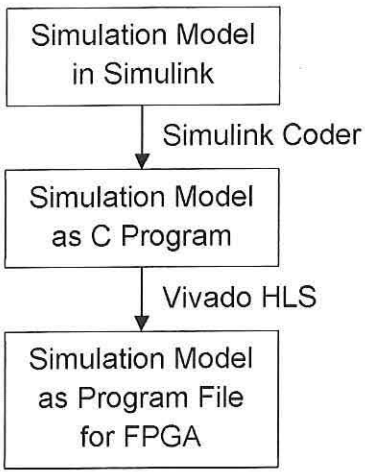

Figure 2: High-Level Synthesis Implementation Process

While high-level synthesis provides a mechanism for accelerated system implementation, it still requires users to manually modify model programs generated by Simulink Coder and have some knowledge of FPGA design in order to generate an FPGA configuration file. A transparent model translation tool that performs these modifications for the user would enable those without FPGA expertise to accelerate their simulations and final system implementation using FPGA technology. Toward that end, we investigated how a translation tool might help support automated implementation of Simulink models in FPGAs.

1.2. Model Verification

Verification of the simulation results ensure confidence in the use of the developed simulation system. Multiple methodologies may be required to perform verification depending on the modeled component or system, with comparison to analytical models, models implemented in other simulation systems, and experimental data possible. The nature of the chosen verification method places different requirements on the FPGA simulation system for the collection or comparison of data with some reference. Ensuring the accurate processing of data for verification purposes requires investigation of software and FPGA methods for bulk data handling and comparison at reasonable speeds. Work on this project has verified model implementations by comparing results across the implementations. Supporting these comparisons has required development of data processing and collection mechanisms appropriate for each implementation, including programs to parse network traffic, Verilog testbenches, and scripts to calculate statistical result differences. Results were compared between implementations in Simulink, C software generated by Simulink Coder toolbox, HLS implementation, and our existing simulation system.

Model verification was limited due to the limited participation of research partners, who were planned to have developed verification methodologies and provide experimental data or analytical models for comparison purposes.

1.3. Distributed System Integration

CSU, Chico research partners at NAVSSES and Drexel University have an interest in using the FPGA simulation system as a critical component of a distributed system for the study of the dynamics and control of power systems. The CSU, Chico FPGA simulation system could play multiple roles within a larger distributed system for research partners, including providing faster simulation results than existing systems, enabling hardware-in-the-loop simulation, supporting research on distributed control systems with real delays, and providing a platform for faster and simpler verification and validation of system components and models. Integration into the distributed system could occur through many possible direct or indirect communication interfaces. To support the broadest range of system arrangements, the FPGA simulation system interfaces to the broader distributed system through the IP and TCP protocols over an Ethernet network, the most common communication interfaces used by computer systems. Supporting these common protocols enables the distributed system to operate over both local connections within a single lab or remote operation over the Internet. Previous research has integrated the FPGA simulator into other environments, including the Virtual Test Bench (VTB) and Simulink. Additionally, basic network connectivity was previously developed allowing communication with a single, fixed PC over an Ethernet network using IP and UDP networking protocols.

Efforts in this project focused on expanding the network capabilities of the FPGA system to connect using the IP and TCP protocols using an embedded ARM processor in Zynq FPGAs and developing the interface between the ARM processor and FGPA logic for low latency communication. The Zynq FPGA was selected to leverage the processing power of an embedded ARM processor, which provides greater computational and memory resources than a soft processor, such as Microblaze. Additionally, the embedded ARM processor allows for execution of software within the FPGA for tasks, such as the TCP/IP networking stack, that are poorly suited for implementation within the reconfigurable FPGA logic.

2. High-Speed Simulation Techniques

Investigation of high-speed simulation techniques was based upon the System Model provided by research partners, Figure 1, and has followed three parallel paths: development of a mathematical representation of the model, an implementation in our existing simulation system, and an implementation using high-level synthesis techniques.

2.1. Mathematical Representation

We completed the analysis of most System Model components, including the induction motor, alternator, diesel engine governor, AC1A Excitation unit, and loads. Previous research derived models similar to the induction motor and alternator, so only verification was required for those components. The mathematical representations of all available components were verified against their equivalent Simulink blocks over the parameters specified by research partners. Appendix A: Diesel Engine Governor Model provides the detailed analysis of the Diesel Engine Governor and Appendix B: AC1A Model provides the analysis and model for the AC1A excitation unit. The only significant component without a verified mathematical representation is the transformer, which

was not used in initial implementations. During the analysis of component models, the project explored alternative implementation methodologies for high-speed implementation and cataloged implementation requirements for simulation systems that might implement these models.

For the Diesel Engine Governor, complex and real arithmetic representations were derived and compared in several configurations. The optimal configuration decouples most state variables to perform calculations in parallel while performing only real arithmetic operations. Other configurations decoupled all state variables, which requires fewer mathematical operations, but uses complex arithmetic at a considerable overhead. Appendix A: Diesel Engine Governor Model details the model configurations and performance comparisons.

The AC1A Excitation component analysis presented multiple non-linear operations for consideration or adaptation in an FPGA simulation system, including square root and conditional values. Appendix B: AC1A Model details how square root calculations were successfully replaced with a lookup table and how additional functionality is required to implement conditional calculations (if / switch statements) in an FPGA. While not completed in this project, conditional calculations might be implemented in an FPGA through additional state variables similar to integer linear programming representations or in combination with saturation operations.

2.2. Implementation in Existing Simulation System

Implementation of the System Model in the existing simulation system requires a mathematical representation of the system and the transfer of the simulation system to the new Virtex 7 FPGAs used for the project. These operations occurred in parallel over the project, but were not united due to the reduction in resources and time. Much of the development on porting the existing system to the new hardware was accomplished by student research assistants hired by the project.

Analysis of the existing simulation system in the new FPGAs yielded some preliminary performance measurements that will prove useful in scaling models to larger systems in subsequent projects. Table 1 lists timing measurements for the simulation system broken down by major functional operations that occur in each time step for two different potential system clock frequencies.

Further analysis considered how FPGA resources would scale with problem size, defined as the number of equations in the mathematical model. Scaling was dominated by two resources: the clock cycles per matrix multiply and the number of digital signal processing (DSP) blocks within the FPGA. At the current size of 32 equations, the simulator takes 104 clock cycles for each matrix multiply and 448 DSP blocks of the available 900 DSP blocks (approximately 50%). At 50 equations the matrix multiply would take 158 clock cycles and 79% of DSP blocks and at 63 equations 197 clock cycles and 99% of DSP blocks are required. While there are enough resources to theoretically implement 63 equations within a single FPGA, it is unlikely that the simulation would close timing at the high resource utilization of 99%. A practical upper limit within a single FPGA would be 50 or fewer equations.

Table 1: Timing Measurements for FPGA Simulation System

Operation	Clock Cycles	200 MHz Clock	125 MHz Clock
Send Outputs	34	170 ns	272 ns
Matrix Multiply	104	520 ns	832 ns
Non-linear Calculation	29	145 ns	232 ns
Receive Inputs	1	5 ns	8 ns
Total	**168**	**840 ns**	**1,344 ns**

2.3. High-Level Synthesis Implementation

To accelerate implementation of the model the project also pursued the use of high-level synthesis design techniques by generating a C program version of the Simulink model using the Simulink Coder toolbox. After significant modification, the C program is then used by the Vivado HLS program from Xilinx to program the FPGAs.

An initial implementation of a subset of the System Model, shown in Figure 3, was completed and showed promising results. The reduced Partial System Model consists of the System Model with the transformer, 1000 MVA source, and 5 MW load removed. Table 2 presents performance results when using this implementation. The results indicate we achieved faster execution times than Simulink at a comparable accuracy with only moderate effort in system optimization. Our initial implementation executed one time step every 100 µs for reasonable simulation frame times. After some optimizations, the system executes one time step in approximately 20 µs. Thus, our system can execute at real time for frame sizes greater than 20 µs and executes faster than Simulink for all frame sizes. Initial success using HLS design techniques resulted in a publication [7] and motivated efforts in this area.

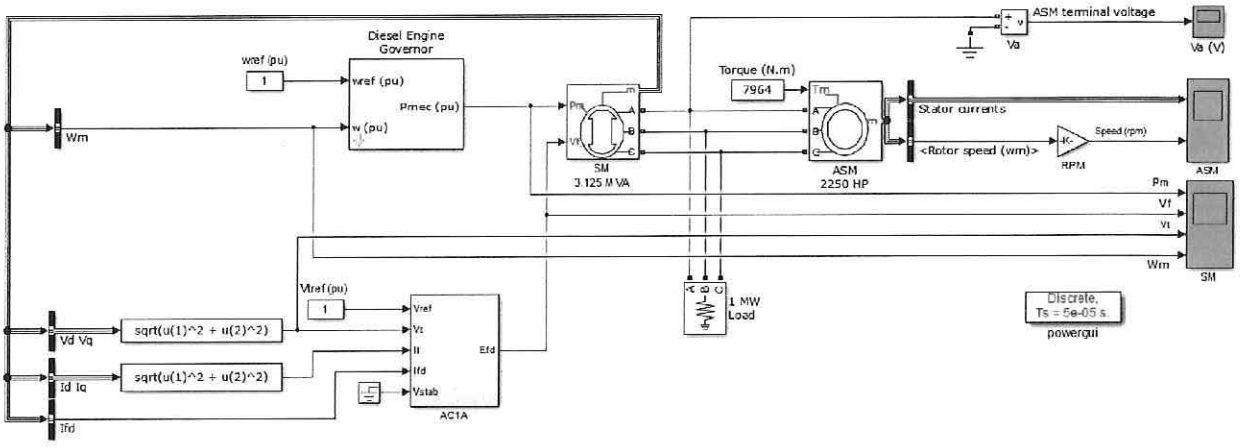

Figure 3: Partial System Model Diagram

Table 2: Execution Time and Error for HLS Implementation and Simulink Simulation of Partial System Model (Figure 3). Simulink results at 1 μs used as reference for all error results.

Time Step Size	Execution Time		Relative Error	
	Simulink	High-Level Synthesis (HLS)	Synchronous Machine Line Voltage	Asynchronous Machine Stator Current
100 μs	6.36 s	1.93 s	4.64 %, both	15.55 %, both
75 μs	8.10 s	2.59 s	2.78 %, both	6.32 %, both
50 μs	11.59 s	3.88 s	1.52 %, both	3.97 %, both
25 μs	22.07 s	7.80 s	0.63 %, both	1.51 %, both
5 μs	107.54 s	42.57 s	0.09 %, both	0.23 %, both
1 μs	525.68 s	274.27 s	Simulink reference, HLS 1.10×10^{-9} %	Simulink as reference, HLS 9.80×10^{-9} %

Progress continued with the implementation of the complete System Model (Figure 1) using HLS techniques. Performance results for this implementation showed an execution time improvement of 5X compared to Simulink with relative errors of approximately 10^{-11} % compared to Simulink at the same time step size. Further, Simulink can maintain real-time operation down to time step sizes of approximately 40 μs while the FPGA-based simulation can maintain a real-time operation down to step sizes of approximately 6.5 μs. Detailed results are located in Table 3 and a corresponding publication [8].

While high-level synthesis provides a mechanism for accelerated system implementation, it still requires users to manually modify model programs generated by Simulink Coder with some knowledge of FPGA design in order to generate an FPGA configuration file. A transparent model translation tool that performed these modifications for the user would enable those without FPGA expertise to accelerate their simulations and final system implementation using FPGA technology. Toward that end, we developed an initial translation tool that performs a small subset of the required modifications. The tool takes a C program generated by Simulink Coder and manipulates the source files to remove dynamic memory allocation, which is not supported in high-level synthesis methodologies. The tool uses the LLVM and clang compiler suite to build an abstract syntax tree for the program and then manipulates the tree to produce the desired modifications. For the preliminary tool, this involves removing calls to `calloc` and `free`, adding static memory allocation statements, and performing data type and size detection. The preliminary tool performs as expected and shows promise in supporting automated simulation implementation. Additionally, development of the automated translation tool has provided guidelines on further modifications that are required or would be beneficial for implementation in FPGAs. Development of the tool resulted in a publication [9].

Table 3: Execution Time and Error for HLS Implementation and Simulink Simulation of Complete System Model (Figure 1). Simulink results at the same time step size used as reference for HLS error results.

Time Step Size	Execution Time		Relative Error		
	Simulink	High-Level Synthesis (HLS)	Synchronous Machine Line Voltage	Synchronous Machine Field Voltage	Diesel Engine Output
100 µs	3.70 s	0.64 s	4.32×10^{-11} %	7.28×10^{-12} %	4.37×10^{-11} %
75 µs	4.69 s	0.85 s	1.60×10^{-11} %	3.26×10^{-12} %	1.13×10^{-11} %
50 µs	6.58 s	1.78 s	2.10×10^{-11} %	4.66×10^{-12} %	5.91×10^{-11} %
25 µs	12.47 s	2.56 s	2.34×10^{-11} %	2.58×10^{-12} %	4.73×10^{-11} %
5 µs	59.37 s	12.30 s	1.74×10^{-11} %	1.47×10^{-12} %	3.48×10^{-11} %

3. Model Verification

While initial model verification goals were reduced due to the limited participation of research partners, verification occurred across all implementation and development efforts. Verification across multiple implementations and representations required development of data collection and processing appropriate for the task.

The mathematical representation for System Model components underwent verification against corresponding Simulink blocks individually. In these instances, a Matlab m-code file simulated the component using the mathematical representation and saved appropriate signal values to a file at each time step. Graphical comparison of the results indicated whether the mathematical model accurately represented the Simulink block. Appendix A: Diesel Engine Governor Model and Appendix B: AC1A Model illustrate this verification approach.

High-level synthesis tools have a testing infrastructure to compare implementations throughout the process, including the model as a C program and the emulated implementation within the FPGA. Verification of HLS implementations occurred by comparing results with Simulink and with the C program generated by Simulink Coder.

Testing implementations within the FPGA occurred in two ways. First, testbenches enabled the functional verification of the FPGA implementation compared to Simulink results. Secondly, the project created programs that parse the network packets generated by the hardware FPGA implementation to extract the raw data for comparison to results from Simulink. These efforts allowed for verification of the FPGA implementation and simplifies the data collection process.

HLS and FPGA results were compared to reference values statistically using relative error, as described in published work [7, 8].

4. Non-Linear Operations

Support for non-linear functionality has focused on the trigonometric functions sine and cosine. The project completed implementation and comparison of three methods for computation of sine and cosine in FPGAs, with each providing a different combination of computational latency, resource requirements, and accuracy. Our approaches include calculating sine and cosine through a lookup table, the Coordinate Rotation Digital Computer (CORDIC) algorithm, and a modified lookup table technique based on sum-of-angle identities. Neither of the lookup tables use interpolation.

The modified lookup table method relies on the sum of angle identity in the equation below.

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$

For small β, $\sin\beta \approx \beta$ and $\cos\beta \approx 1$, which yields the approximation below, which also includes a similar approximation for cosine.

$$\sin(\alpha + \beta) \approx \sin\alpha + \beta\cos\alpha \qquad \cos(\alpha + \beta) \approx \cos\alpha - \beta\sin\alpha$$

These approximations allow for multiple optimizations across accuracy, resource utilization, and latency in FPGAs. Comparisons in this project focused on accuracy and resource utilization.

Performance comparisons, detailed in Table 4, indicate that CORDIC has the longest latency, which is problematic for real-time operation, but has a fixed accuracy and resource requirement. Results from Simulink sine function from 0 to $\frac{\pi}{2}$ were used as reference values. A standard lookup table implementation has the smallest latency with lower accuracy and resource requirements compared to CORDIC. The modified lookup table approach provides performance between the other techniques by trading resource requirements and latency for the multiply and addition operations for increased accuracy. The Modified lookup table approach used the same lookup table resources as the standard lookup table method. Table 4 details the performance results for the three methods evaluated.

Table 4: Trigonometric Function Implementation Performance Measurements. Simulink used as reference.

Resource	CORDIC	LUT	Sum of Angles
Clock Cycles	73	3	13
LUTs	9452	578	2918
Flip Flops	8657	466	4090
BRAMs	0	4	4
DSPs	0	13	39
Maximum Relative Error	3.9×10^{-12} %	0.12 %	1.2×10^{-4} %

5. Distributed System Integration

The project successfully expanded the networking capability of the existing simulation system so it may be integrated into a distributed simulation system. Implementation of full network processing, particularly the TCP standard common across the Internet, within traditional FPGAs would require a lengthy and complex design process with an uncertain performance benefit, so the current simulation system was expanded to include FPGAs with embedded hardcore ARM processors. Implementation of network processing can now occur in the ARM processor while the FPGA logic executes the simulation models like previous systems.

Moving network processing to an embedded ARM processor provides several advantages. First, network processing is easier to implement and update in software that runs on the ARM processor. Second, extending the network capabilities is easier due to the prevalence of software already developed for computer networking. Features such as Dynamic Host Configuration Protocol (DHCP) and Address Resolution Protocol (ARP) are now available to the simulation system while they were difficult to integrate in the previous system. These features are critical for operation within a distributed system across computer networks and the Internet.

Communication between software running on the embedded ARM processor and the simulation system within the FPGA logic occurs through a high-speed bus using Direct Memory Access (DMA). Packets that arrive from the network are parsed by the networking stack and the relevant internal packet data is sent to the simulation logic. A similar operation occurs in reverse for data sent from the simulation logic to the network.

Throughout the integration and development process low latency performance was a primary goal. Using a multi-threaded networking application and shared memory operations yielded the best performance. Table 5 presents timing measurements for data transferred between the embedded ARM processor and the simulation logic within the FPGA. Measurements were made by recording hardware timer values at the start and end of each operation and averaged over 200,000 total transfers. The results indicate that communication across this bus should only be used for communication at larger time step intervals.

Table 5: Transfer Latency between ARM and Processor Logic in Zynq FPGA

Direction	Latency for 32 Transfers of 64 bits		
	Minimum	Average	Maximum
FPGA Logic to ARM	78.45 µs	78.97 µs	94.70 µs
ARM to FPGA Logic	105.16 µs	106.34 µs	120.86 µs

6. Summary and Conclusions

The research program proposed under this grant aimed to develop improved real-time simulation techniques that would be part of a distributed combination of simulated and hardware components. The use of Ethernet and TCP/IP networking technologies enables the simulation system to connect to components in the same laboratory or at a great distance through the Internet.

The research did not complete all the tasks outlined in the proposal due to limited research partner involvement and a reduction in resources allocated to the grant as the program officer redirected the research team to work on other efforts. The period of performance was reduced from 3 years to 2 years 5 months and resources were reduced by more than half. However, the program achieved many of the objectives and leveraged the knowledge gained through the research in support of the research program that replaced it.

Using the Simulink model provided by partners NAVSSES as a basis, significant progress was made in the development of techniques for high-level synthesis of FPGA code. This promises a convenient route to FPGA implementation without the need for FPGA coding expertise at the cost of some loss of execution speed. The model was implemented in the new Virtex 7 FPGAs and a preliminary analysis of timing and FPGA resource allocations was made.

With regard to verification and validation, the team followed its usual practice of using multiple runs of the simulation using different tools and methods to ensure close agreement within expected limits. Any departure from expected performance is thoroughly investigated to determine the cause. Unfortunately, little progress could be made on model validation because of the very limited possibility of collaboration with project partners. Validation requires the availability of performance data from real hardware and this was not directly available to the Chico team.

Foundational work was completed on integrating the existing simulation system into a larger distributed simulation system. The original project goal was to integrate simulations into a distributed system involving interaction with remotely located hardware. Initial steps have been made in preparation for this kind of scenario. Improvements in the ability of one or more FPGAs performing simulations to communicate efficiently with each other and with external hardware is important in both meeting the original aim and with increasing the scalability of simulations enabling them to handle much larger models necessary for representing a complete ship power system.

7. References

[1] R. E. Crosbie, J. J. Zenor, D. Word and N. G. Hingorani, "Fast Real-Time DSP Simulations for On-Line Testing of Hardware and Software," *Modeling and Simulation,* vol. 3, no. 4, 2004.

[2] D. Word, J. J. Zenor, R. Bednar and N. G. Hingorani, "High-Speed Real-Time Simulation for Power Electronic Systems," *Simulation: Transactions of the Society for Modeling and Simulation International,* vol. 84, no. 8/9, pp. 441-456, 2008.

[3] R. E. Crosbie, J. J. Zenor, R. Bednar, D. Word and N. G. Hingorani, "Multi-Rate Simulation Techniques for Electric Ship Design," in *IEEE Electric Ship Technology Symposium*, Arlington, VA, 2007.

[4] R. E. Crosbie, J. J. Zenor, D. Word, R. Bednar and N. G. Hingorani, "Advances in High-Speed Real-Time Multi-Rate Simulation Techniques for Ship Power Systems," in *IEEE Electric Ship Technologies Symposium*, Baltimore, MD, 2009.

[5] R. E. Crosbie, "Using Field-Programmable Gate Arrays For High-Speed Real-Time Simulation," *International Journal of Modeling Simulation and Scientific Computing,* vol. 1, no. 1, 2010.

[6] K. B. Kredo II, J. J. Zenor, R. Bednar and R. E. Crosbie, "FPGA-Accelerated Simulink Simulations of Electrical Machines," in *IEEE Electric Ship Technologies Symposium*, Old Town Alexandria, VA, 2015.

[7] K. B. Kredo II, R. Bednar, R. E. Crosbie and J. J. Zenor, "Techniques for Accelerating Simulations of Electric Ships," in *IEEE Electric Ship Technologies Symposium*, Arlington, VA, 2017.

[8] H. Mustafa, K. B. Kredo II, R. E. Crosbie, R. Bednar and Z. Alavi, "Real-Time FPGA Simulation of Electric Ship Power System Using High-Level Synthesis," in *IEEE Electric Ship Technologies Symposium*, Alexandria, VA, 2019.

[9] K. B. Kredo II, H. Mustafa, R. E. Crosbie, R. Bednar and Z. Alavi, "Toward Automated Simulink Model Implementation and Optimization using High-Level Synthesis for FPGAs," in *IEEE Electric Ship Technologies Symposium*, Arlington, VA, 2019.

Appendix A: Diesel Engine Governor Model

This appendix outlines efforts to find a set of differential equations in a form suitable for multi-rate simulation in an FPGA environment. Simulation results for the Diesel Engine Governor block using the Simulink representation are compared with those generated using three Matlab m-coded differential equation representations of the system.

Mathematical Models of the Diesel Engine Governor

Figure 4 illustrates the internal provides additional information about the Diesel Engine Governor block. A generalized transfer function representation of the linear portion of Figure 4 is shown below:

$$\frac{\Gamma_g(s)}{\Delta\omega(s)} = K' \frac{(s+\frac{1}{T_3})(s+\frac{1}{T_4})}{(s^2+\frac{s}{T_2}+\frac{1}{T_1T_2})(s+\frac{1}{T_5})(s+\frac{1}{T_6})s} \tag{A1}$$

The output $\Gamma_g(s)$ can be viewed as the motor generated torque before limiting and delaying operations. The input $\Delta\omega(s) = \omega(s)_{ref} - \omega(s)$ is the error signal driving the governor-motor. The transfer function denominator in (A1) does not contain a constant term. This has the effect of causing the torque to increase without limit if a constant input $\Delta\omega(t)$ is applied.

The transfer function in (A1) can be converted to a set of linear differential equations in several different ways. For multi-rate simulation, we would like the differential equations to be uncoupled in the sense that the differential equation for one state variable does not depend on any of the other state variables. When the resultant differential equations are discretized into first order difference equations, this means a small integration step size can be used in a difference equation with rapidly changing dynamics, and a large step size in a difference equation with slowly

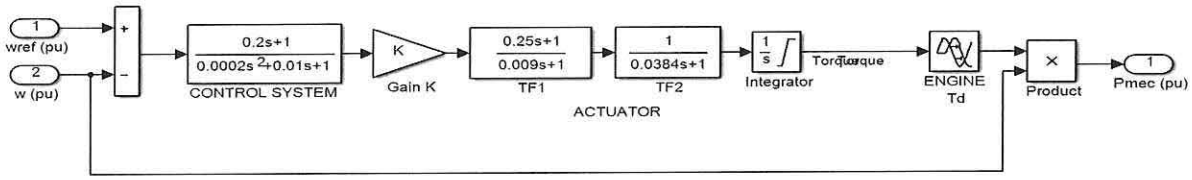

Figure 4: Diesel Engine Governor Block Details

changing dynamics. This decoupling can be achieved using a similarity transformation. However, for a low order, single-input, single-output system such as the one shown in (A1), we can obtain the decoupled equations in a simpler way using a partial fraction expansion as shown below.

The quadratic term in the denominator of (A1) can be factored as shown in (A2).

$$s^2 + \frac{s}{T_2} + \frac{1}{T_1T_2} = (s+p_1)(s+p_2) \tag{A2}$$

where

$$p_1 = \frac{1}{2T_2}\left(1+\sqrt{1-\frac{4T_2}{T_1}}\right),\ p_2 = \frac{1}{2T_2}\left(1-\sqrt{1-\frac{4T_2}{T_1}}\right) \tag{A3}$$

From (A3) we see that p_1 and p_2 will be complex if $T_1 < 4T_2$.This is indeed the case for the Navy selected values of T_1 = 0.01 and T_2 = 0.02.

In a similar manner, let

$$p_3 = \frac{1}{T_5},\ p_4 = \frac{1}{T_6},\ p_5 = 0,\ z_1 = \frac{1}{T_3},\ z_2 = \frac{1}{T_4},\ \text{and}\ K' = \frac{KT_3T_4}{T_1T_2T_5T_6} \tag{A4}$$

Then, the transfer function in (A1) can be written as:

$$\frac{\boldsymbol{\Gamma_g}(s)}{\boldsymbol{\Delta\omega}(s)} = K' \frac{(s+z_1)(s+z_2)}{(s+p_1)(s+p_2)(s+p_3)(s+p_4)(s+p_5)} \tag{A5}$$

The next step is to do a partial fraction expansion of the right side of (A5). The result is shown below:

$$\frac{\boldsymbol{\Gamma_g}(s)}{\boldsymbol{\Delta\omega}(s)} = \frac{c_1}{s+p_1} + \frac{c_2}{s+p_2} + \frac{c_3}{s+p_3} + \frac{c_4}{s+p_4} + \frac{c_5}{s+p_5} \tag{A6}$$

where $c_1 = \dfrac{-K'(z_1-p_1)(z_2-p_1)}{(p_2-p_1)(p_3-p_1)(p_4-p_1)p_1}$ $c_2 = \dfrac{-K'(z_1-p_2)(z_2-p_2)}{(p_1-p_2)(p_3-p_2)(p_4-p_2)p_2}$

$c_3 = \dfrac{-K'(z_1-p_3)(z_2-p_3)}{(p_1-p_3)(p_2-p_3)(p_4-p_3)p_3}$ $c_4 = \dfrac{-K'(z_1-p_4)(z_2-p_4)}{(p_1-p_4)(p_2-p_4)(p_3-p_4)p_4}$ $c_1 = \dfrac{K'z_1z_2}{p_1p_2p_3p_4} = K$ (A7)

A graphical representation of (6A) is shown in Figure 6. Also shown are the state variables $\mathbf{X}_1(s)$, $\mathbf{X}_2(s)$, ..., $\mathbf{X}_5(s)$ which are taken to be the outputs of the subsystem blocks. Each subsystem block can be inverse Laplace transformed into a differential equation as shown for $\mathbf{X}_1(s)$ in (A8) below.

$$\dot{x}_1(t) = -p_1x_1(t) + \Delta\omega(t) \tag{A8}$$

The same procedure can be followed for the remaining terms in (A6) to give us the set of differential equations shown in Figure 5. The differential equations can also be put into standard state variable representation as shown in (A9) and (A10) below. The diagonal system matrix in (A9) is another way of showing the uncoupled state equations.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_5(t) \end{bmatrix} = \begin{bmatrix} -p_1 & 0 & 0 & 0 & 0 \\ 0 & -p_2 & 0 & 0 & 0 \\ 0 & 0 & -p_3 & 0 & 0 \\ 0 & 0 & 0 & -p_4 & 0 \\ 0 & 0 & 0 & 0 & -p_5 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \Delta\omega(t) \tag{A9}$$

$$\gamma_{\mathbf{g}}(t) = [c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5] \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} \tag{A10}$$

The differential equations in Figure 5 or in (A9) can be solved in any order using different integration methods. They are in a form that can be solved numerically with either real or complex values for p_1 and p_2 as shown in Figure 5. However, if p_1 and p_2 are complex (which occurs when $T_1 < 4T_2$), then an FPGA implementation will require the ability to do complex arithmetic to solve the difference equations that follow from the discretization of the differential equations. In addition, the desired output torque $\gamma_{\mathbf{g}}(t)$ in (A10) will be real even if p_1 and p_2 are complex. An alternative approach for complex p_1 and p_2 would be to do the partial expansion shown in (A6) differently. (This approach is discussed below.)

$$x_1(0) = x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$$

$$\dot{x}_1(t) = -p_1 x_1(t) + \Delta\omega(t)$$
$$\dot{x}_2(t) = -p_2 x_2(t) + \Delta\omega(t)$$
$$\dot{x}_3(t) = -p_3 x_3(t) + \Delta\omega(t)$$
$$\dot{x}_4(t) = -p_4 x_4(t) + \Delta\omega(t)$$
$$\dot{x}_5(t) = -p_5 x_5(t) + \Delta\omega(t)$$

$$\gamma_g(t) = c_1 x_1(t) + c_2 x_2(t) + c_3 x_3(t) + c_4 x_4(t) + c_5 x_5(t)$$

where $K' = \frac{K T_3 T_4}{T_1 T_2 T_5 T_6}$, $p_1 = \frac{1}{2T_2}\left(1 + \sqrt{1 - \frac{4T_2}{T_1}}\right)$, $p_2 = \frac{1}{2T_2}\left(1 - \sqrt{1 - \frac{4T_2}{T_1}}\right)$

$p_3 = \frac{1}{T_5}$, $p_4 = \frac{1}{T_6}$, $p_5 = 0$, $z_1 = \frac{1}{T_3}$, and $z_2 = \frac{1}{T_4}$

$$c_1 = \frac{-K'(z_1 - p_1)(z_2 - p_1)}{(p_2 - p_1)(p_3 - p_1)(p_4 - p_1)p_1} \qquad c_2 = \frac{-K'(z_1 - p_2)(z_2 - p_2)}{(p_1 - p_2)(p_3 - p_2)(p_4 - p_2)p_2}$$

$$c_3 = \frac{-K'(z_1 - p_3)(z_2 - p_3)}{(p_1 - p_3)(p_2 - p_3)(p_4 - p_3)p_3} \qquad c_4 = \frac{-K'(z_1 - p_4)(z_2 - p_4)}{(p_1 - p_4)(p_2 - p_4)(p_3 - p_4)p_4} \qquad c_5 = \frac{K' z_1 z_2}{p_1 p_2 p_3 p_4} = K$$

Figure 5: First Set of Differential Equations for Diesel Engine Governor. Complex Arithmetic Version.

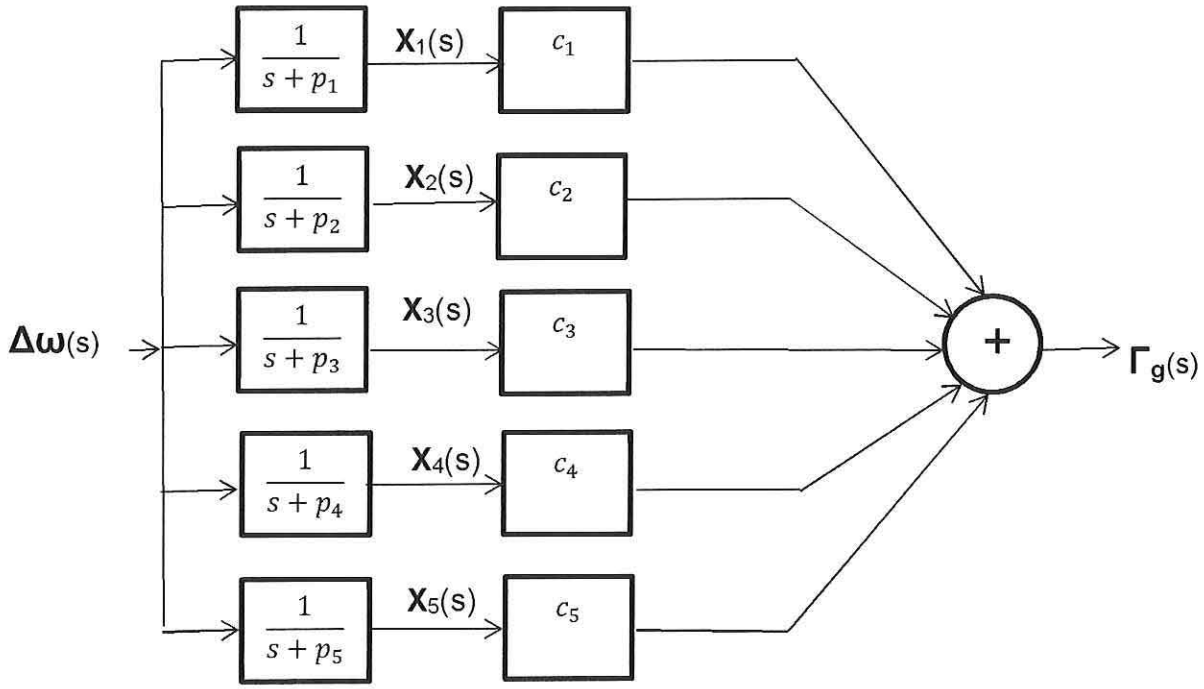

Figure 6: Decoupled Form of Diesel Engine Governor Transfer Function

The next step in the simulation process is to convert the differential equations shown in Figure 5 into corresponding difference equations. While there are several ways of doing this, the sample and hold approach used in our past work will be used here. For the differential equation in (A8), we have:

$$x_1[(k+1)h_1] = \emptyset_1 x_1(kh_1) + \gamma_1 \cdot \Delta\omega(kh_1) \tag{A11}$$

where $$\emptyset_1 = e^{-p_1 h_1} \text{ and } \gamma_1 = \frac{1-e^{-p_1 h_1}}{p_1} \tag{A12}$$

If we assume the integration step h_1 is small, equation (A11) reduces to the forward Euler integration approximation:

$$x_1[(k+1)h_1] = (1-p_1 h_1)x_1(kh_1) + h_1 \cdot \Delta\omega(kh_1) \tag{A13}$$

Under the assumption that the input $\Delta\omega(kh_1)$ is a sample and hold signal (with step size h_1) then (A11) is an exact solution at time $t = kh_1$ to the differential equation in (A8). The same reasoning can be applied to converting the differential equations for $\dot{x}_2(t)$, $\dot{x}_3(t)$, and $\dot{x}_4(t)$ in (A9) to difference equations of the form similar to that shown in (A11). A starting point for the difference equation that goes along with the $\dot{x}_5(t)$ differential equation is shown in (A14):

$$x_5[(k+1)h_5] = \emptyset_5 x_5(kh_5) + \gamma_5 \cdot \Delta\omega(kh_5) \tag{A14}$$

where $$\emptyset_5 = e^{-p_5 h_5} \text{ and } \gamma_5 = \frac{1-e^{-p_5 h_5}}{p_5}, p_5 = 0. \tag{A15}$$

To avoid the division by 0 in γ_5, we will set $\gamma_5 = \lim_{p_5 \to 0} \frac{1-e^{-p_5 h_5}}{p_5} = h_5$. The difference equation in (A14) then reduces to:

$$x_5[(k+1)h_5] = x_5(kh_5) + h_5 \cdot \Delta\omega(kh_5) \tag{A16}$$

For the case of no multi-rate (common step size h), a set of difference equations that goes along with the differential equations in (A9) is shown in Figure 6 below.

Given constants: $T_1, T_2, T_3, T_4, T_5, T_6, K$, step sizes where $h_1 = h_2 = h_3 = h_4 = h_5 \stackrel{\text{def}}{=} h$ for single step size integration.

Initial conditions $x_1(0) = x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$.

Calculate constants: $K' = \frac{KT_3T_4}{T_1T_2T_5T_6}$, $p_3 = \frac{1}{T_5}$, $p_4 = \frac{1}{T_6}$, $z_1 = \frac{1}{T_3}$, $z_2 = \frac{1}{T_4}$

$$p_1 = \frac{1}{2T_2}\left(1 + \sqrt{1 - \frac{4T_2}{T_1}}\right), \quad p_2 = \frac{1}{2T_2}\left(1 - \sqrt{1 - \frac{4T_2}{T_1}}\right)$$

$$c_1 = \frac{-K'(z_1 - p_1)(z_2 - p_1)}{(p_2 - p_1)(p_3 - p_1)(p_4 - p_1)p_1} \quad c_2 = \frac{-K'(z_1 - p_2)(z_2 - p_2)}{(p_1 - p_2)(p_3 - p_2)(p_4 - p_2)p_2}$$

$$c_3 = \frac{-K'(z_1 - p_3)(z_2 - p_3)}{(p_1 - p_3)(p_2 - p_3)(p_4 - p_3)p_3} \quad c_4 = \frac{-K'(z_1 - p_4)(z_2 - p_4)}{(p_1 - p_4)(p_2 - p_4)(p_3 - p_4)p_4} \quad c_5 = K$$

$$\emptyset_i = e^{-p_i h_i} \text{ and } \gamma_i = \frac{1 - e^{-p_i h_i}}{p_i}, \; i = 1,2,3,4$$

Iterate on the following six equations (k = 0, 1, 2, …):

$$x_1[(k+1)h_1] = \emptyset_1 x_1(kh_1) + \gamma_1 \cdot \Delta\omega(kh_1)$$
$$x_2[(k+1)h_2] = \emptyset_2 x_2(kh_2) + \gamma_2 \cdot \Delta\omega(kh_2)$$
$$x_3[(k+1)h_3] = \emptyset_3 x_3(kh_3) + \gamma_3 \cdot \Delta\omega(kh_3)$$
$$x_4[(k+1)h_4] = \emptyset_4 x_4(kh_4) + \gamma_4 \cdot \Delta\omega(kh_4)$$
$$x_5[(k+1)h_5] = x_5(kh_5) + h_5 \cdot \Delta\omega(kh_5)$$
$$\gamma_g(kh) = c_1 x_1(kh_1) + c_2 x_2(kh_2) + c_3 x_3(kh_3) + c_4 x_4(kh_4) + c_5 x_5(kh_5)$$

Figure 7: First Set of Difference Equations for Diesel Engine Governor. Complex Arithmetic Version.

Complex Root Case

The results in Figure 5 and Figure 7 provide simulation differential and difference equations for the transfer function shown in (A1). The quadratic term in the denominator of (A1) has complex roots p_1 and p_2 when $T_1 < 4T_2$. The differential equations in Figure 5 and the difference equations in Figure 7 can still be used for simulation but complex arithmetic must be used. In this section, an alternative set of differential and difference equations will be developed. These will not require the use of complex numbers for run-time calculations, and they can still be used whether or not $T_1 < 4T_2$. The starting point for finding the differential equations for the quadratic root case is to slightly modify (A6) taking into account that:

$$p_1 = \frac{1}{2T_2}\left(1 + j\sqrt{\frac{4T_2}{T_1} - 1}\right) \quad p_2 = p_1^* = \frac{1}{2T_2}\left(1 - j\sqrt{\frac{4T_2}{T_1} - 1}\right) \tag{A17}$$

The result is shown in (A18):

$$\frac{\Gamma_g(s)}{\Delta\omega(s)} = \frac{c_1}{s+p_1} + \frac{c_2}{s+p_1^*} + \frac{c_3}{s+p_3} + \frac{c_4}{s+p_4} + \frac{c_5}{s+p_5} \tag{A18}$$

It can be shown that $c_1^* = c_2$. (Figure 6 is changed to Figure 8 for the complex roots case considered here.)

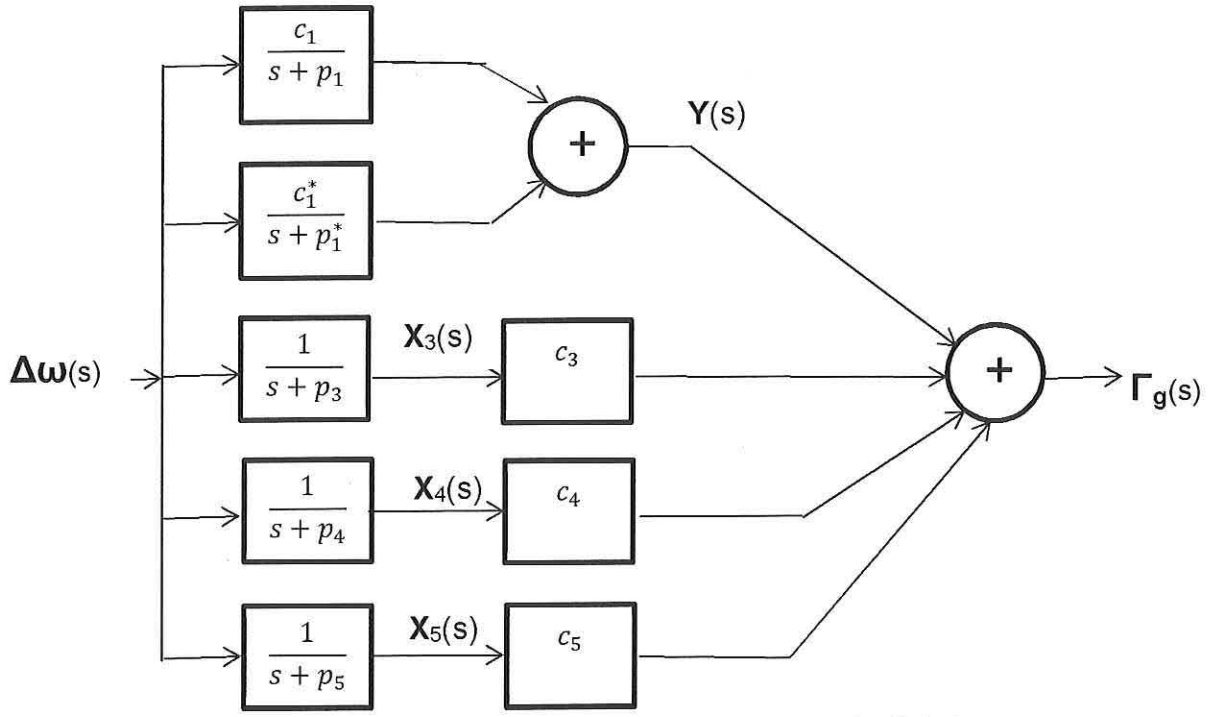

Figure 8: Decoupled Form of Diesel Engine Governor Transfer Function for Complex Pole Case

Equation (A18) can then be written as (A19):

$$\frac{\Gamma_g(s)}{\Delta\omega(s)} = \frac{(c_1+c_1^*)s+c_1p_1^*+c_1^*p_1}{s^2+(p_1+p_1^*)s+p_1p_1^*} + \frac{c_3}{s+p_3} + \frac{c_4}{s+p_4} + \frac{c_5}{s+p_5} \quad \text{(A19)}$$

The first term on the right side of (A19) can be written in a more standard notation as shown below.

$$\frac{\Gamma_g(s)}{\Delta\omega(s)} = \frac{as+b}{s^2+2\xi\omega_n s+\omega_n^2} + \frac{c_3}{s+p_3} + \frac{c_4}{s+p_4} + \frac{c_5}{s+p_5} \quad \text{(A20)}$$

where $$2\xi\omega_n = p_1 + p_1^* = \frac{1}{T_2} \text{ and } \omega_n^2 = \frac{1}{p_1p_1^*} = \frac{1}{T_1T_2} \quad \text{(A21)}$$

Parameters c_3, c_4, c_5 in (A20), which are real numbers, could be calculated the same way as before (see (A7)). However, c_3 and c_4 contain complex numbers p_1 and p_2. To avoid using complex arithmetic, we can substitute the complex expressions for p_1 and p_2 into the expressions for c_3 and c_4. After simplifying the expressions, we obtain the alternative all-real results for c_3 and c_4 shown in (A22) below:

$$c_3 = \frac{KT_5(T_5-T_3)(T_5-T_4)}{(T_5^2-T_1T_5+T_1T_2)(T_6-T_5)} \quad c_4 = \frac{KT_6(T_6-T_3)(T_6-T_4)}{(T_6^2-T_1T_6+T_1T_2)(T_5-T_6)} \quad \text{(A22)}$$

Expressions for constants a and b in (A20) will now be found.

Two methods for finding constant a in (A20)

Method 1: Comparing (A19) with (A20), we see that

$$a = c_1 + c_1^* \quad \text{(A23)}$$

Since $a^* = a$, it follows that a is real. However, c_1 is a complex number. If we have access to software like Matlab that can evaluate complex numbers, we can directly calculate (A23).

Method 2: We can find a without working directly with complex number c_1. To do this, first take $\lim_{s\to\infty}\left(\frac{s\Gamma_g(s)}{\Delta\omega(s)}\right)$ on the right side of (A20). Then do the same thing with (A1). Then set the two expressions equal and solve for a. The result is in (A24) below:

$$a = -(c_3 + c_4 + c_5) \tag{A24}$$

Since c_3, c_4, and $c_5 = K$ are all real (see (A22)), no complex number capability is needed to find a.

Two methods for finding constant b in (A20)

Method 1: Comparing (A19) with (A20), we see that

$$b = c_1 p_1^* + c_1^* p_1 \tag{A25}$$

Since $b^* = b$, it follows that *b* is real. All of the terms on the right side of (A25) are complex. If we have a computer with complex number manipulation capability, we can calculate *b* using (A25).

Method 2. If we do not want to work with complex numbers such as those in (A25), we can find *b* the following way. Set s = 1 on the right side of (A20). Also, set s = 1 in the transfer function shown in (A1). Equate these two expressions for $\frac{\Gamma_g(s)}{\Delta\omega(s)}$ and then solve for *b*:

$$b = \frac{K\prime(1+z_1)(1+z_2)}{(1+p_3)(1+p_4)} - (1 + \frac{1}{T_2} + \frac{1}{T_1 T_2})\left(\frac{c_3}{1+p_3} + \frac{c_4}{1+p_4} + c_5\right) - a \tag{A26}$$

All the terms on the right side of (A26) are real so no complex arithmetic capability is needed to find *b*.

Differential Equations for the Diesel Engine Governor-Complex Root Case

The next step is to find a set of two differential equations associated with the term $\frac{Y(s)}{\Delta\omega(s)}$ that appears in (A20) above.

$$\frac{Y(s)}{\Delta\omega(s)} \stackrel{\text{def}}{=} \frac{as+b}{s^2+2\xi\omega_n s+\omega_n^2} \tag{A27}$$

A useful choice is shown in (A28):

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_n^2 & -2\xi\omega_n\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\Delta\omega(t) \tag{A28}$$

$$y(t) = bx_1(t) + ax_2(t) \tag{A29}$$

As a check, if we Laplace transform (A28) and (A29), and then combine the results we obtain (A27).

Thinking ahead to forming a set of difference equations for the diesel motor governor, we will need the transition matrix $\mathbf{\Phi}(t)$ that goes along with (A28). This can be found using (A30):

$$\mathbf{\Phi}(t) = \mathcal{L}^{-1}\left\{\left[\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\xi\omega_n \end{bmatrix}\right]^{-1}\right\} = \begin{bmatrix} \emptyset_{11}(t) & \emptyset_{12}(t) \\ \emptyset_{21}(t) & \emptyset_{22}(t) \end{bmatrix} \tag{A30}$$

where

$$\emptyset_{11}(t) = \frac{e^{-\xi\omega_n t}}{\omega_d}[\xi\omega_n \sin(\omega_d t) + \omega_d \cos(\omega_d t)], \qquad \omega_d = \omega_n\sqrt{1-\xi^2} \tag{A31}$$

$$\emptyset_{12}(t) = \frac{e^{-\xi\omega_n t}}{\omega_d}\sin(\omega_d t) \tag{A32}$$

$$\emptyset_{21}(t) = -\frac{\omega_n^2 e^{-\xi\omega_n t}}{\omega_d}\sin(\omega_d t) \tag{A33}$$

$$\emptyset_{22}(t) = \frac{e^{-\xi\omega_n t}}{\omega_d}[-\xi\omega_n \sin(\omega_d t) + \omega_d \cos(\omega_d t)] \tag{A34}$$

In addition to the two differential equations in (A28), we will need three differential equations associated with the terms $X_3(s)$, $X_4(s)$, and $X_5(s)$ shown in Figure 8. However, these terms have real poles, so we can reuse the earlier results shown in (A9) which are repeated below:

$$\dot{x}_3(t) = -p_3 x_3(t) + \Delta\omega(t) \tag{A35}$$

$$\dot{x}_4(t) = -p_4 x_4(t) + \Delta\omega(t) \tag{A36}$$

$$\dot{x}_5(t) = -p_5 x_5(t) + \Delta\omega(t) \tag{A37}$$

where the (real) p_3, p_4, p_5 are defined in (A4). From Figure 8, the desired output $\gamma_g(t)$ of the diesel engine governor is shown below:

$$\gamma_g(t) = y(t) + c_3 x_3(t) + c_4 x_4(t) + c_5 x_5(t) \tag{A38}$$

Lastly, if we substitute (A29) into (A38), we have:

$$\gamma_g(t) = b x_1(t) + a x_2(t) + c_3 x_3(t) + c_4 x_4(t) + c_5 x_5(t) \tag{A39}$$

In matrix form, the state equations can be written as shown in (A40), (A41) below:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_5(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\omega_n^2 & -2\xi\omega_n & 0 & 0 & 0 \\ 0 & 0 & -p_3 & 0 & 0 \\ 0 & 0 & 0 & -p_4 & 0 \\ 0 & 0 & 0 & 0 & -p_5 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}\Delta\omega(t) \tag{A40}$$

$$\gamma_{\mathbf{g}}(t) = [b \;\; a \;\; c_3 \;\; c_4 \;\; c_5]\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} \tag{A41}$$

From (A22) the expressions for c_3, c_4, c_5 = K that appear in (A41) are real numbers. Hence, (A40) and (A41) are a set of real equations.

Figure 9 summarizes the differential equations for the Diesel Engine Governor for the complex roots case ($T_1 < 4T_2$.). These equations can also be used when $T_1 > 4T_2$..

$$x_1(0) = x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0$$

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -\omega_n^2 x_1(t) - 2\xi\omega_n x_2(t) + \Delta\omega(t)$$

$$\dot{x}_3(t) = -p_3 x_3(t) + \Delta\omega(t)$$

$$\dot{x}_4(t) = -p_4 x_4(t) + \Delta\omega(t)$$

$$\dot{x}_5(t) = -p_5 x_5(t) + \Delta\omega(t)$$

$$\gamma_g(t) = b x_1(t) + a x_2(t) + c_3 x_3(t) + c_4 x_4(t) + c_5 x_5(t)$$

where $p_3 = \frac{1}{T_5}$, $p_4 = \frac{1}{T_6}$, $p_5 = 0$, $z_1 = \frac{1}{T_3}$, $z_2 = \frac{1}{T_4}$, $\omega_n^2 = \frac{1}{T_1 T_2}$, $2\xi\omega_n = \frac{1}{T_2}$

$$c_3 = \frac{KT_5(T_5-T_3)(T_5-T_4)}{(T_5^2 - T_1 T_5 + T_1 T_2)(T_6-T_5)} \quad c_4 = \frac{KT_6(T_6-T_3)(T_6-T_4)}{(T_6^2 - T_1 T_6 + T_1 T_2)(T_5-T_6)} \quad c_5 = K, \quad K' = \frac{KT_3T_4}{T_1T_2T_5T_6}$$

$$a = -(c_3 + c_4 + c_5) \quad b = \frac{K'(1+z_1)(1+z_2)}{(1+p_3)(1+p_4)} - (1 + \frac{1}{T_2} + \frac{1}{T_1T_2})\left(\frac{c_3}{1+p_3} + \frac{c_4}{1+p_4} + c_5\right) - a$$

Figure 8 Second set of differential equations for linear part of the diesel motor regulator as shown in Figure 2a or Figure 2b. These equations use real arithmetic

Figure 9: Second Set of Differential Equations for Diesel Engine Governor. Real Arithmetic Version.

Another Set of Differential Equations for the Diesel Engine Governor

For linear systems such as the one shown in (A1), the "control canonical form" is an easy way to obtain a set of differential equations. By expanding the terms in the numerator and the denominator, the transfer function in (A1) can be written in the standard form shown in (A43):

$$\frac{\boldsymbol{\Gamma_g}(s)}{\boldsymbol{\Delta\omega}(s)} = \frac{b_3 s^2 + b_4 s + b_5}{s^5 + a_1 s^4 + a_2 s^3 + a_3 s^2 + a_4 s + a_5} \tag{A43}$$

where $b_3 = K'$

$$b_4 = K'\left(\frac{1}{T_3} + \frac{1}{T_4}\right)$$

$$b_5 = K'\left(\frac{1}{T_3T_4}\right)$$

$$a_1 = \frac{1}{T_2} + \frac{1}{T_5} + \frac{1}{T_6}$$

$$a_2 = \frac{1}{T_1T_2} + \frac{1}{T_2T_5} + \frac{1}{T_2T_6} + \frac{1}{T_5T_6}$$

$$a_3 = \frac{1}{T_1T_2T_5} + \frac{1}{T_1T_2T_6} + \frac{1}{T_2T_5T_6}$$

$$a_4 = \frac{1}{T_1T_2T_5T_6}$$

$$a_5 = 0$$

Equation (A43) could be converted to the differential equation shown in (A44):

$$\gamma_g^{[5]}(t) + a_1\gamma_g^{[4]}(t) + a_2\gamma_g^{[3]}(t) + a_3\ddot{\gamma}_g(t) + a_4\dot{\gamma}_g(t) + a_5\gamma_g(t) = b_3\Delta\ddot{\omega}(t) + b_4\Delta\dot{\omega}(t) + b_5\Delta\omega(t) \tag{A44}$$

However, we would like to solve an equivalent set of differential equations that do not require us to calculate the time derivatives of $\Delta\omega(t)$ shown on the right side of (A44). To do this, define state variables $x_1,\ x_2,\ x_3,\ x_4,\ x_5$ in the following way:

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t)\\
\dot{x}_2(t) &= x_3(t)\\
\dot{x}_3(t) &= x_4(t)\\
\dot{x}_4(t) &= x_5(t)\\
\dot{x}_5(t) &= -a_5x_1(t) - a_4x_2(t) - a_3x_3(t) - a_2x_4(t) - a_1x_5(t) + \Delta\omega(t)
\end{aligned} \tag{A45}$$

Take the Laplace transform of both sides of the equations in (A45). This gives us the following:

$$\begin{aligned}
X_2(s) &= sX_1(s)\\
X_3(s) &= sX_2(s) = s^2X_1(s)\\
X_4(s) &= sX_3(s) = s^3X_1(s)\\
X_5(s) &= sX_4(s) = s^4X_1(s)
\end{aligned} \tag{A46}$$

and

$$sX_5(s) = -a_5X_1(s) - a_4X_2(s) - a_3X_3(s) - a_2\,X_4(s) - a_1X_5(s) + \boldsymbol{\Delta\omega(s)} \tag{A47}$$

Next, combine (A46) and (A47) to give (A48) below:

$$(\,s^5 + a_1s^4 + a_2\,s^3 + \,a_3\,s^2\, + a_4s + a_5\,)X_1(s) = \boldsymbol{\Delta\omega(s)} \tag{A48}$$

Multiply the numerator and the denominator on the right side of (A43) by $X_1(s)$. This gives us (A49).

$$\frac{\boldsymbol{\Gamma_g}(s)}{\boldsymbol{\Delta\omega}(s)} = \frac{(b_3\,s^2 + b_4\,s^1 + b_5)X_1(s)}{(s^5 + a_1\,s^4 + a_2\,s^3 + a_3\,s^2 + a_4s + a_5)X_1(s)} \tag{A49}$$

Lastly, we substitute (A48) into (A49) to give us (A50):

$$\boldsymbol{\Gamma_g}(s) = \,(b_3\,s^2 + b_4\,s^1 + b_5)X_1(s) = b_3X_3(s) + b_4X_2(s) + b_5X_1(s) \tag{A50}$$

Inverting back to the time domain, we have our desired torque output:

$$\gamma_g(t) = b_5x_1(t) + b_4x_2(t) + b_3x_3(t) \tag{A51}$$

To summarize, our objective is to solve a set of differential equations that our consistent with the transfer function shown in Figure 2b. We can do this by solving the five differential equations shown in (A45). We determine the desired generated motor torque $\gamma_g(t)$ by using (A51).

The first advantage of using the set of equations in (A45) and (A51) is that they are not complicated. Second, this set of equations can be used with no modifications for either real or complex roots to the quadratic polynomial in the denominator of the transfer function in Figure 2b. Third, the set of equations in (A45) and (A51) can be used with no modifications when repeated roots occur in the denominator of the transfer function in (A1). These advantages disappear if we try to solve the equations by decoupling them and then using multi-rate. To uncouple the equations we can use a similarity transformation. If the poles of the transfer function in (A1) are repeated, we cannot in general create a diagonal system matrix by using a similarity transformation. (For certain special cases, a diagonal matrix can be found; there is a test to see if it can be done). Also, if the quadratic term in (A1) has complex roots, then the diagonal matrix used in the similarity transformation will have complex elements requiring the use of complex arithmetic.

This approach for converting a transfer function into a set of differential equations was used to convert the transfer function in (A27) into the differential equations shown in (A28).

Difference Equations for the complex root case.

The next step is to obtain a set of multi-rate capable difference equations that go along with the differential equations in Figure 9. We see from (A40) that the system matrix is not diagonal. If we use a similarity transformation to make this matrix diagonal, we will end up with complex numbers along the diagonal. However, we can view the system matrix as block diagonal with a second order system (states $x_1(t)$ and $x_2(t)$) occupying the upper left block. The remaining blocks along the diagonal are one by one blocks. All the blocks are uncoupled. The difference equations for the second order block will be found first by starting with the differential equations in (A28). These differential equations can be written as:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\,\Delta\omega(t) \tag{A52}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\xi\omega_n \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{A53}$$

The sample and hold discrete version of (A52) is shown in (A54) below:

$$\mathbf{x}[(k+1)\mathrm{T}] = \mathbf{\Phi}(\mathrm{T})\mathbf{x}(\mathrm{kT}) + \mathbf{\Gamma}(\mathrm{T})\,\boldsymbol{\Delta\omega}(\mathrm{kT}) \tag{A54}$$

where T = hold time. The transition matrix $\mathbf{\Phi}(t)$ for **A** in (A53) has already been found (See (30)). Also, $\mathbf{\Gamma}(\mathrm{T}) = \int_0^T \mathbf{\Phi}(\rho)d\rho\,\mathbf{B}$. For numerical integration purposes, it will be assumed that the integration step size is h=T. Then $\mathbf{\Phi}(\mathrm{T}) = \mathbf{\Phi}(\mathrm{h})$. The expression for $\mathbf{\Gamma}(\mathrm{T})$ can be simplified by noting that the determinant of matrix **A** is $\omega_n^2 = 1/(T_1T_2)$ which is not zero so $\mathbf{A}^{-1}$ exists. Performing the integration, we have:

$$\mathbf{\Gamma}(\mathrm{T})=\mathbf{\Gamma}(\mathrm{h})= \int_0^h \mathbf{\Phi}(\rho)d\rho\,\mathbf{B} = (\,\mathbf{\Phi}(\mathrm{h}) - \mathbf{I})\mathbf{A}^{-1}\mathbf{B} \tag{A55}$$

$$= \left[\begin{bmatrix} \emptyset_{11}(h) & \emptyset_{12}(h) \\ \emptyset_{21}(h) & \emptyset_{22}(h) \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right]\begin{bmatrix} -2\xi\omega_n^{-1} & -\omega_n^{-2} \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

or

$$\mathbf{\Gamma}(\mathrm{T}) = \mathbf{\Gamma}(\mathrm{h}) = \frac{1}{\omega_n^2}\begin{bmatrix} 1 - \emptyset_{11}(h) \\ -\emptyset_{21}(h) \end{bmatrix} \tag{A56}$$

where $\emptyset_{11}(h)$ is given by (A31) and $\emptyset_{21}(h)$ is given by (A33) with t replaced by h.

From (A30), the transition matrix $\mathbf{\Phi}(\mathrm{h})$ is:

$$\mathbf{\Phi}(\mathrm{h}) = \begin{bmatrix} \emptyset_{11}(h) & \emptyset_{12}(h) \\ \emptyset_{21}(h) & \emptyset_{22}(h) \end{bmatrix} \tag{A57}$$

The elements of $\mathbf{\Phi}(\mathrm{h})$ are shown in (A31)-(A34) after setting t = h. Combining the above results, we have the following for the first two difference equations:

$$x_1[(k+1)h] = \emptyset_{11}(h)x_1(\mathrm{kh}) + \emptyset_{12}(h)x_2(\mathrm{kh}) + \frac{[\,1-\emptyset_{11}(h)]}{\omega_n^2}\Delta\omega(kh) \tag{A58}$$

$$x_2[(k+1)h] = \emptyset_{21}(h)x_1(\mathrm{kh}) + \emptyset_{22}(h)x_2(\mathrm{kh}) - \frac{\emptyset_{21}(h)}{\omega_n^2}\Delta\omega(kh) \tag{A59}$$

The remaining three differential equations in (A40) can be converted to difference equations using the same procedure that was used to convert the last three differential equations shown in (A9). The results are shown below:

$$x_3[(k+1)h] = \emptyset_3 x_3(kh) + \gamma_3 \cdot \Delta\omega(kh) \tag{A60}$$

$$x_4[(k+1)h] = \emptyset_4 x_4(kh) + \gamma_4 \cdot \Delta\omega(kh) \tag{A61}$$

$$x_5[(k+1)h] = x_5(kh) + h \cdot \Delta\omega(kh) \tag{A62}$$

where:

$$\emptyset_k = e^{-p_k h} \quad \text{and} \quad \gamma_k = \frac{1-e^{-p_k h}}{p_k}, \ \mathrm{k} = 3,4 \tag{A63}$$

$$\mathrm{p}_3 = \frac{1}{T_5}, \ \mathrm{p}_4 = \frac{1}{T_6}$$

The desired output is:

$$\gamma_g(kh) = bx_1(kh) + ax_2(kh) + c_3x_3(kh) + c_4x_4(kh) + c_5x_5(kh) \tag{A64}$$

where *a* is given by (A24) and *b* is given by (A26).

The matrix version of the state difference equations is shown in (A65) and (A66) below:

$$\begin{bmatrix} x_1[(k+1)h] \\ x_2[(k+1)h] \\ x_3[(k+1)h] \\ x_4[(k+1)h] \\ x_5[(k+1)h] \end{bmatrix} = \begin{bmatrix} \emptyset_{11}(h) & \emptyset_{12}(h) & 0 & 0 & 0 \\ \emptyset_{21}(h) & \emptyset_{22}(h) & 0 & 0 & 0 \\ 0 & 0 & \emptyset_3 & 0 & 0 \\ 0 & 0 & 0 & \emptyset_4 & 0 \\ 0 & 0 & 0 & 0 & \emptyset_5 \end{bmatrix} \begin{bmatrix} x_1(kh) \\ x_2(kh) \\ x_3(kh) \\ x_4(kh) \\ x_5(kh) \end{bmatrix} + \begin{bmatrix} \frac{1-\emptyset_{11}(h)}{\omega_n^2} \\ \frac{-\emptyset_{21}(h)}{\omega_n^2} \\ \gamma_3 \\ \gamma_4 \\ h \end{bmatrix} \Delta\omega(kh) \tag{A65}$$

$$\gamma_g(kh) = \begin{bmatrix} b & a & c_3 & c_4 & c_5 \end{bmatrix} \begin{bmatrix} x_1(kh) \\ x_2(kh) \\ x_3(kh) \\ x_4(kh) \\ x_5(kh) \end{bmatrix} \tag{A66}$$

Given constants: $T_1, T_2, T_3, T_4, T_5, T_6, K$, step size h

Initial conditions $x_1(0) = x_2(0) = x_3(0) = x_4(0) = x_5(0) = 0.$

Calculate constants: $K' = \frac{KT_3T_4}{T_1T_2T_5T_6}$, $p_3 = \frac{1}{T_5}$, $p_4 = \frac{1}{T_6}$, $z_1 = \frac{1}{T_3}$, $z_2 = \frac{1}{T_4}$, $\omega_n = \frac{1}{\sqrt{T_1T_2}}$, $\xi = \frac{1}{2\omega_n T_2}$

$$c_3 = \frac{KT_5(T_5-T_3)(T_5-T_4)}{(T_5^2-T_1T_5+T_1T_2)(T_6-T_5)} \quad c_4 = \frac{KT_6(T_6-T_3)(T_6-T_4)}{(T_6^2-T_1T_6+T_1T_2)(T_5-T_6)} \quad c_5 = K, \quad \omega_d = \omega_n\sqrt{1-\xi^2}$$

$$a = -(c_3 + c_4 + c_5)$$

$$b = \frac{K'(1+z_1)(1+z_2)}{(1+p_3)(1+p_4)} - (1+\frac{1}{T_2}+\frac{1}{T_1T_2})\left(\frac{c_3}{1+p_3}+\frac{c_4}{1+p_4}+c_5\right) - a$$

$$\emptyset_{11}(h) = \frac{e^{-\xi\omega_n h}}{\omega_d}[\xi\omega_n \sin(\omega_d h) + \omega_d \cos(\omega_d h)]$$

$$\emptyset_{12}(h) = \frac{e^{-\xi\omega_n h}}{\omega_d}\sin(\omega_d h)$$

$$\emptyset_{21}(h) = -\frac{\omega_n^2 e^{-\xi\omega_n h}}{\omega_d}\sin(\omega_d h)$$

$$\emptyset_{22}(h) = \frac{e^{-\xi\omega_n h}}{\omega_d}[-\xi\omega_n \sin(\omega_d h) + \omega_d \cos(\omega_d h)]$$

$$\emptyset_3 = e^{-p_3 h}, \quad \emptyset_4 = e^{-p_4 h}, \quad \gamma_3 = \frac{1-e^{-hp_3}}{p_3}, \quad \gamma_4 = \frac{1-e^{-hp_4}}{p_4}$$

Iterate on the following six equations (k = 0, 1, 2, ...):

$$x_1[(k+1)h] = \emptyset_{11}(h)x_1(\text{kh}) + \emptyset_{12}(h)x_2(\text{kh}) + \frac{[1-\emptyset_{11}(h)]}{\omega_n^2}\Delta\omega(kh)$$

$$x_2[(k+1)h] = \emptyset_{21}(h)x_1(\text{kh}) + \emptyset_{22}(h)x_2(\text{kh}) - \frac{\emptyset_{21}(h)}{\omega_n^2}\Delta\omega(kh)$$

$$x_3[(k+1)h] = \emptyset_3 x_3(kh) + \gamma_3 \cdot \Delta\omega(kh)$$

$$x_4[(k+1)h] = \emptyset_4 x_4(kh) + \gamma_4 \cdot \Delta\omega(kh)$$

$$x_5[(k+1)h] = x_5(kh) + h \cdot \Delta\omega(kh)$$

$$\gamma_g(kh) = bx_1(kh) + ax_2(kh) + c_3x_3(kh) + c_4x_4(kh) + c_5x_5(kh)$$

Figure 10: Second Set of Difference Equations for Diesel Engine Governor. Real Arithmetic Version.

Simulation Results

In this section, simulation results using the above integration methods are compared with those generated by using the equivalent Simulink\SimPowerSystems simulation. Both the case when only real arithmetic is used and the case when complex arithmetic is used are considered. The main purpose for generating the simulation results presented here is to verify the differential and difference equations shown in Figure 5, Figure 7, Figure 9, and Figure 10. Another reason was to see if the complex arithmetic approach executes more quickly than the real arithmetic approach. All the simulation equations used here do not include the saturation limiter or the output delay shown in Figure 4. In order to effectively disable the torque saturation limiter in the SimPowerSystems simulation, the limits used were increased to ± 100 pu. The output delay was set to 0. Referring to Figure 4, the reference input was set to w_{ref} = 1 pu, and the speed input was set to w = 0.8 pu. Thus the differential input $\Delta\omega(t) \stackrel{def}{=} w_{ref} - w$ is a step input of amplitude 0.2 pu.

Case 1: Complex Roots Example. Referring to the quadratic polynomial in (A1), this means that we are assuming $T_1 < 4T_2$. The simulation parameters used here are provided in Table 6.

Table 6: Simulation Parameter for Complex Root Example

Parameter	Value
$\mathbf{T_1}$	**0.01** sec
T_2	0.02 sec
T_3	0.20 sec
T_4	0.25 sec
T_5	0.009 sec
T_6	0.0384 sec
K	40

The Diesel Engine Governor system in (A1) was simulated first using the difference equations shown in Figure 7 (integration step size h = 0.01 sec). These equations require the use of complex arithmetic when $T_1 < 4T_2$, but are of a simpler form than those that use only real arithmetic. The use of complex arithmetic would probably make a FPGA simulation more complicated, but perhaps it would run faster once implemented. This was checked here using a Matlab coded simulation that has complex arithmetic capability. Figure 11 shows the predicted mechanical power output. To check the result in Figure 11, the simulation was repeated using the difference equations shown in Figure 10. These equations only use real arithmetic. The plot of mechanical power using these equations is shown in Figure 12 and shows good agreement with the plot in Figure 11. The results in Figure 11 and Figure 12 were also checked using the control canonical form shown in equation (A45). These differential equations were solved using the Matlab built-in variable step size differential equation solver called ode23tb. From past experience, ode23tb is the most reliable Matlab variable step size solver. The plot of mechanical power shown in Figure 13 is in good agreement with the plots in Figure 11 and Figure 12. While useful for checking purposes, the set of differential equations in (A45) is not in a form that can be directly used for multi-rate simulation. The last simulation technique considered here is to use the SimPowerSystems simulation to calculate the mechanical power. The simulation result is shown in Figure 14 and shows good agreement with results using the other methods mentioned above; see Figure 15 and Figure 16. The plotting points for the "real arithmetic" and the "complex arithmetic" coincide. These data points represent exact solution points, not just approximations, to the original set of differential equations for the diesel engine governor.

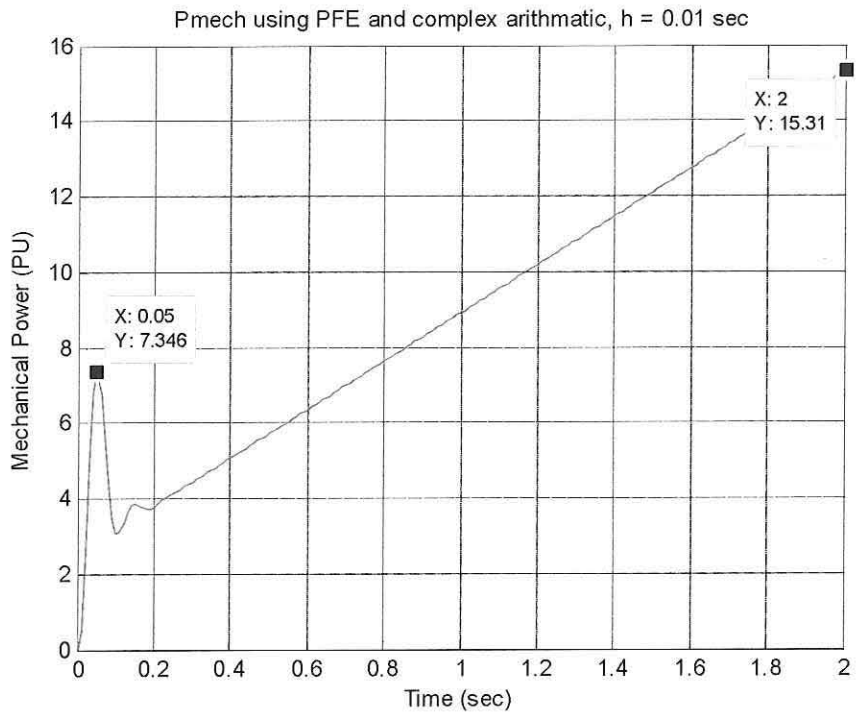

*Figure 11: Mechanical Power using complex arithmetic simulation (see Figure 7, T1 < 4*T2)*

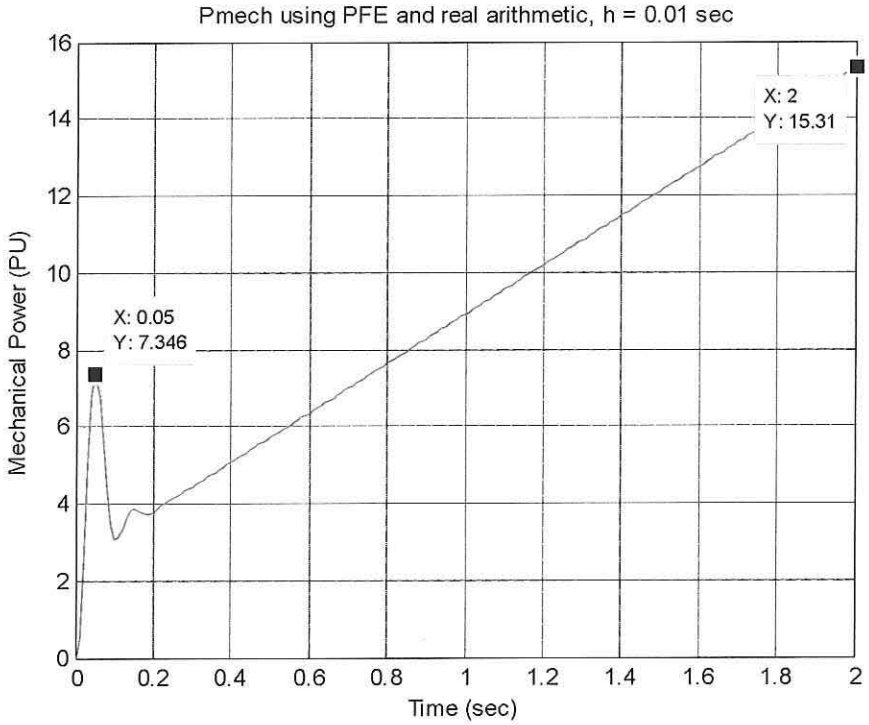

*Figure 12: Mechanical Power using real arithmetic simulation (see Figure 10, T1 < 4*T2)*

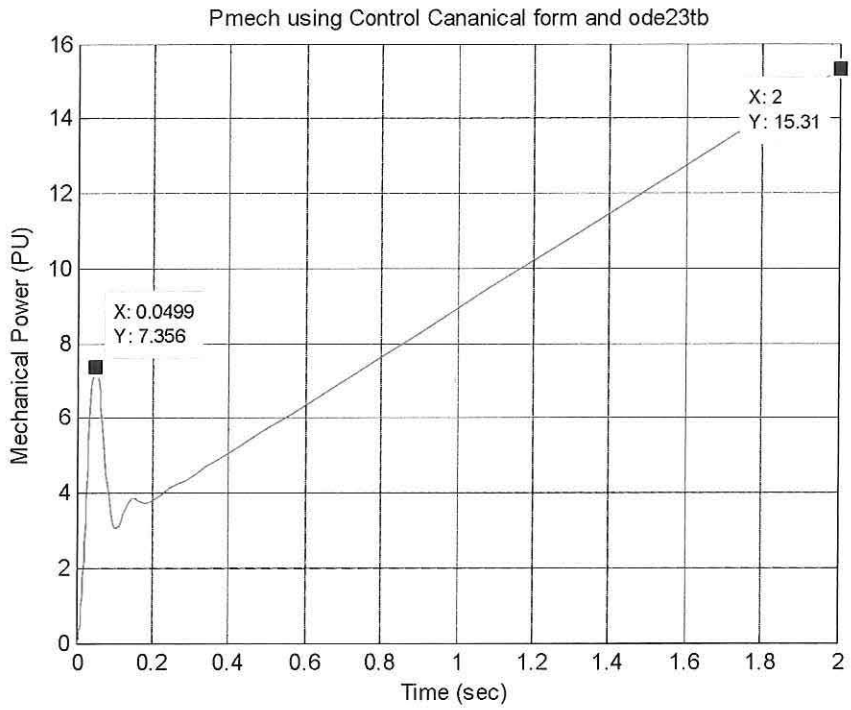

*Figure 13: Mechanical Power using control canonical form simulation (see (A45), T1 < 4*T2)*

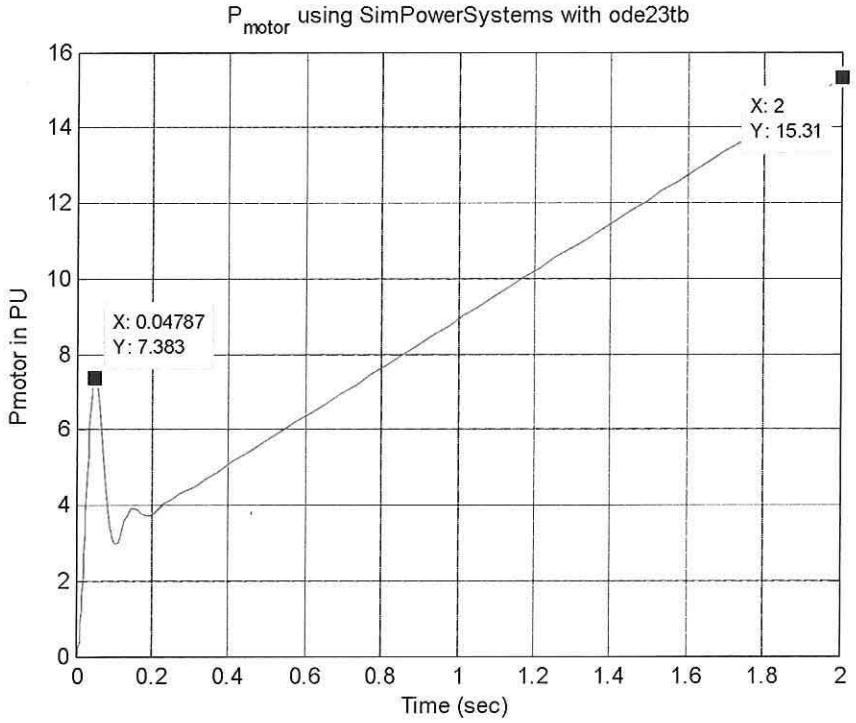

*Figure 14: Mechanical Power using SimPowerSystems with ode23tb (T1 < 4*T2)*

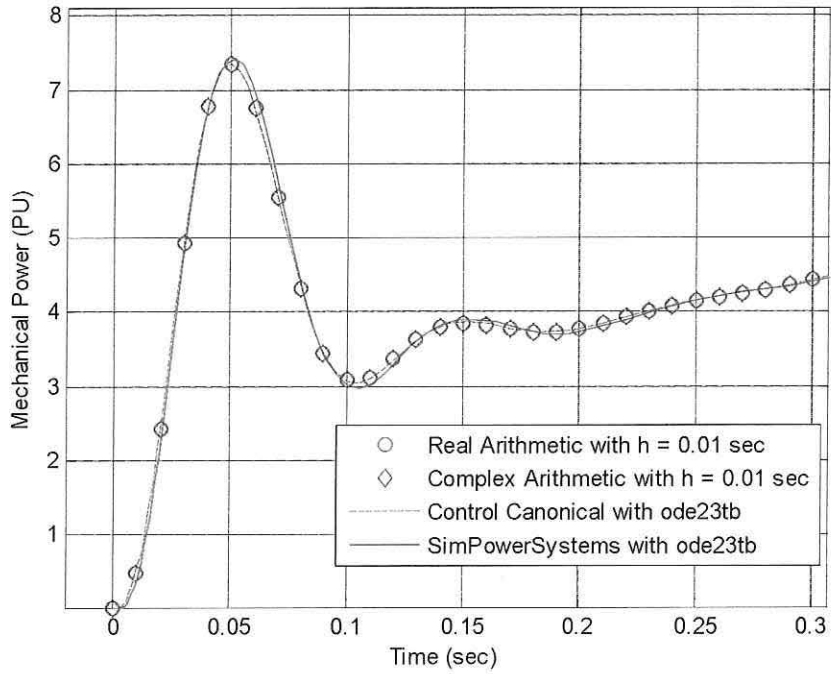

*Figure 15: Zoomed-in Mechanical Power using different integration methods (T1 < 4*T2)*

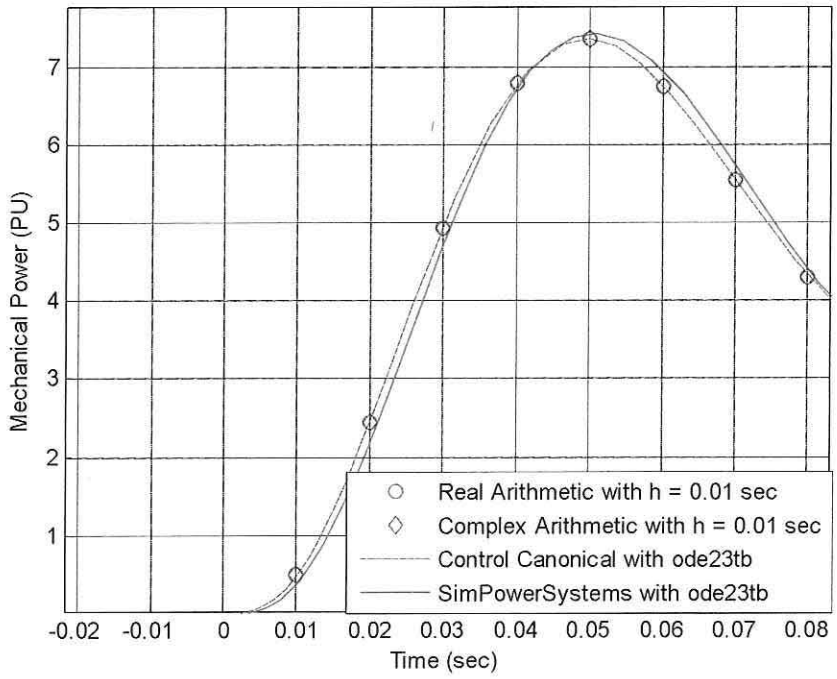

*Figure 16: Further Zoomed-in Mechanical Power using different integration methods (T1 < 4*T2)*

The next step was to change the variable integration method, ode23tb, used in the SimPowerSystems simulation to the Navy-selected ode3 with fixed step size h=5e-5 sec. Figure 17 shows zoomed-in plots comparing the mechanical power using different integration methods and different step sizes (for ode3). The graphs for ode3, h=5e-5 and h =5e-4 nearly coincide. Based on this limited testing, it appears that a significantly larger step size than the selected h = 5e-5 sec. could be used for the diesel engine governor subsystem of the overall simulation.

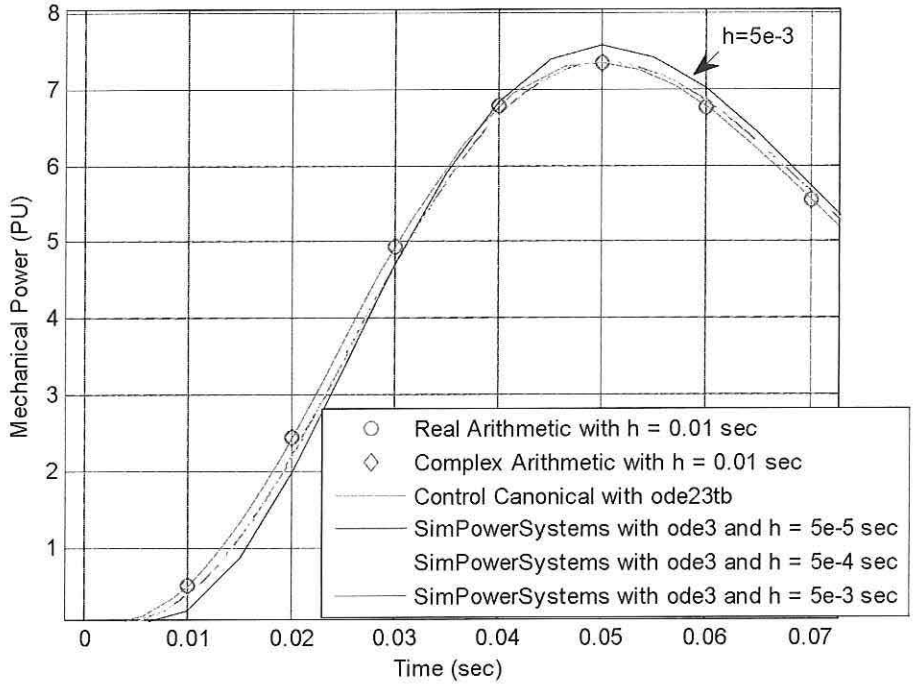

*Figure 17: Mechanical Power using additional integration methods (T1 < 4*T2)*

The topic of simulation execution time is also important. By "execution time" it is meant the wall clock or stop watch time necessary to complete a given "simulation time" such as the 2 sec. shown in Figure 11. For Matlab m-coded simulations (not Simulink/SimPowerSystems) the execution time can be calculated using the "tic" and "toc" Matlab commands. Table 7 provides comparisons of execution times using the Matlab tic and toc commands for 2 seconds of simulated time.

*Table 7: Simulation Calculation Time Comparisons (T1 < 4*T2)*

Integration Method	Average Time (ms)
PFE Real Arithmetic (h=0.01)	7.92
PFE Complex Arithmetic (h=0.01)	8.97
Control Canonical Form (ode23tb)	406.6

Case 2: Real Roots Example. Referring to the quadratic polynomial in (A1), this means that we are now assuming $T_1 > 4T_2$. The simulation parameters used here are the same as those provided in Table 6 except T1 is increased from 0.01 to T1 = 0.1. See Table 8.

*Table 8: Simulation Parameters Used for Real Root Example (T1 > 4*T2)*

Parameter	Value
T_1	**0.10** sec
T_2	0.02 sec
T_3	0.20 sec
T_4	0.25 sec
T_5	0.009 sec
T_6	0.0384 sec
K	40

All the different simulation approaches described above were repeated with the parameters in Table 8 instead of Table 6. Figure 18 compares the step responses using the partial fraction with sample and hold methods of integration (Figure 7, complex arithmetic and Figure 10, real arithmetic), the control canonical form, and SimPowerSystems using ode23tb. These show excellent agreement.

In Figure 19 the simulation result using the SimPowerSystems with ode23tb in Figure 18 is replaced with SimPowerSystems results using ode3 with three different step sizes. As with the Complex Roots Example above, it appears a fixed step size larger than 5e-5 sec. can be used for the simulation of the diesel motor governor. Table 9 compares the simulation execution times for the Matlab-coded simulations over 2 simulated seconds for the real root case.

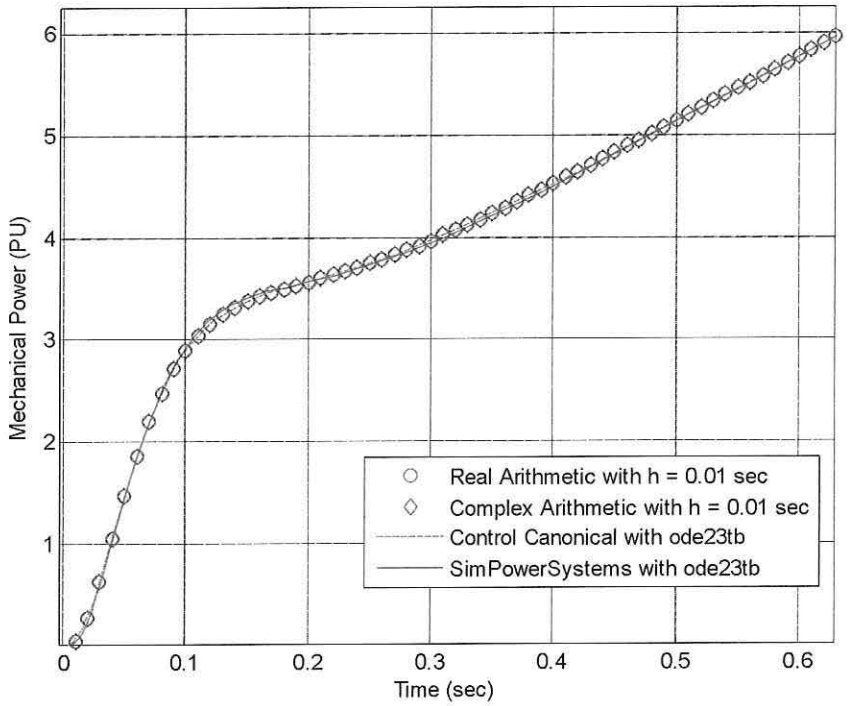

*Figure 18: Mechanical Power using different integration methods (T1 > 4*T2)*

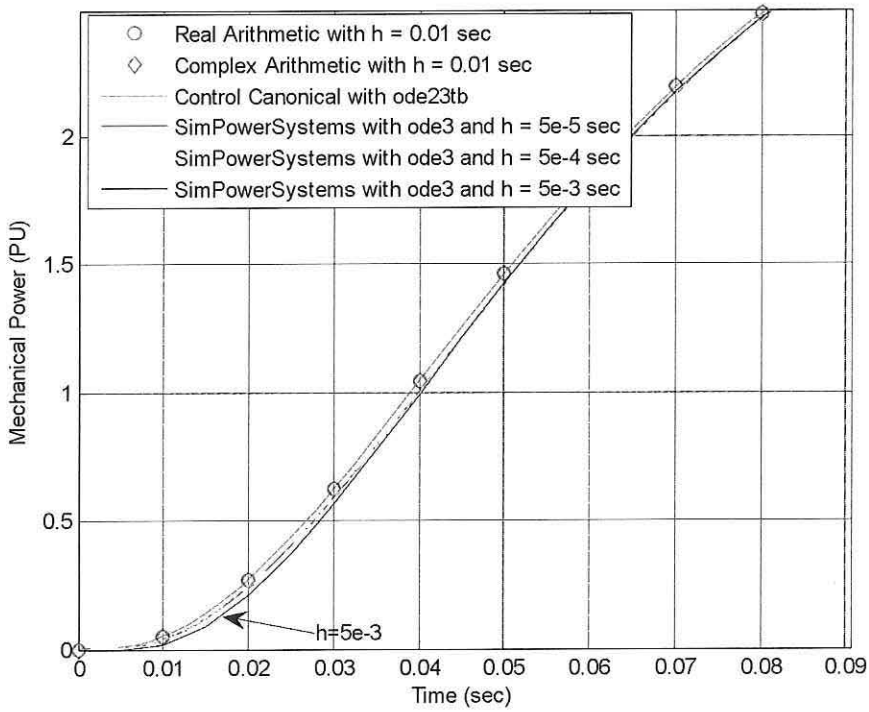

*Figure 19: Mechanical Power using additional different integration methods (T1 > 4*T2)*

*Table 9: Simulation Calculation Time Comparisons (T1 > 4*T2)*

Integration Method	Average Time (ms)*
PFE Real Arithmetic (h=0.01)	9.09
PFE Complex Arithmetic (h=0.01)	6.32
Control Canonical Form (ode23tb)	410.9

Summary and Conclusions

Three sets of differential equations have been obtained from the SimPowerSystems block called Diesel Motor Governor. Simulation results using these sets of equations were used to confirm the results of the SimPowerSystems simulation. Two of the simulation methods (Figure 7 and Figure 10), which are based on a Partial Fraction Expansion and Sample and Hold, were selected because of their ability to readily use multi-rate integration. One of the two methods uses complex arithmetic (Figure 5 and Figure 7) while the other uses only real arithmetic (Figure 9 and Figure 10). The complex arithmetic version is simpler mathematically than the real arithmetic version. However, the simulation examples used here show no obvious speed advantage using complex arithmetic (compare results in Table 7 and Table 9). Thus, the real arithmetic version (Figure 10) is recommended for FPGA implementation. The preliminary simulation results using ode3 indicate a significantly larger integration step than the supplied value of h = 50 μs could be used for the diesel engine governor portion of the overall simulation. Thus, the use of multi-rate integration should be examined further to see if the overall system could be speeded up.

Appendix B: AC1A Model

Figure 1 shows the Simulink diagram in Matlab SimPowerSystems under investigation. To create a mathematical model for implementation in FPGAs, we will need to convert this diagram into an equivalent set of equations. In this appendix the state space equations of an excitation model are presented. A close-up view of this block is shown in Figure 20 below, with the "Block Parameters" information menu shown in Figure 21. To obtain the equations, Matlab coder was used to generate the C code equivalent to the Simulink diagram, and then the software was refined to have one simple code sequence in C. This final C code was then converted into a Matlab code with all needed constants, initial values, and state space equations. To verify the correctness of the new C and Matlab code, input and output values of the AC1A block in the original Simulink diagram were saved from the scopes and compared to the C code and Matlab code derived from the C code. Figure 22 shows that the same input that was used in Simulink code generates the same output in both the Simulink code and Matlab code with very small error.

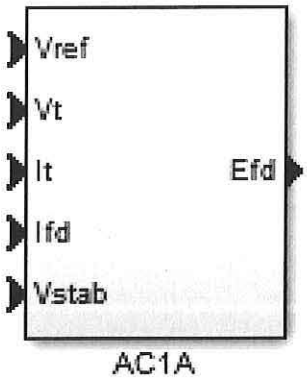

Figure 20: Zoomed-in view of AC1A Excitation block from Figure 1.

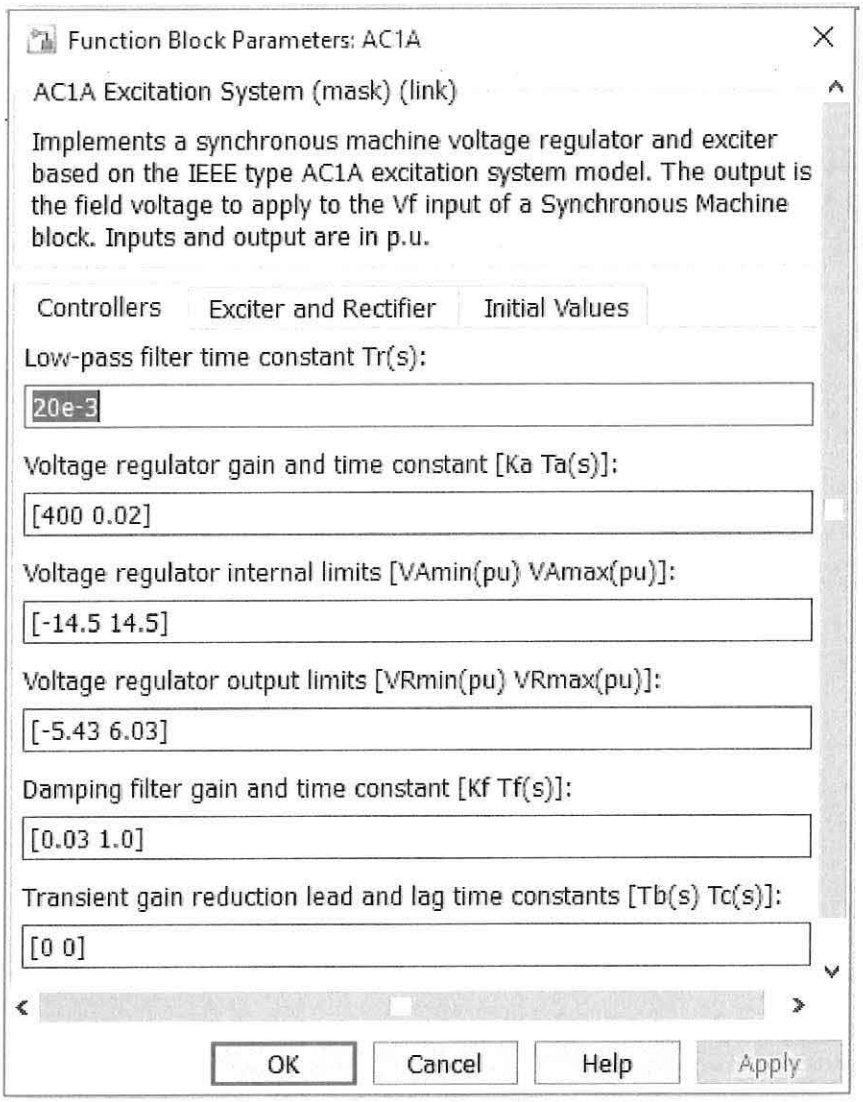

Figure 21: Parameter list for AC1A Excitation block shown in Figure 20.

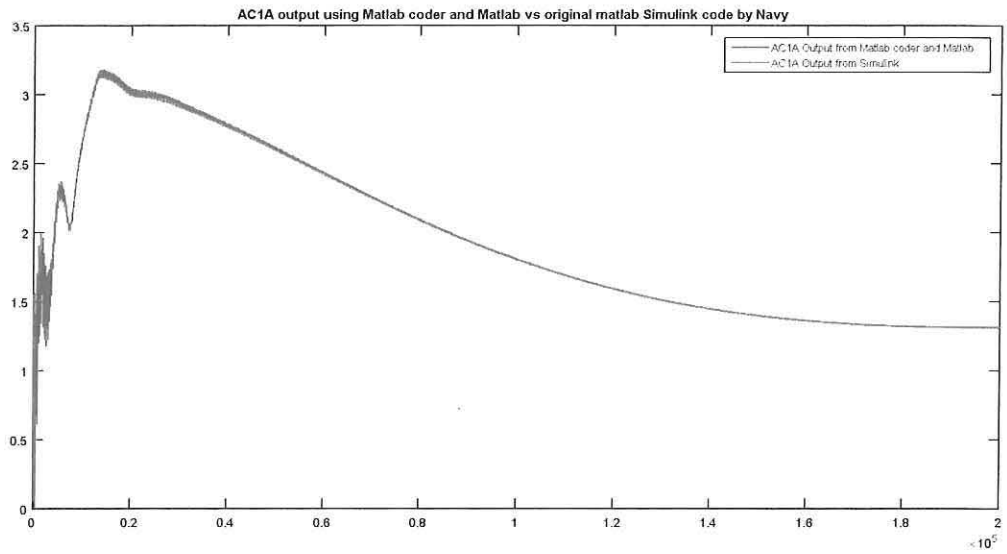

Figure 22: Output of AC1A from the Matlab and Simulink code

The AC1A excitation model has five states:

VR_state: output and state of Main Regulator
Ve: output and state of Discrete-Time Integrator
Efd_next: state of Unit Delay
lpf_state: state of Low Pass Filter on Vt
Vf_state: state of Damping filter

Therefore, the state vector is:

state_vector = [VR_state; Ve; Efd_next; lpf_state; Vf_state];

Up to here, the efforts led into an equivalent code that did not need SimPowerSystems, and state space equations. The obtained equations are a mixture of linear and nonlinear equations.

There are some nonlinearities in the AC1A block diagram such as in the Rectifier block and Saturation Block. These nonlinearities need to be removed before being implemented in FPGA or FPGA functionality must be expanded. If we look under the Mask of the block of the Rectifier and Saturation by right clicking the blocks, we see the components shown in Figure 23. As can be seen the Rectifier block has switch cases which are equivalent to "if statements" in Matlab scripts. There needs to be a way for Switch Cases to be implemented in FPGA. The other nonlinearity is having square root in the Rectifier block. The following expression needs to be calculated in a linear format:

Fex = sqrt (0.75 - rtb_In * rtb_In);

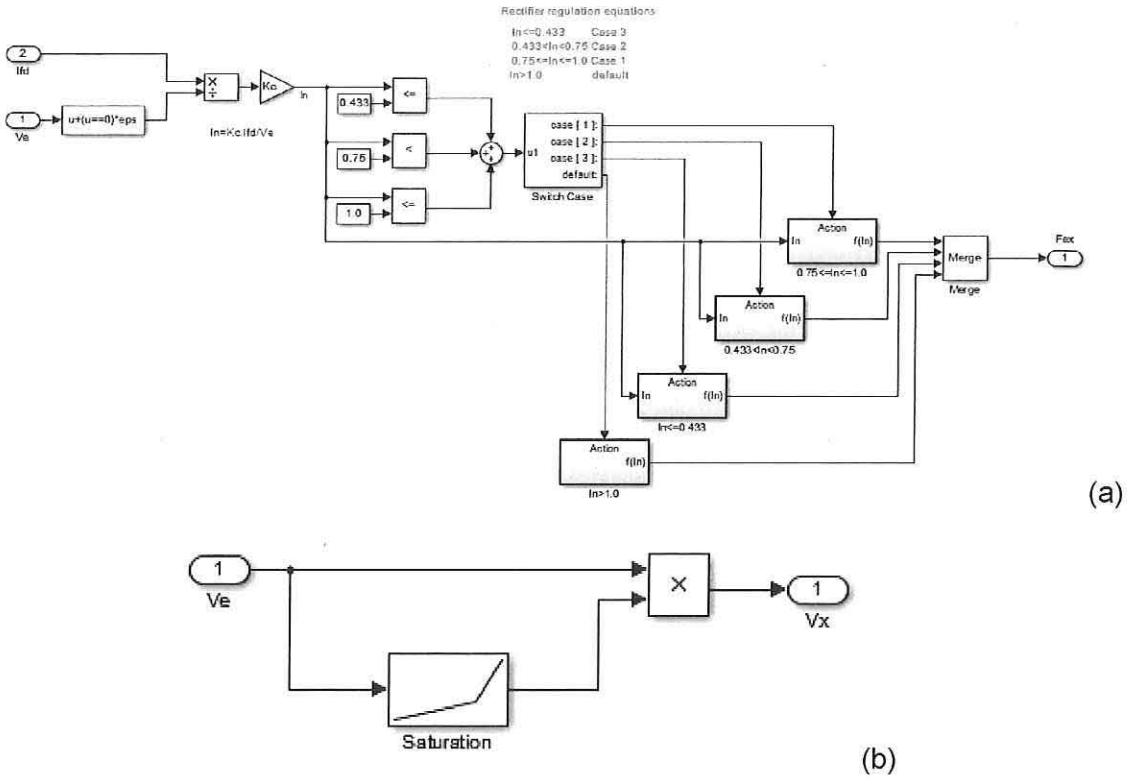

Figure 23: (a) Inside block of Rectifier (b) Inside block of Saturation

A look up table was utilized to linearize this expression and eliminate the need of the square root:

```
x = 0.433:0.001:0.750;
y = sqrt(0.75 - x .* x);
Fex = interp1(x, y, rtb_In);
```

Using the Look up table, the output of the AC1A in comparison to the original Simulink is shown in Figure 24. As can be seen the output (field voltage) matches the Simulink output. The error is mostly small and in the range of 10^{-5} as indicated in Figure 25.

Because of the presence of the "if statements" in some of the blocks, it is not possible to explicitly express the state space vectors in the following format:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Where *x* is the state vector, *u* is the input vector and *y* is the output vector. *A*, *B*, *C* and *D* are matrices of constant numbers or functions of time.

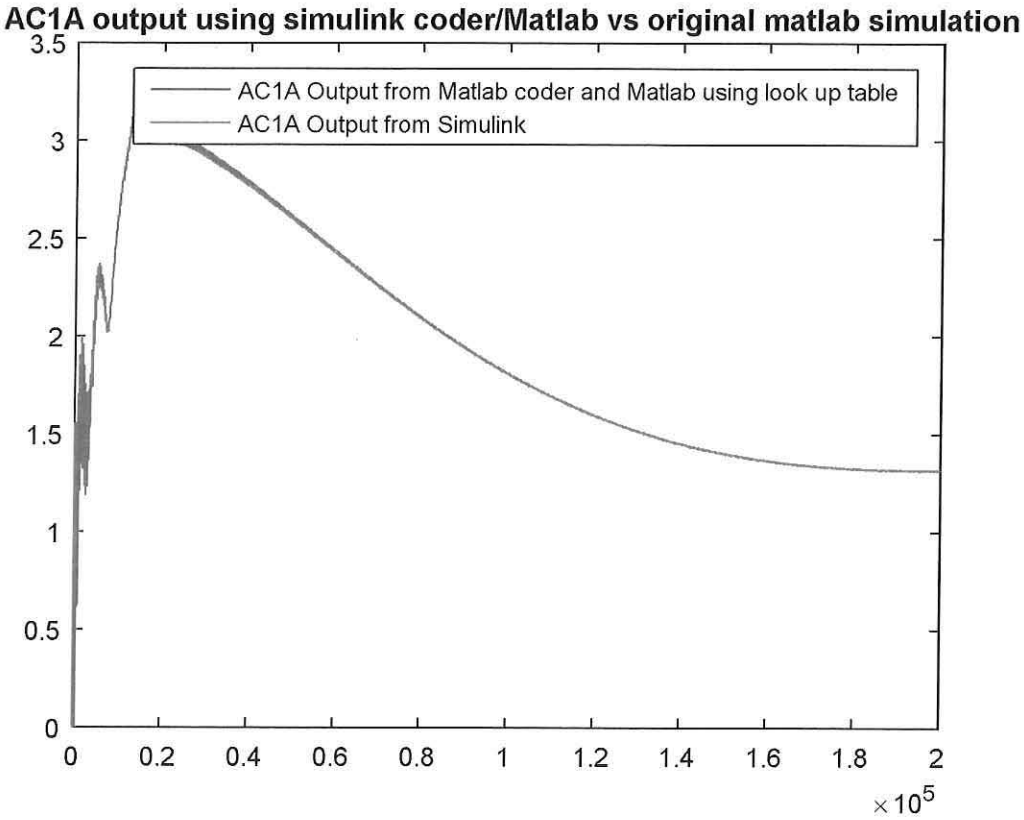

Figure 24: Output of AC1A from Matlab using Look up Table, and Simulink code

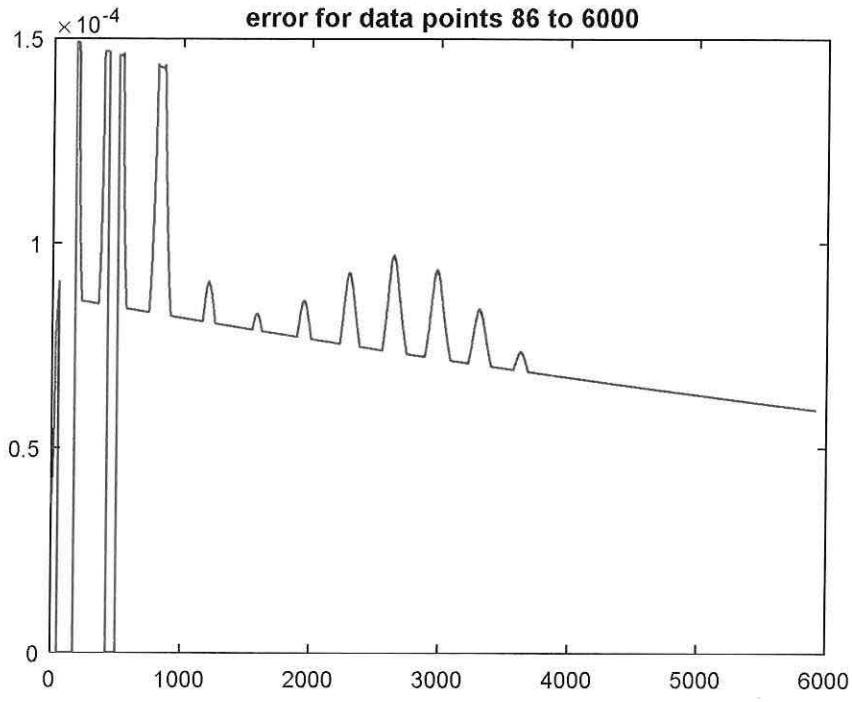

Figure 25: Error between output of the AC1A block using Matlab code and original Simulink file.

Summary and Conclusions

The AC1A excitation model mathematical representation was obtained in a different way than the other components due to the present nonlinearities in the system model. The state space representation could not be implemented in the control canonical form. A look up table was employed to eliminate the square root nonlinear function and was integrated into the state space equations in Matlab to represent the mathematical model of the system. The output voltage of the implemented excitation model completely matched the output that was obtained from the original implementation using the SimPowerSystems Toolbox of Matlab.