Pulse width modulator controller design for a brushless dc motor position servo.

Rossitto, Vincent S.

https://hdl.handle.net/10945/22445
PULSE WIDTH MODULATOR CONTROLLER DESIGN FOR A BRUSHLESS DC MOTOR POSITION SERVO

by

Vincent S. Rossitto

June 1987

Thesis Advisor: Alex Gerba, Jr.

Approved for public release; distribution unlimited.
PULSE WIDTH MODULATOR CONTROLLER DESIGN FOR A BRUSHLESS DC MOTOR POSITION SERVO

ROSSITTO, Vincent S.

Master's Thesis

Pulse width modulator design

Computer aided design, brushless dc motor model, transient response waveshaping, closed loop four quadrant operation, pulse width modulator design.

Recent interest in positioning cruise missile flight control surfaces using electromechanical actuation has prompted a detailed study of brushless dc motor performance in such an application. While the superior response characteristics of these electronically commutated motors are particularly well suited to unidirectional velocity drives, destructive electrical transients associated with rotational reversals of the motor limit its positioning performance. This thesis involves computer aided design of functionally robust brushless dc motor position controller using pulse width modulation. Lumped parameter model simulation and phase plane analysis were performed to attain
preliminary phase plane design, parameter optimization by simulation method.

a preliminary parametric design of the controller. Comprehensive electrical and mechanical analyses were conducted using detailed model simulation to arrive at the final design by parametric optimization. FORTRAN source code listings for all simulations discussed in this thesis are appended and were run on a personal computer.
Pulse Width Modulator Controller Design
for a Brushless DC Motor Position Servo

by

Vincent S. Rossitto
Lieutenant, United States Navy
B.S., United States Naval Academy, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1987
ABSTRACT

Recent interest in positioning cruise missile flight control surfaces using electromechanical actuation has prompted a detailed study of brushless dc motor performance in such an application. While the superior response characteristics of these electronically commutated motors are particularly well suited to unidirectional velocity drives, destructive electrical transients associated with rotational reversals of the motor limit its positioning performance. This thesis involves computer aided design of a functionally robust brushless dc motor position controller using pulse width modulation. Lumped parameter model simulation and phase plane analysis were performed to attain a preliminary parametric design of the controller. Comprehensive electrical and mechanical analyses were conducted using detailed model simulation to arrive at the final design by parametric optimization. FORTRAN source code listings for all simulations discussed in this thesis are appended and were run on a personal computer.
# TABLE OF CONTENTS

I. INTRODUCTION ............................................. 13
   A. BACKGROUND ........................................... 13
   B. PURPOSE ............................................... 14
   C. APPROACH ............................................. 15

II. MODEL DESCRIPTION ........................................... 18
    A. BACKGROUND ........................................... 18
    B. SIMULATION METHOD .................................. 22
    C. POWER CONDITIONER SIMULATION ...................... 25

III. TRANSIENT RESPONSE WAVE SHAPING ......................... 34
     A. BACKGROUND ........................................... 34
     B. COMMUTATION ADVANCE & RETARD ..................... 34
     C. DIODE BIAS THRESHOLD ................................. 43
     D. CURRENT DECAY TIME CONSTANT ...................... 49

IV. CLOSED LOOP CONSIDERATIONS ................................. 55
    A. BACKGROUND ........................................... 55
    B. CLOSED LOOP MECHANICS ................................. 56
    C. FOUR QUADRANT OPERATION .............................. 57
    D. TRANSIENT BEHAVIOR .................................. 61

V. PULSE WIDTH MODULATION ................................... 71
   A. BACKGROUND ........................................... 71
   B. DESCRIPTION ........................................... 72
   C. PARAMETER SELECTION ................................... 77
VI. PRELIMINARY PHASE PLANE DESIGN .......................... 87
   A. BACKGROUND ........................................... 87
   B. LINEAR APPROXIMATION ................................. 88
   C. DEVELOPMENT ........................................... 97
   D. SYSTEM PERFORMANCE .................................... 106

VII. PARAMETER OPTIMIZATION BY SIMULATION METHOD .......... 139
   A. BACKGROUND ........................................... 139
   B. SIMULATION ............................................. 141
      1. Small Scale Step Response ........................... 141
      2. Large Scale Step Response ........................... 156
      3. Ramp Response ....................................... 160
      4. Sinusoidal Response .................................. 166
   C. FINAL DESIGN ........................................... 172

VIII. CONCLUSIONS AND RECOMMENDATIONS .......................... 174
   A. CONCLUSIONS ........................................... 174
   B. RECOMMENDATIONS FOR FURTHER STUDY ...................... 176

APPENDIX A: BRUSHLESS DC MOTOR SIMULATION PROGRAM .......... 178
APPENDIX B: PWM OPEN LOOP TRANSFER CHARACTERISTICS
             PROGRAM ........................................... 242
APPENDIX C: LUMPED PARAMETER MODEL SIMULATION PROGRAM 260
APPENDIX D: PHASE PLANE ANALYSIS PROGRAM ..................... 290
LIST OF REFERENCES ........................................ 324
BIBLIOGRAPHY ................................................ 326
INITIAL DISTRIBUTION LIST .................................. 327
LIST OF TABLES

I. Sensor and Switching Logic .................................... 35
II. Average System Parameters ................................. 89
III. Linear Approximation Validation .................... 97
IV. Piecewise Continuous Approximation .............. 103
V. Summary of Preliminary Design Parameters ........ 140
VI. Summary of Final Design Parameters ............... 173
LIST OF FIGURES

2.1 Basic Modeling Structure .................. 20
2.2 Three Phase Bridge Circuit ................. 21
2.3 Two Window Equivalent Circuit ............. 21
2.4 Motor Current Response with Diodes Inhibited .. 29
2.5 Motor Current Response with Diodes Functioning . 29
2.6 Phase Current Response with Diodes Inhibited .. 30
2.7 Phase Current Response with Diodes Functioning . 30
2.8 Voltage Response (Q5) with Diodes Inhibited .. 31
2.9 Voltage Response (Q5) with Diodes Functioning . 31
3.1 Motor Current Response (Commutation Retarded 5°) 36
3.2 Center Point Trajectory (Commutation Retarded 5°) 36
3.3 Motor Current Response (No Commutation Offset) . 37
3.4 Center Point Trajectory (No Commutation Offset) 37
3.5 Motor Current Response (Commutation Advanced 5°) 38
3.6 Center Point Trajectory (Commutation Advanced 5°) 38
3.7 Mechanical Performance Factor Description ... 40
3.8 Electrical-Mechanical Performance vs Commutation 42
3.9 Modified 3-Phase Bridge Circuit ............. 44
3.10 Motor Current Response (63% Voltage Divider) .. 46
3.11 Motor Current Response (50% Voltage Divider) .. 46
3.12 Phase Current Response (63% Voltage Divider) .. 47
3.13 Phase Current Response (50% Voltage Divider) .. 47

8
3.14 Transistor #5 Voltage Response (63% Voltage Divider) .......................... 48
3.15 Transistor #5 Voltage Response (50% Voltage Divider) .................. 48
3.16 Phase C Current Response (RDADJ = 0Ω) .......................... 51
3.17 Transistor #5 Voltage Response (RDADJ = 0Ω) .......................... 51
3.18 Phase C Current Response (RDADJ = 5Ω) .......................... 52
3.19 Transistor #5 Voltage Response (RDADJ = 5Ω) .......................... 52
3.20 Phase C Current Response (RDADJ = 3Ω) .......................... 53
3.21 Transistor #5 Voltage Response (RDADJ = 3Ω) .......................... 53
4.1 Four Quadrant Operation of a Brushless DC Motor .......................... 60
4.2 Fin Reversal During Position 10° Step Response .......................... 63
4.3 Motor Current Response with No Commutation Reversal Delay .............. 64
4.4 Motor Current Response with Commutation Reversal Delay of 500 μsec .... 64
4.5 Center Point Trajectory with No Commutation Reversal Delay .............. 65
4.6 Center Point Trajectory with Commutation Reversal Delay of 500 μsec .... 65
4.7 Motor Current Response with No Dead Zone .................................. 68
4.8 Motor Current Response with 2° of Dead Zone .................................. 68
4.9 Center Point Trajectory with No Dead Zone .................................. 69
4.10 Center Point Trajectory with 2° of Dead Zone .................................. 69
5.1 Pulse Width Modulation Functional Model .................................. 73
5.2 Relationship Between |E| and Pulse Duty Cycle .......................... 74
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>PWM Open Loop Transfer Characteristics (Linear Input, Low Gain, No Dead Zone)</td>
<td>78</td>
</tr>
<tr>
<td>5.4</td>
<td>PWM Open Loop Transfer Characteristics (Linear Input, High Gain, 1° Dead Zone)</td>
<td>79</td>
</tr>
<tr>
<td>5.5</td>
<td>PWM Open Loop Transfer Characteristics (Sinusoidal Input, Low Gain, No Dead Zone)</td>
<td>80</td>
</tr>
<tr>
<td>5.6</td>
<td>PWM Open Loop Transfer Characteristics (Sinusoidal Input, High Gain, 1° Dead Zone)</td>
<td>81</td>
</tr>
<tr>
<td>5.7</td>
<td>PWM Open Loop Transfer Characteristics (Damped Sinusoidal Input, Low Gain, No Dead Zone)</td>
<td>82</td>
</tr>
<tr>
<td>5.8</td>
<td>PWM Open Loop Transfer Characteristics (Damped Sinusoidal Input, High Gain, 1° Dead Zone)</td>
<td>83</td>
</tr>
<tr>
<td>6.1</td>
<td>Third Order Lumped Parameter Model</td>
<td>89</td>
</tr>
<tr>
<td>6.2</td>
<td>Brushless DC Motor Fin Velocity Step Response</td>
<td>92</td>
</tr>
<tr>
<td>6.3</td>
<td>Third Order Model Fin Velocity Step Response</td>
<td>92</td>
</tr>
<tr>
<td>6.4</td>
<td>Brushless DC Motor Current Step Response</td>
<td>93</td>
</tr>
<tr>
<td>6.5</td>
<td>Third Order Model Motor Current Step Response</td>
<td>93</td>
</tr>
<tr>
<td>6.6</td>
<td>Reduced Order Lumped Parameter Model</td>
<td>94</td>
</tr>
<tr>
<td>6.7</td>
<td>Reduced Order Model Fin Velocity Step Response</td>
<td>96</td>
</tr>
<tr>
<td>6.8</td>
<td>Reduced Order Model Motor Current Step Response</td>
<td>96</td>
</tr>
<tr>
<td>6.9</td>
<td>Closed Loop Reduced Order Model State Diagram</td>
<td>99</td>
</tr>
<tr>
<td>6.10</td>
<td>Phase Plane / Typical PWM Controller</td>
<td>108</td>
</tr>
<tr>
<td>6.11</td>
<td>Phase Plane / Equivalent Relay Controller</td>
<td>109</td>
</tr>
<tr>
<td>6.12</td>
<td>Phase Plane / Equivalent Saturation Amplifier Controller</td>
<td>110</td>
</tr>
<tr>
<td>6.13</td>
<td>Phase Plane Response of 1KHz PWM Controller</td>
<td>113</td>
</tr>
</tbody>
</table>
6.14 Phase Plane Response of 5KHz PWM Controller . 114
6.15 Phase Plane Response of 10KHz PWM Controller . 115
6.16 Phase Plane Response of 20KHz PWM Controller . 116
6.17 Phase Plane Response of $K_{PWM}=.1$ PWM Controller . 118
6.18 Phase Plane Response of $K_{PWM}=3$ PWM Controller . 119
6.19 Phase Plane Response of $K_{PWM}=10$ PWM Controller . 120
6.20 Stable Limit Cycle Due to $K_{PWM}=50$ . . . . . . . 122
6.21 Stable Limit Cycle Due to $f_{PWM}=1KHz$ . . . . . . 122
6.22 PWM Time Response Describing Switching Delay . 123
6.23 Phase Plane Portrait Showing Natural Response . 126
6.24 Phase Plane / Overdamped Step Response . . . . . 127
6.25 Phase Plane / Critically Damped Step Response . 128
6.26 Phase Plane / Underdamped Step Response . . . . 130
6.27 Phase Plane / Preliminary Optimal Design
   Step Response . . . . . . . . . . . . . . . . . . . . . . . 131
6.28 Reduced Order Model Fin Position Step Response . 132
6.29 Third Order Model Fin Position Step Response . 132
6.30 Phase Plane / Preliminary Optimal Design
   Ramp Response . . . . . . . . . . . . . . . . . . . . . . . 134
6.31 Reduced Order Model Fin Position Ramp Response . 135
6.32 Third Order Model Fin Position Ramp Response . 135
6.33 Phase Plane / Unstable Ramp Response . . . . . . 137
7.1 Fin Position Step Response (Preliminary Design) 142
7.2 Motor Current Step Response (Preliminary Design) 143
7.3 Node O Voltage Step Response(Preliminary Design) 144
7.4 Motor Torque Step Response (Preliminary Design) 145
7.5 Fin Position Step Response (TDPLUG=10 μsec) . . 148
7.6 Motor Current Step Response (TDPLUG=10 μsec) . . 149
7.7 Node O Voltage Response (TDPLUG=10 μsec) .... 150
7.8 Motor Torque Step Response (TDPLUG=10 μsec) . . 151
7.9 Fin Position Step Response (Dead Zone = ±.5°) . 153
7.10 Motor Current Step Response (Dead Zone = ±.5°) . 154
7.11 Node O Voltage Response (Dead Zone = ±.5°) . . 155
7.12 Fin Position Large Scale Step Response .... .. 157
7.13 Motor Current Large Scale Step Response .... .. 158
7.14 Node O Voltage Large Scale Step Response .... .. 159
7.15 Fin Position Ramp Response .. .. .. .. .. .. .. .. .. 161
7.16 Motor Current Ramp Response .. .. .. .. .. .. .. .. .. 162
7.17 Motor Torque Ramp Response .. .. .. .. .. .. .. .. .. 163
7.18 Node O Voltage Ramp Response .. .. .. .. .. .. .. .. .. 164
7.19 Transistor #5 Voltage Ramp Response .. .. .. .. 165
7.20 Fin Position Sinusoidal Response .. .. .. .. .. .. .. .. .. 168
7.21 Motor Current Sinusoidal Response .. .. .. .. .. .. .. .. .. 169
7.22 Transistor #5 Voltage Sinusoidal Response .. .. .. 170
7.23 Node O Voltage Sinusoidal Response .. .. .. .. .. .. .. .. .. 171
I. INTRODUCTION

A. BACKGROUND

The ever increasing demand for small, high performance motors for use in such applications as missile flight and space-vehicle control actuators, robotics, and disk drives, justify development of brushless dc motor technology.

Although conventional dc motors are highly efficient and are proven to be well suited for servo motor application, brushless dc motors are generally superior in performance. Perhaps the most significant factor in the recognition of the permanent magnet dc motor as a viable electro-mechanical actuator is the recent advances made in the area of rare-earth magnetic materials. Despite the increased initial cost of samarium-cobalt, for example, its characteristically high magnetic remanence and coercive force provide about twice the flux density of a similar ferrite magnet. The increased torque-to-inertia ratio is desired in high performance actuators. Additionally, improvements in semiconductor technology have further enhanced the realization of electronic commutation. Rapid switching characteristics and low power consumption of solid state devices have widened the gap in performance between brushless and conventional dc motors. Field windings of a brushless dc motor are located in the stator with permanent
magnet in the rotor and commutation is performed using solid state devices, resulting in cooler motor operations. Inherent to the removal of mechanical brushes and commutators is the elimination of arcing and general maintenance associated with conventional dc motors. The physical separation of commutation electronics and the motor provide for a much smaller actuator, capable of functioning in areas previously thought to be too restrictive for electro-mechanical actuation.

B. PURPOSE

Although application of brushless dc motors is rapidly expanding, their popular use is inhibited by the requirement for relatively complex control and power conditioning electronic circuitry. The intrinsic high performance characteristics of these electronically commutated motors are particularly well suited to velocity devices, where the direction of rotation is not routinely reversed. Thus, recent exploitation has been primarily confined to variable speed drives. The intent of this study, however, is to design a functionally robust positioning device capable of cruise missile fin actuation over a broad range of missile flight dynamics.

Unlike the unidirectional kinematics associated with velocity controlled motors, positioning response is invariably characterized by rotational reversals. While the
superior torquing capability of brushless dc motors supports high performance position control, the electronic components which facilitate commutation are subjected to excessive voltage transients and, hence, reduced operational reliability. The central theme of this study involves the design compromises required between system performance and corresponding electrical characteristics.

C. APPROACH

Graphic techniques employed in this thesis are supported by a variety of interactive computer aided design developed specifically for studying the behavior of brushless dc motors. Previous efforts by Thomas [Ref.1] and MacMillan [Ref.2] using the IBM 370 mainframe compiler serve as a foundation from which this study stems. All simulation is performed on a microcomputer rather than the IBM 370 mainframe, and lower level programming in FORTRAN77 replaces the Continuous System Modeling Program (CSMP) language [Ref.3]. While CSMP provides an excellent environment for general simulation, it impedes programming accessibility to various structural mechanisms and primarily provides output characteristics of imbedded functions. Along with enhancing process visibility, coding in FORTRAN facilitates program portability. An objective of this thesis is delivery of appropriate design tools to support on-going efforts in
brushless dc motor development at Naval Weapons Center (NWC), China Lake.

Correlation between electrical and mechanical response characteristics of brushless dc motors is prerequisite to the design problem and best determined with the motor configured as an open loop velocity device. Power conditioner modifications which reduce high voltage transients are investigated. Pulse width modulation is examined as a method for providing accurate position control in a manner congruous with power conditioner electronic limitations.

Optimal selection of design parameters cannot readily be made with the detailed model of MacMillan due to the degree of its complexity. Therefore, preliminary design is accomplished using lumped parameter modeling which provides simulation simplicity, speed, and insight. Additionally, with limited availability of manufacturer's and experimental performance data, which is summarized in [Refs.4&5], the simplified model provides an excellent means of validation and verification of the brushless dc motor simulation.

Phase plane methods used in this study support the graphic nature of nonlinear design and provide considerable insight not only to system performance, but also to the motor's dynamics. The culmination of this study involves
analysis of the linearly approximated design with the brushless dc motor computer simulation.
II. MODEL DESCRIPTION

A. BACKGROUND

Previous efforts in brushless dc motor simulation at NPS have exploited the IBM 370 mainframe and Continuous System Modeling Program (CSMP). Significant attractions of these assets include programming simplicity and computational versatility due mostly to stiff integration methods available. However, current microprocessor capabilities justify the development of such a simulation for analysis on a personal computer.

The programming language chosen for this undertaking is Microsoft FORTRAN77 V3.31. Graphic output is attained via subroutine calls to Plotworks PLOT88 graphics library. Because of the intense computational demand of the simulation, an INTEL 8087 numeric co-processor was exploited.

The motivation for conducting such a study on a personal computer lies in the inherent portability of the product. On-going efforts at NWC, China Lake in brushless dc motor design and analysis are not fully benefited by present NPS computer simulations because of mainframe inaccessibility. Additionally, batch processing of CSMP simulation results in cumbersome development of the model. While microprocessor architecture lacks the computational
power of the mainframe, it is superior in terms of portability and work station availability. All simulations conducted in this study were performed on INTEL 8088/8086 machines, although the universal nature of FORTRAN coding could be easily implemented on any machine capable of being programmed in FORTRAN. The source code for the detailed brushless dc motor simulation program is listed in Appendix A.

The basic modeling structure delineated in MacMillan's work [Ref.2] is illustrated in Figure 2.1 and serves as the starting point of this study. Based on the assumption that the network is balanced and the power supply configuration is split, the relatively complex 3-phase bridge circuit shown in Figure 2.2 simplifies into the two window network of Figure 2.3 [Ref.4]. Back emf voltage of each phase is modeled by MacMillan and summed two at a time to determine loop currents. Reference 2 gives a detailed development of the power conditioner model which includes assumptions for the switching transistor dynamics used in commutation as well as the development of the harmonic air-gap flux used in the motor model.
Figure 2.1 Basic Modeling Structure
Figure 2.2 Three Phase Bridge Circuit

Figure 2.3 Two Window Equivalent Circuit
B. SIMULATION METHOD

Since the simulation environment intrinsic to CSMP is not available in FORTRAN, a viable means of dynamic programming was devised. Natural parallel processing observed in the behavior of electronic components of the motor is simulated using iterative numerical techniques.

Continuous system modeling is attained via the dynamic solution of two nested systems of coupled, nonlinear differential equations. The inner loop establishes incremental states of motor current. Using Thevenin equivalence techniques, potential is applied across nonlinear resistance. This resistance is dependent upon the state of diodes and transistors, as well as the linear resistance of the phase windings. An adaptive Newton-like method for solving these equations was developed. Of particular interest in the inner loop is the presence of multiple solutions, where lax iterative methods could result in convergence to an incorrect solution. Specifically, the parallel relationship between the nonlinear equivalent resistance and motor phase current allow for convergence to either the low or high diode resistance solution. This is particularly prevalent during periods of diode free-wheeling associated with commutation switching. The numerical solution of the inner system of coupled equations becomes extremely stiff and voltage sensitive near the bias threshold of the protective diode. A quantized two-state
solution exists over a narrow range of inputs, resulting in somewhat unpredictable numerical convergence and possible local instability. One solution for phase current incorporates a relatively fast time constant due to the high equivalent resistance contributed by the non-conducting diode, while the conducting diode solution is described by a slower time constant.

The stiff characteristics of the inner loop are adequately handled through the use of a cautious iterative method which is invoked prior to numerical bracketing of the actual solution. During non-bracketed iterations, the simulation step is halved and the system's condition is investigated to ensure that the solution has not been bypassed. This halving technique increases the resolution of the high/low resistance solutions which might otherwise mask each other. This method works reasonably well, but numerical "chattering" is still observed during conditions immediately prior to diode turn-off. Although available stiff integration methods would further minimize the numerical oscillation, size and speed constraints prohibit such integration in this program.

The outer loop solves the motor's position and velocity states and the induced back emf characteristics simultaneously. A simple Newton iterative scheme works well
on the outer loop because of filtering and smoothing provided by the motor's low pass response characteristics.

Despite numerically stiff behavior, the highly nonlinear and discontinuous nature of the model supports the use of a simple small step trapezoidal integrator. Variable step integration involves considerable coding structure and computation time during stiff conditions, neither of which is well suited for the limited architecture of a personal computer. Direct comparison of fixed step results gained in this simulation with variable step results described by MacMillan support the validity of fixed step integration. Convergence criteria establish the solution accuracy and efficiency and are monitored during the execution of the simulation program. Stiff numerical conditions which burden fixed step integration are identified and displayed at the console.

The successful modeling of an analog system in a digital simulation relies heavily on the step size. Particularly, the discontinuous nature of this model increases the likelihood of false response characteristics if the simulation interval is not carefully selected. Oscillations due to subharmonic interaction between the simulation increment and the discontinuous pulse width modulated forcing function may result. Position control with pulse width modulation is best simulated using a small, fixed step rather than a variable step scheme common to
stiff integrators. As a general rule of thumb, the simulation step increment should be at least an order of magnitude less than the PWM reference period to eliminate aliasing and other problems associated with periodically sampled systems.

C. POWER CONDITIONER SIMULATION

Accurate modeling of power conditioning and commutation in brushless dc motors requires explicit definition of the physical behavior of power transistors and their associated protective diodes. Lower level programming is required to properly imbed the conditions necessary to naturally trigger the diodes and permit realistic simulation of their performance. It is presumptuous to force the diodes into the conducting state during the entirety of their thirty mechanical degrees of protective duty. Simulation of the natural behavior of electronic components is of particular importance when the model is configured for position control. Reversal of the direction of rotation may be ordered by the controller at any time or position. The randomness of the forcing function as observed by the power conditioning components necessitates that behavior algorithms be generalized to ensure simulation robustness.

Coil inductance action in brushless dc motors is significant and deserves particular attention in this study. When current, I, flows into a circuit whose inductance is L,
the electromagnetic energy, \( E \), will be stored in it and is given by Equation 2.1.

\[
E = \frac{1}{2}LI^2 \quad \text{(2.1)}
\]

The corresponding voltage across the inductive coil is determined by the change in current flow through it and is described by Equation 2.2.

\[
V = L \frac{di}{dt} \quad \text{(2.2)}
\]

When the inductive circuit is opened, the voltage induced is generally sufficient to forward bias the appropriate free-wheeling diode and cause it to conduct. As long as a low resistance return path for the exponentially decaying current flow is provided, the corresponding voltage response is acceptable. If the protective diode changes to a non-conducting state because of component failure or reverse bias conditions, the equivalent resistance of the return path for current flow becomes very high. The resulting faster time constant of the circuit promotes rapid dissipation of any electromagnetic energy stored in the inductive coil. If significant energy remains stored in the coil at the time of diode turn off, a large \( \frac{di}{dt} \) and undesired high voltage condition will result.
Figures 2.4 through 2.9 describe step velocity current and voltage response characteristics for conditions where the protective diodes are both functional and inhibited. The key observations made from these six plots involve current decay rate dependence on the functional status of the protective diodes and the corresponding voltage conditions observed at the power transistors. Unacceptably high voltage transients that occur across the transistor/diode pair, as shown in Figure 2.8, result when free-wheeling diodes malfunction. Voltage spikes are limited to 800 volts in the computer simulation for purposes of graphic scaling. In Figure 2.9, functioning diodes are observed to reduce peak voltage values to 240 volts. However, when current is decayed with a time constant smaller than the simulation step size, discontinuous behavior results. As long as current decay is numerically discontinuous, the induced coil potential determined by Equation 2.2 is principally dependent on the simulation step size. Therefore, the magnitude of the voltage spike resulting from the rapid decay of 12 amps of phase current is dominated by coil potential and may be approximated analytically.

\[
V = L_{eq} \frac{di}{dt} = 2L_{eq} \frac{\Delta I}{\Delta T}
\]

\[
\Delta I = 12 \text{ amps} ; \Delta T = 4r_{\text{decay}} = 4L_{eq}/R_{eq} = 0.64 \mu\text{volts}
\]

\[
V = 30,000 \text{ volts}
\]
For di to be approximated as $\Delta I$ at a value of 12 amps, then $\Delta t$ may be chosen as small as four times the high resistance decay time constant, $\tau_{\text{decay}}$. Clearly, 30,000 volts of potential across a power transistor is not realizable and junction punch through would most likely relax this condition. However, resulting transistor damage is highly probable. The acceptable peak voltage conditions depicted in Figure 2.9 are affected, but not dominated, by induced coil potential. As shown in Figures 2.5 and 2.7, motor and phase current is decayed with the low resistance time constant, allowing reasonable dissipation of stored electromagnetic energy from the coil.

Digital simulation of an analog component or circuit generally may be accomplished using numerical methods and a step size sufficiently small to preserve the formation of its continuous response characteristics. However, the parallel processing behavior of the inductive coil in a highly nonlinear and discontinuous environment is not modeled as easily.

A protective diode is triggered when its biasing threshold is exceeded, usually the result of commutation switching action. Each finite increment of the digital simulation is uniquely described by a set of state conditions. The modeling of ideal diode characteristics introduces discontinuous behavior in the circuit and precludes numerical solution using simple iterative methods.
Figure 2.4 Motor Current Response with Diodes Inhibited

Figure 2.5 Motor Current Response with Diodes Functioning
Figure 2.6 Phase Current Response with Diodes Inhibited

Figure 2.7 Phase Current Response with Diodes Functioning
Figure 2.8 Voltage Response (Q5) with Diodes Inhibited

Figure 2.9 Voltage Response (Q5) with Diodes Functioning
Consider the incremental voltage and current response of an affected phase leg immediately following commutation switching. The voltage response of the inductive coil is determined by the current flow to which the coil is subjected. When the current flow is apparently cutoff, a large and negative induced voltage is immediately observed. This voltage condition is sufficient to forward bias the protective diode and provide a low resistance return path for the decaying current. However, the low resistance and positive current flow result in a voltage which reverse biases the diode. This apparent toggling of the diode presents difficulty in the numerical convergence and instantaneous solution for each affected step in the simulation.

If the step size could be reduced to an infinitesimally small period, analog simulation would be realized and the diode would behave realistically. Continuous modeling is alternatively achieved through programming methods. The polarity of the coil potential is fixed and not allowed to toggle once the diode has been forward biased, thereby eliminating the numerical instability caused by the discontinuous behavior of the diode. Electromagnetic energy is allowed to dissipate until the magnitude of the coil potential is insufficient to sustain a forward bias condition. This method provides robust modeling of
inductive behavior and eliminates the undesired effects of
digital sampling.

This chapter has described numerical methods employed
to facilitate simulation of a brushless dc motor on a
personal computer. Model development has focused primarily
on electronic commutation and power conditioning. The task
of buffering the voltage transient behavior associated with
commutation switching is investigated in Chapter III.
III. TRANSIENT RESPONSE WAVESHAPING

A. BACKGROUND

Relatively minor adaptations to the power conditioning model developed by MacMillan and illustrated in Figure 2.2 result in dramatic variation of the electronic environment to which the solid state components are subjected. In general, modifications which limit hazardous voltage conditions at the power transistors during commutation switching restrict the motor's overall torque capability.

Stiff current characteristics indicate exaggeration of the transient voltage response acting on solid state devices. Refinement and reshaping of the phase current waveforms may be undertaken with the prospect of minimizing adverse voltage effects.

Modifications to power conditioning circuitry is investigated in this chapter using step velocity response characteristics of the brushless dc motor. Unidirectional kinematics associated with open loop behavior facilitate response waveshaping without the complication of rotational reversal.

B. COMMUTATION ADVANCE & RETARD

The switching logic detailed by Thomas in [Ref.1] for motor rotation in clockwise and counterclockwise directions
is summarized in Table I. This logic assumes maximum
torquing conditions; that is, the rotor is located in such a
position that its field flux is orthogonal to the stator's
electromagnetic field. Hall-effect sensors are used to
provide quantized rotor position information to the power
conditioner for generation of sequencing logic.

TABLE I
SENSOR AND SWITCHING LOGIC

<table>
<thead>
<tr>
<th>Rotor Position</th>
<th>RPS</th>
<th>Clockwise</th>
<th>Counterclockwise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B C Q1 Q2 Q3 Q4 Q5 Q6</td>
<td>Q1 Q2 Q3 Q4 Q5 Q6</td>
</tr>
<tr>
<td>0-30°</td>
<td>1 0 1</td>
<td>0 0 0 1 1 0</td>
<td>0 0 1 0 0 1</td>
</tr>
<tr>
<td>30-60°</td>
<td>1 0 0</td>
<td>1 0 0 1 0 0</td>
<td>0 1 1 0 0 0</td>
</tr>
<tr>
<td>60-90°</td>
<td>1 1 0</td>
<td>1 0 0 0 0 1</td>
<td>0 1 0 0 1 0</td>
</tr>
<tr>
<td>90-120°</td>
<td>0 1 0</td>
<td>0 0 1 0 0 1</td>
<td>0 0 0 1 1 0</td>
</tr>
<tr>
<td>120-150°</td>
<td>0 1 1</td>
<td>0 1 1 0 0 0</td>
<td>1 0 0 1 0 0</td>
</tr>
<tr>
<td>150-180°</td>
<td>0 0 1</td>
<td>0 1 0 0 1 0</td>
<td>1 0 0 0 0 1</td>
</tr>
</tbody>
</table>

Deviation from this orthogonal relationship may be
accomplished by either advancing or retarding commutation.
Displacement in relative angular position between the
electromagnetic field and the field flux is attainable
through modification of the commutation sequencing logic.

Figures 3.1 through 3.6 describe the motor's electrical
response characteristics when commutation is advanced and
retarded. Operational reliability of the motor, which is
largely influenced by the voltage conditions experienced by
power conditioning electronic components, must be considered
when maximizing its mechanical performance. Numerous

35
Figure 3.1 Motor Current Response (Commutation Retarded 5°)

Figure 3.2 Center Point Trajectory (Commutation Retarded 5°)
Figure 3.3 Motor Current Response (No Commutation Offset)

Figure 3.4 Center Point Trajectory (No Commutation Offset)
Figure 3.5 Motor Current Response (Commutation Advanced 5°)

Figure 3.6 Center Point Trajectory (Commutation Advanced 5°)
simulations with varying degrees of commutation advance and retard indicate maximum system responsiveness at orthogonal field conditions.

The free floating center connection of the wye configured model (Node O in Figure 2.2) serves as a common voltage node for the three phase network. This node is offset from its nominal value of 0 volts when load distribution of the network is unbalanced. The trajectory of the center connection is determined by Kirchoff's voltage law and is a function of the generated back emf, phase current through the field windings, and the inductive coil potential. Back emf is predictable under most load conditions and may be uncoupled from the effects of the coil. The trajectory provides a comprehensive indication of voltage transient behavior in the motor during commutation. Minimization of the trajectory's peak voltage values will reduce high voltage conditions experienced by the power transistors.

It is of particular interest to compare the step velocity center point trajectory shown in Figure 3.4 with the response determined by MacMillan and given in [Ref.2, Fig.5.8]. A fundamental premise of MacMillan's model is that the three phase network behaves in a symmetric fashion. This assumption permits the center point trajectory to be
trajectory to be described solely in terms of magnetic flux and ignores the cumulative effects of the inductive coil.

Fin step velocity transient response characteristics provide an appropriate means of quantizing the general responsiveness of the motor. Even during the most demanding periods of flight control, fin actuation dynamics are contained within the initial transient response of the open loop configured velocity device. A mechanical performance factor describing fin positioning responsiveness may be determined by computing the normalized average slope of the fin velocity step response measured between 0 and 60% of steady state velocity. Figure 3.7 illustrates the quantization of mechanical performance of a brushless dc motor with no commutation offset.

![Figure 3.7](image-url)
Electrical performance is well described by the motor's center point voltage response. The transient effects of commutation switching and back emf generation characterize the trajectory. Excessively high voltage conditions observed at the center connection correspond to unacceptable behavior at the power transistors. Therefore, electrical performance is quantized by normalizing average peak values of the center point trajectory.

Figure 3.8 illustrates the relationship between commutation offset and electrical-mechanical performance of the brushless dc motor for ±30° of commutation offset angle. Commutation at orthogonal field conditions provides maximum system responsiveness, which degrades in a somewhat symmetrical manner as advance and retard angles are increased.

Center point trajectory peak voltage values in excess of 500 volts are illustrated in Figure 3.6. Therefore, the unacceptable electrical response behavior associated with advanced commutation invalidates its application in this design. Additionally, significant retarding of commutation is required to noticeably reduce the high voltage conditions associated with commutation switching. Because of the substantial compromise in performance at these retarding offsets, neither commutation advancing nor retarding was deemed advantageous and, therefore, not implemented.
PERFORMANCE VS COMMUTATION

Figure 3.8 Electrical-Mechanical Performance vs Commutation
C. DIODE BIAS THRESHOLD

The demanding operating conditions encountered by the brushless dc motor in service necessitate current limiting protection. Previous models have provided this protection with a relatively large power supply series resistance. Fluctuations in load and general performance of the motor are comfortably absorbed using such a method of current limitation. However, the scheme reduces motor supply line voltage to an unacceptably low level with respect to biasing threshold requirements of the protective diodes. Even though protective diodes are functioning properly during commutation switching, stiff current conditions are observed near the peaks of current ripple in the response shown in Figure 3.3. During cyclic conditions of maximum emf generation, protective diodes are falsely forward biased due to insufficient line voltage. Random diode triggering may short active circuits and result in abrupt behavior in current and voltage responses.

Modification of the 3-phase model shown in Figure 2.2 allows for the same value of equivalent supply resistance to be divided in series across each transistor-diode pair and is illustrated in Figure 3.9. The reconfiguration permits selection of the voltage divider relationship necessary for correct triggering of the protective diodes during commutation switching and maintains a viable means of current limiting protection through series resistance.

43
Voltage divider percentage is determined by Equation 3.1.

\[ \text{Voltage Divider \%} = \frac{R_{\text{in}}}{R_{\text{in}} + R_{\text{out}}} \] (3.1)

Figures 3.10 through 3.15 illustrate the effects on current and voltage response when the free-wheeling diode's bias threshold is varied by voltage divider adjustment. As observed in Figure 3.3, insufficient bias threshold results in unexpected diode firing and short circuiting of the affected phase. On the other hand, excessive bias threshold
conditions, such as the 63% voltage divider configuration illustrated in Figures 3.10, 3.12, and 3.14, result in premature cutoff of the protective diodes during free-wheeling conditions. The electromagnetic energy stored in the coil is not sufficiently dissipated through a low resistance path and high voltage conditions are observed across the power transistors in Figure 3.14. It is interesting to note that excessive bias threshold conditions preclude forward biasing of protective diodes during only a single switching operation as illustrated in Figure 3.10 at \( t=2.5\text{msec} \). This is explained by the velocity dependence of back emf. At lower speeds, the inductive coil is insufficiently supplemented by back emf generation to forward bias the protective diode. At higher speeds, however, back emf generation is significant and adequately contributes towards satisfying diode triggering criteria. A 50% voltage divider ensures robust diode operation and provides effective current limiting. The corresponding electrical transient characteristics of this configuration are depicted in Figures 3.11, 3.13, and 3.15. Figure 3.15 clearly indicates reduction of the peak voltage across the transistor.
Figure 3.10 Motor Current Response (63% Voltage Divider)

Figure 3.11 Motor Current Response (50% Voltage Divider)
Figure 3.12 Phase Current Response (63% Voltage Divider)

Figure 3.13 Phase Current Response (50% Voltage Divider)
Figure 3.14 Transistor #5 Voltage Response (63% Voltage Divider)

Figure 3.15 Transistor #5 Voltage Response (50% Voltage Divider)
D. CURRENT DECAY TIME CONSTANT

The exponential decay of current in a relaxing phase is described uniquely by a dynamic time constant. The circuit's time constant is determined by the phase inductance, L, divided by its equivalent path resistance. Previous modeling provides either high or low path resistance conditions determined by the state of the protective diode. A high resistance state yields extremely fast time constants which results in high di/dt values. Conversely, the low resistance state is described by a relatively slow decay transient. If the transient's duration exceeds the time required for 30° of mechanical rotation, then the remaining electromagnetic energy stored in the coil will be exponentially dissipated with the fast time constant. This condition is characterized by undesired high voltage transients across the power transistors and is found to be more prevalent at higher operating speeds where less time for energy dissipation is available.

The simulation program is provided with an adjustable resistance placed in series with each protective diode to permit control of the phase current's decay time during free-wheeling. Location of the adjustable resistance (RDADJ) in the power conditioning model is given in Figure 3.9. Average response characteristics of the motor are unaffected by this added resistance since it is negligible when compared to the equivalent loop resistance. However,
RDADJ dominates the current decay return path resistance and essentially determines the circuit's time constant. Physical implementation is feasible since the added resistor is external to the motor and actually a component of the power conditioner.

Figures 3.16 through 3.21 graphically depict the waveshaping effects of RDADJ on phase current and transistor voltage. Selection of the rate adjusting resistance is a fairly straightforward task since it is clearly bounded by its consequences. Increased decay rate results from increasing the resistance (RDADJ) and corresponds directly to higher voltage transients at the power transistors. Figures 3.18 and 3.19 illustrate the intolerable consequences associated with the selection of a very fast time constant. A 3Ω resistance for RDADJ improves the decay time constant significantly with an increased but acceptable transistor voltage transient peak, which is shown in Figure 3.21.
Figure 3.16 Phase C Current Response (RDADJ = 0Ω)

Figure 3.17 Transistor #5 Voltage Response (RDADJ = 0Ω)
Figure 3.18 Phase C Current Response (RDADJ = 5Ω)

Figure 3.19 Transistor #5 Voltage Response (RDADJ = 5Ω)
Figure 3.20  Phase C Current Response (RDADJ = 3Ω)

Figure 3.21  Transistor #5 Voltage Response (RDADJ = 3Ω)
Step velocity response waveshaping has been exploited to optimize the transient characteristics of the brushless dc motor and its associated electronic commutation circuitry. Many of the benefits of such refinement are not readily apparent during unidirectional rotation. However, the highly accentuated transient behavior associated with closed loop position control is significantly affected by these modifications and will receive further attention in the Chapter IV.
IV. CLOSED LOOP CONSIDERATIONS

A. BACKGROUND

NPS studies of brushless dc motors primarily support variable speed control applications [Refs. 1,2,6,7]. Very limited documentation concerning its viability in closed loop position control appears in the referenced literature. Previous research at NPS has also been restricted mainly to open loop (velocity) performance characteristics. However, brushless dc motor cruise missile fin control was investigated and simulated by Franklin using an equivalent lumped parameter model [Ref.7]. Additionally, MacMillan's detailed model provides rudimentary observation of motor current transient behavior when rotational direction is reversed.

In addition to the unidirectional kinematics normally associated with variable speed control, closed loop position control includes such conditions whereby mechanical rotation and electrically developed torque oppose each other. The intention of this chapter is to study closed loop positioning characteristics of brushless dc motors, with particular emphasis on the transient behavior associated with reversing rotational direction.
B. CLOSED LOOP MECHANICS

The mechanics of position control of a permanent magnet dc motor are broadly described by three electromechanical conditions. The natural operational state of a working dc motor is known as motoring. This condition is characterized by electrical to mechanical energy conversion and occurs when the motor is driven between zero and steady-state speeds. When the motor is forced to rotate at a greater rate than its steady-state speed either by external disturbance or decreased supply voltage, generator action is realized. Generation converts mechanical to electrical energy and, if suitably configured, can be used to replenish the power supply through regeneration. The mechanical energy consumed in electrical regeneration dynamically brakes motor rotation and provides increased deceleration during relaxation conditions. The third operational state is forced braking and occurs when the motor is forced to rotate against its present direction. Associated with the resulting intense rotational deceleration are electrical transients that are accentuated and often result in destructive magnitude of current through and voltage across the power transistor/diode pair. [Ref. 8]
C. FOUR QUADRANT OPERATION

The electronic commutation of a three phase, 4-pole brushless dc motor configured for position control may be more comprehensively described as four quadrant operation [Ref.9]. Following in similar manner, Figure 4.1 illustrates the electromechanical conditions describing the four quadrants. Electromagnetic field orientation determines the electrically developed torque, $T$, which forces rotation in a prescribed direction. This torquing direction is denoted by the variable "DIR" in the computer simulation and is accomplished through logic sequencing of the power transistors. Actual mechanical rotation, $\omega$, describes the kinematic state of the rotor.

The transient dynamics involved in attaining positional steady state may be classified as forward motoring, forward braking, reverse motoring, and reverse braking. Except for directional differences, response characteristics of forward and reverse motoring are symmetric, as are those of forward and reverse braking. As shown in Figure 4.1, reversal of commutation direction may be buffered by generator action if any degree of system relaxation is provided by the controller.

Quadrants II and IV of Figure 4.1 describe motoring conditions, where actual mechanical rotation ($\omega$) and electrically developed torque ($T$) are in the same direction. During this mode of operation, response
characteristics are electrically similar to unidirectional, open loop behavior. Transient conditions are relatively well behaved since protective diodes are able to dissipate electromagnetic energy stored in relaxed phase inductive coils as intended. The back emf generated while motoring converts the mechanical energy of rotation to electrical energy proportional to the speed of rotation and is opposite in polarity of the forcing potential. The superposition of source and generated voltage net a reduced electromotive force and, therefore, impede and eventually limit the motor's rotational rate.

Forced braking immediately follows the reversal of electronic commutation and is illustrated in quadrants I and III of Figure 4.1. This condition is commonly referred to as plugging in a brush-type motor and occurs when the applied voltage is reversed in polarity. Plugging of brushless dc motors is accomplished through commutation logic sequencing, since supply voltage polarity is fixed. Consider the lightly damped fin angle stepped response also shown in the figure. When the forward running motor comes to zero speed under braking action at \( t=t_1 \), it automatically gets accelerated in the reversed direction since the commutation logic for forward braking and reverse motoring is identical. While rapid braking and reversing characteristics are desired in a position control system, the exaggerated transient behavior of the motor is
electronically demanding and may result in damage to solid state devices.

Figure 4.1 also indicates the presence of generator action immediately adjacent to commutation reversal. If the controller provides suitable conditions for natural response of the motor, energy is converted by regenerative processes and dynamic braking is attained. Controllers which operate with a dead zone or have soft response characteristics often promote generator action and are associated with better behaved transients. Conversely, stiff controllers with no dead zone, such as an ideal relay, do not experience natural response conditions when plugged. Therefore, the buffering effect provided by dynamic braking is not exploited by high performance controllers and the corresponding electrical response is characterized by excessive transient behavior.
Figure 4.1 Four Quadrant Operation of a Brushless DC Motor
D. TRANSIENT BEHAVIOR

The excessive transient behavior observed during braking is explained by several factors. Generated back emf is now of the same polarity as the forcing potential and the superposition of source and generated voltages net an enhanced electromotive force. The resulting responsiveness of the system, as well as its transient behavior, is increased.

Electromagnetic energy stored in an apparently cutoff phase inductive coil for the three phase, 4-pole motor is intended to be dissipated within the time required for thirty degrees of mechanical rotation. When the motor is initially plugged, less time is likely to be available for decay and high voltage conditions may result.

During forced braking conditions, the direction of actual mechanical rotation (ω) opposes the prescribed commutation logic sequence and results in commutation switching performed in reverse order. This apparent back-stepping through the sequencing logic results in reversal, rather than redirection, of phase current flow within the motor. The effects of rapidly reversing current flow within the motor is significant due to the nature of the inductive coil and result in intolerable voltage transients.

The commutation logic which sequences the switching of power transistors should be sensitive to the controller's
mode of operation. While in the forced braking mode, complete transistor switching cannot be performed simultaneously without catastrophic results, and a two step sequence is called for. Stored coil energy in the affected phase should be minimized prior to reversing the direction of the current flow through it. Therefore, a time delay is required between switching off the existing current and switching on the reversing current. During this momentary period of natural response, phase current should be decayed with a reasonable time constant to reduce the voltage transients associated with current reversal. If the commutation delay exceeds the current decay time, near complete dissipation of the energy stored in the coil will be realized and the reversal of current can be performed acceptably.

The rate of decay of electromagnetic energy stored in a relaxed coil is determined by the circuits time constant. The addition of a 3Ω resistance in the decay path, as discussed in Chapter III, provides more rapid dissipation of the stored energy. Less delay time is required for faster decaying current conditions, thereby minimizing system performance degradation during commutation switching. Figure 4.2 depicts a typical first reversal encountered during fin position response when a 10° step of demand is applied using an ideal relay type controller.
The effect on current and voltage waveform of commutation switching delay in fin position control during this reversal is illustrated in Figures 4.3 through 4.6. Transient current behavior and center connection voltage response (Node 0) with no commutation delay applied are shown in Figures 4.3 and 4.5. Current is observed to be decayed very rapidly in Figure 4.3 when plugged due to the nearly instantaneous reversal of phase current. The rapid change in current flow induces very large coil potential, exhibited in Figure 4.5 as a voltage spike in excess of 1000 volts.
CLOSED LOOP MOTOR CURRENT RESPONSE
NO COMMUTATION REVERSAL DELAY

Figure 4.3 Motor Current Response with No Commutation Reversal Delay

CLOSED LOOP MOTOR CURRENT RESPONSE
.0005sec COMMUTATION REVERSAL DELAY

Figure 4.4 Motor Current Response with Commutation Reversal Delay of 500 μsec
Figure 4.5 Center Point Trajectory with No Commutation Reversal Delay

Figure 4.6 Center Point Trajectory with Commutation Reversal Delay of 500 µsec
Figures 4.4 and 4.6 depict much more acceptable response behavior when a 500µsec delay is provided. The exponential decay of motor current shown in Figure 4.4 is much slower, however, and does not completely dissipate the coil energy. A voltage spike of 160 volts is observed in Figure 4.6 and corresponds to the rapid dissipation of remaining coil energy. Complete dissipation of the stored coil energy is apparently not practical, particularly at high rotational rates, due to relatively long decay time requirements. However, voltage transients during braking are minimized by nearly an order of magnitude using a delay time of 500µsec.

In general, highly responsive position control of a brushless dc motor is attained at the expense of its electrical transient behavior. Specifically, the level of energy stored in an inductive coil immediately prior to braking determines the extent of its transient. Extremely stiff controllers, such as an ideal relay, provide maximum electromotive force to the motor at all times during positioning. Figures 4.7 and 4.9 illustrate the unacceptable transient current and voltage behavior associated with the unbuffered bang-bang response. Figure 4.7 indicates a rapid transient behavior of plugged phase current which results in the 1000 volt spike shown in Figure 4.9. The inclusion of a ±2° dead zone invokes generator action whose effects are described in Figures 4.8
and 4.10. The relatively large window provided for system relaxation by \( \pm 2^\circ \) dead zone is clearly depicted in Figure 4.8, with current characterized by the dominant effects of back emf and dissipating coil energy. Figure 4.10 indicates a corresponding voltage spike of only 150 volts was induced. A finite amount of stored coil energy was converted to mechanical energy during motor relaxation provided by controller dead zone. However, system accuracy is sacrificed when dead zone is applied and the duration of natural system response required to significantly reduce the high voltage transients is unacceptably large. Additionally, a fixed dead zone which is effective at lower speeds, may not provide sufficient system relaxation at greater rotational rates where less time is required to transit the dead zone. Figure 4.10 shows an increased voltage spike magnitude of 300 volts associated with a higher speed reversal. While dead zone can be used to reduce voltage transients, it is not solely sufficient to ensure acceptable behavior.
Figure 4.7 Motor Current Response with No Dead Zone

Figure 4.8 Motor Current Response with 2° of Dead Zone
Figure 4.9 Center Point Trajectory with No Dead Zone

Figure 4.10 Center Point Trajectory with 2° of Dead Zone
It has been shown in this chapter that high performance position control of brushless dc motors is generally attained at the expense of its power conditioning electronic component reliability. In support of the need for responsive fin actuation, reasonable steady state accuracy and buffered electrical transient behavior, pulse width-modulation is examined in Chapter V.
V. PULSE WIDTH MODULATION

A. BACKGROUND

Two distinct categories of error signal amplifiers, linear and nonlinear, are available for this application. The difference between the two lies primarily in the manner in which they drive the solid state devices of the power conditioner. Linear servo-amplifiers drive bipolar transistors in their active regions and permit continuous voltage regulation, while nonlinear servo-amplifiers maintain transistor operation in either saturation or cutoff. Pulse width modulation (PWM) is one method of providing nonlinear servo-amplifier control of a brushless dc motor and is the focus of discussion in this chapter.

While linear servo-amplifier circuits are relatively simple and essentially free of electrical noise, they are subject to thermal damage [Ref.8]. The fact that these amplifiers drive bipolar transistors in their linear region explains the substantial amount of power dissipation into heat that takes place in the commutation switching electronics. Space constraints in the cruise missile fin actuator application preclude the use of large heat sinking elements necessary to protect solid state components from thermal damage. Conversely, pulse width modulated amplification is characterized by negligible power
dissipation into heat. Although two state transistor operation is thermally advantageous, it is a source of increased electrical noise due to switching. Fortunately, PWM is negligibly influenced by noise since the random variation of amplitude has little effect on the information contained in the pulse width modulated signal. Additionally, the extent of current ripple and subharmonic oscillation due to the periodic output of the pulse width modulator is insignificant [Ref.6]. Since the power transistors are switched on and off at a frequency far exceeding the motor's bandwidth, the majority of the high frequency component of the modulated signal is filtered by the motor.

Pulse width modulation is a technique for regulating the amount of energy supplied to the motor for development of torque. Effective regulation is accomplished by modulating a fixed voltage amplitude on a reference frequency and linearly scaling its pulse duty cycle.

B. DESCRIPTION

The PWM scheme modeled in this study behaves similarly to the Dither Method outlined by Askinas [Ref.6] and is illustrated in Figure 5.1. A reference sawtooth waveform of desired frequency and amplitude is generated through independent circuitry and is differenced with the closed loop system's amplified position error signal.
The amplifier is comprised of linear and nonlinear regions of operation. Nonlinear gain, denoted as "$K_{PWM}\$" in the computer simulations appended, is applied as saturation amplification to the error signal. Relay elements are used to convert the bias of the summed signal to a tri-state logic output which is summarized in Figure 5.1. Because the polarity of the source voltage is fixed, directional rotation is accomplished using separate commutation sequences as shown in Table I. "DIR" is the variable used to indicate commutation direction in the simulation and is assigned a value of $+1$ for clockwise rotation and $-1$ for counterclockwise rotation. With "DIR" accounting for
directional convention, forcing potential (VIN) is controlled by two-state logic. When PWM logic triggers the application of 150V supply voltage to the motor and determines the commutation sequencing logic requested by the controller, powering conditions remain fixed for the remainder of the reference period.

Figure 5.2 illustrates the convenient one-to-one relationship between the nonlinearly amplified error signal (|E|) and the duty cycle of the pulsed output (PULSE). Due to the saturation amplification of the system error (POSERR), |E| is bounded within the range from zero to unity.

![Figure 5.2 Relationship Between |E| and Pulse Duty Cycle](image)

Figure 5.2 Relationship Between |E| and Pulse Duty Cycle
THRESH describes the biased error, and is given by Equation 5.1. When THRESH is less than the reference sawtooth signal during each cycle, VIN is pulsed on by the PULSE signal.

\[
\text{THRESH} = 1 - |E| \quad (5.1)
\]

The linear nature of the reference sawtooth signal over each pulsing period results in linear scaling of pulse duty cycles when the error is in the range \(0 < |E| < 1\). The relationship observed in Figures 5.1 and 5.2 between \(E\) and the duty cycle of VIN is given by Equation 5.2.

\[
\text{Duty Cycle of VIN} = |E|*T \quad (5.2)
\]

Figures 5.3 through 5.8 are examples of open loop transfer characteristics of the pulse width modulator modeled in this study. Pulse width modulation of ramp and sinusoidal (undamped and exponentially decaying) error signals are examined with varying amplifier gain and dead zone to illustrate its properties. Open loop PWM transfer characteristics are ascertained using the simulation listed in Appendix B. Figure 5.3 depicts the linear scaling range of the PWM amplifier for a specific gain. In this case, \(K_{PWM}\) (gain) equals 0.5 and, when applied to the continuous input ramp error signal, results in an increased range of
modulation for the directional logic signal. Figure 5.4 illustrates the effect of increased amplifier gain as well as the inclusion of a controller dead zone. For an identical ramp error signal input, the increased gain results in a reduced linear region of modulation. Dead zone is observed in Figure 5.4 near the zero-crossing of the input error signal \((t=2.5\text{msec})\) and is characterized by a corresponding Directional Logic level of 0. Therefore, the use of dead zone in the PWM amplifier buffers the effects of switching when the error signal changes polarity. Figure 5.5 illustrates the pulse width modulation of an undamped sinusoidal error signal with low amplifier gain and no dead zone. The sinusoidal waveform is reasonably well preserved, but would be much improved with a higher sampling frequency. Modulation of the same sinusoid at a higher gain and with dead zone is shown in Figure 5.6. Very little width modulation takes place when gain is high due to saturation effects and causes VIN to resemble the output characteristics of a relay with dead zone. The pulse width modulation of relatively large scale sinusoidal error inputs is examined in Figures 5.5 and 5.6. In position control application, however, error amplitudes associated with stable systems tend to decrease with time. Additionally, larger amplifier gains for this Type 1 system provide improved steady state accuracy and is further discussed in Chapter VI. The exponentially decaying
sinusoid used as an input error signal in Figures 5.7 and 5.8 presents a more realistic representation of the range of inputs to be encountered. The low gain configuration illustrated in Figure 5.7 results in linear modulation of the waveform over the range examined and is characterized by the Directional Logic signal pulse width being proportional to the input error signal amplitude. The pulse width characteristically decreases with the exponentionally decaying magnitude of the input. Conversely, the modulation using high gain with the same input is shown in Figure 5.8 and results in saturation of the output pulse as shown by fixed pulse width during periods of relatively large input magnitude.

C. PARAMETER SELECTION

Time averaging provided by low pass characteristics of the brushless dc motor smooth the effects of the discontinuous pulse width modulated forcing function. The PWM may be analytically treated as a piecewise continuous function if specific approximation criteria are satisfied. Validity of such an approximation is principally determined by the reference sawtooth frequency of the PWM. Lumped parameter analysis of this typical motor used in this study reveals an average mechanical time constant, $r_M$, of 0.1
Figure 5.3 PWM Open Loop Transfer Characteristics (Linear Input, Low Gain, No Dead Zone)
Figure 5.4 PWM Open Loop Transfer Characteristics
(Linear Input, High Gain, 0.1° Dead Zone)
PWM OPEN LOOP PERFORMANCE
GAIN = 0.50    DEAD ZONE = 0.00

Figure 5.5  PWM Open Loop Transfer Characteristics
(Sinusoidal Input, Low Gain, No Dead Zone)
Figure 5.6 PWM Open Loop Transfer Characteristics
(Sinusoidal Input, High Gain, .1° Dead Zone)
Figure 5.7 PWM Open Loop Transfer Characteristics (Damped Sinusoidal Input, Low Gain, No Dead Zone)
Figure 5.8 PWM Open Loop Transfer Characteristics (Damped Sinusoidal Input, High Gain, .1° Dead Zone)
seconds and an electrical time constant, $r_E$, of 0.00165 seconds. Despite the dominant low pass characteristics of the slow mechanical time constant, empirical derivation based on computed results indicates the pulse width modulation reference sawtooth frequency should be at least 10KHz. At such frequencies, the motor effectively smooths the pulsed forcing action of the PWM. Phase plane methods discussed in Chapter VI are used to graphically illustrate the effects of pulse frequency on system response. System simulations using lower sawtooth waveform frequencies indicate stable but very oscillatory response behavior. It is recognized that solid state power switching characteristics pose a limitation on the maximum realizable PWM reference frequency. Therefore, a 10KHz pulse frequency is selected on the basis of response smoothing and hardware constraints.

The selection of simulation step size is largely dependent upon the reference sawtooth waveform frequency. The Nyquist rate determines the minimum sampling frequency required to reconstruct the assumed bandlimited sawtooth waveform. If the step size is chosen less than one-half of the reciprocal of the highest significant frequency component ($f_C$) of the sawtooth waveform's Fourier transform, then aliasing will be avoided [Ref.10].
Equations 5.3 and 5.4 determine the Nyquist sampling period ($T_s$) of a periodic sawtooth waveform of period $T_p$ and amplitude $E_M$.

$$H(f) = \int_0^{T_p} \frac{E_M}{T_p} te^{-j2\pi ft} \, dt$$

$$T_s = \frac{1}{2f_c} = 50 \mu\text{sec}$$

However, simulation based on the Nyquist rate is observed to be inadequate and does not provide accurate pulse width modulation of error signals. The full range of the sawtooth response ($0 \leq \text{VREF} \leq 1$) must be reconstructed in the simulation to support modulation of small amplitude error signals. Observation of computed results indicates that a minimum simulation step size of an order of magnitude less than the PWM pulse period is required for accurate simulation.

PWM amplifier gain determines the region of pseudo linear operation. Consequently, large gains result in hard response characteristics near steady state, while low gains allow soft behavior. The interaction of PWM frequency and gain may result in unacceptable residual oscillations and ballistic overshoot conditions. This self excited oscillation, or limit cycle, is commonly associated with
nonlinear servo-amplifier control and will be studied more completely in Chapter VI using phase plane methods.
VI. PRELIMINARY PHASE PLANE DESIGN

A. BACKGROUND

The highly nonlinear and discontinuous nature of a pulse width modulated position controller precludes the use of most classical stability analysis methods. Computer simulation lends itself well to simple time response analysis, but restriction to such a method often masks many of the underlying characteristics which determine a system's general behavior. Nonlinear controllers are most often analyzed with either describing function or phase plane techniques. The describing function is a frequency response method which ultimately determines transfer function characteristics of the control system. However, the discontinuous nature of pulse width modulation limits the effectiveness of the describing function. Phase plane methods, on the other hand, are well suited to handle the PWM stability problem provided that a second order model is applicable and the methods are complemented by numerical simulation of the time response of state trajectories.

The underlying premise of analyzing a second order system in the phase plane is that linear and nonlinear components of its characteristic differential equation can be partitioned. Consider the general form of a second
order differential equation describing the response of \( e(t) \) and given by Equation 6.1.

\[
\dot{e} + f(e, \dot{e}) = 0 \tag{6.1}
\]

The term \( f(e, \dot{e}) \) is a nonlinear function of \( e(t) \) and \( \dot{e}(t) \) when equation 6.1 describes the fin actuator's performance. Rather than studying \( e(t) \) and \( \dot{e}(t) \) parametrically, the composite function \( \dot{e}(e(t)) \) can be determined and studied graphically. The phase plane portrait of \( \dot{e}(t) \) versus \( e(t) \) bears a wealth of information describing a system's dynamics which is otherwise lost in parametric analysis of the time solutions.

The intent of this chapter is to develop the phase plane method of analysis as a primary tool for studying the dynamics associated with position control of brushless dc motors. Additionally, preliminary values for design parameters will be determined based on system performance criteria.

B. LINEAR APPROXIMATION

State space methods associated with phase plane portrait development are not well suited for analysis of the detailed brushless dc motor model. Such an approach involves the use of a third order model with very stiff dynamics and would be computationally cumbersome and inefficient. However, the detailed model may be
approximated by the linear lumped parameter third order model shown in Figure 6.1. Average system parameters for the typical motor used in this study are given in Table II.

![Diagram of Third Order Lumped Parameter Model]

**Figure 6.1 Third Order Lumped Parameter Model**

**TABLE II**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Viscous Friction</td>
<td>$f$</td>
<td>.01 [oz-in/sec]</td>
</tr>
<tr>
<td>Motor Inertia</td>
<td>$J$</td>
<td>.001 [oz-in/sec$^2$]</td>
</tr>
<tr>
<td>Equiv Resistance</td>
<td>$R$</td>
<td>9.7 [ohms]</td>
</tr>
<tr>
<td>Equiv Inductance</td>
<td>$L$</td>
<td>.0016 [henries]</td>
</tr>
<tr>
<td>Torque Constant</td>
<td>$K_T$</td>
<td>15.9 [oz-in/amp]</td>
</tr>
<tr>
<td>Back EMF Constant</td>
<td>$K_B$</td>
<td>.112 [volts/rad/sec]</td>
</tr>
</tbody>
</table>

The lumped parameter model's validity as an acceptable representation of the 3-phase, 4-pole brushless dc motor may be demonstrated using analytic and computer simulation methods. The closed loop transfer function describing the
block structure of Figure 6.1 for $\Omega(s)/E_a(s)$ is given by Equation 6.2.

$$\frac{\Omega(s)}{E_a(s)} = \frac{K_T}{(R + sL)(f + sJ) + K_T K_B}$$  \hspace{1cm} (6.2)

For a step input magnitude of $E_a = 150v$, steady state output can be obtained using the final value theorem. The steady state velocity ($\omega_{ss}$) is determined in Equation 6.3.

$$\omega_{ss} = \frac{E_a - K_T}{fR + K_T K_B} = 1270 \text{ rad/sec}$$  \hspace{1cm} (6.3)

Assuming a 10:1 mechanical transmission speed reduction, velocity may be converted from rad/sec to fin-deg/sec.

$$\omega_{fss} = \frac{(1270 \text{ rad})(57.3 \text{ deg})(0.1)}{\text{sec rad sec}} = 7277 \text{ fin-deg sec}$$

Motor current is also of interest in the verification of the averaged parameter model and the transfer function $I(s)/E_a(s)$ is given by Equation 6.4.

$$\frac{I(s)}{E_a(s)} = \frac{f + sJ}{(R + sL)(f + sJ) + K_T K_B}$$  \hspace{1cm} (6.4)
Again, using a step input of magnitude $E_a = 150\text{v}$, steady state current is obtained using the final value theorem and is calculated in Equation 6.5.

$$I_{ss} = \frac{E_a f}{fR + K_T K_B} = 0.79 \text{amps} \quad (6.5)$$

Velocity and current response characteristics of both the third order lumped parameter model and the detailed brushless dc motor model are simulated using programs listed in Appendices A and C, respectively, and the results for fin velocity and current response are illustrated in Figures 6.2 through 6.5. Simulated response characteristics support analytically determined steady state behavior of the third order lumped parameter model. The ripple component in the motor current caused by back emf generation in the brushless dc motor constitutes the principal difference in performance between the two models. General performance similitude indicates the brushless dc motor may be accurately represented with respect to its mechanical and electrical response using an averaged parameter model.

Although phase space methods support solution of third order systems, the extra simulation time and programming effort involved do not justify inclusion of a third dimension in the portrait of this system. The electrical
Figure 6.2 Brushless DC Motor Fin Velocity Step Response

Figure 6.3 Third Order Model Fin Velocity Step Response
Figure 6.4 Brushless DC Motor Current Step Response

Figure 6.5 Third Order Model Motor Current Step Response
and mechanical time constants of the motor are determined by Equations 6.6 and 6.7.

\[ r_E = \frac{L}{R} = \frac{0.0016}{9.7\Omega} \quad (6.6) \]

\[ r_M = \frac{J}{f} = \frac{0.001 \text{ (oz-in/s}^2\text{)}}{0.1 \text{ (oz-in/s)}} = 0.1 \text{ sec} \quad (6.7) \]

Clearly, the mechanical time constant dominates general system response and results in low pass behavior. Neglecting the fast time constant, that is, assuming no circuit inductance, the system performance should remain virtually unchanged. This reduced order model is shown in Figure 6.6 and can be verified as a legitimate substitute for the third order model.

![Reduced Order Lumped Parameter Model](Figure 6.6 Reduced Order Lumped Parameter Model)
Equation 6.8 gives the closed loop transfer function relationship describing the reduced order model.

\[
\Omega(s) = \frac{K_T/JR}{E_a(s)} \frac{s + fR + K_TK_B}{JR} \quad (6.8)
\]

The final value theorem is used to determine the steady state velocity when a step amplitude \( E_a = 150\, v \) is applied.

\[
\omega_{ss} = \frac{K_T E_a}{fR + K_TK_B} = 1270 \, \text{rad/sec} \quad \omega_{ss} = 7277 \, \text{fin-deg/sec} \quad (6.9)
\]

The motor current response start-up performance is significantly affected by the removal of circuit inductance and its associated lag, but retains similar steady state characteristics as determined by Equations 6.10 and 6.11 and illustrated in Figures 6.7 and 6.8.

\[
I(s) = \frac{f + sJ}{R (f + sJ) + K_TK_B} \quad (6.10)
\]

\[
I_{ss} = \frac{E_a f}{fR + K_TK_B} = 0.79 \, \text{amps} \quad (6.11)
\]

Table III summarizes performance characteristics of the detailed model of the brushless dc motor, its third order linear approximation, and the reduced order model.
Figure 6.7 Reduced Order Model Fin Velocity Step Response

Figure 6.8 Reduced Order Model Motor Current Step Response
Correlation is very strong between the detailed and reduced order models, substantiating its use.

### TABLE III
LINEAR APPROXIMATION VALIDATION

<table>
<thead>
<tr>
<th></th>
<th>Brushless DC Motor</th>
<th>Third Order Approximation</th>
<th>Reduced Order Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_{ss}$ [°/sec]</td>
<td>7670</td>
<td>7277</td>
<td>7277</td>
</tr>
<tr>
<td>$I_{ss}$ [amps]</td>
<td>1.4 (avg)</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>$t_{\text{settle}}$</td>
<td>0.0275</td>
<td>0.0225</td>
<td>0.0225</td>
</tr>
<tr>
<td>$(\pm 2%)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{\text{peak}}$</td>
<td>14.59</td>
<td>14.20</td>
<td>15.5</td>
</tr>
<tr>
<td>$t_{\text{rise}}$</td>
<td>0.0145</td>
<td>0.0118</td>
<td>0.0120</td>
</tr>
<tr>
<td>$(10%-90%)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. DEVELOPMENT

The computer aided phase plane program developed for analyzing this system is listed in Appendix D and graphically superpositions the simulated time dependent trajectory and the state space solutions for a sufficiently representative region of values of $e$ and $\dot{e}$. The analytically determined state space solutions are illustrated as slope markers which indicate the instantaneous slopes of $\dot{e}(e(t))$ at any condition $(e,\dot{e})$. If the range of $\dot{e}$ and $e$ to be examined is reasonably

---

$\text{†}$ Steady state conditions not reached at $t = 0.3$ sec. Extended simulation reveals $I_{ss}(t=0.05 \text{ sec}) = 1.1$ amp.
representative of the normal operating conditions of the system, stability may be validated by the exclusion of any singular points, areas of local convergence, or areas of divergence. The criterion for stability in this design is absolute convergence to a stable equilibrium point representing steady state conditions or, as is common in many nonlinear systems, convergence to a limit cycle about steady state.

A limit cycle is an isolated closed path or trajectory in the phase plane and describes a system's self-excited oscillation [Ref.11]. If the limit cycle is unstable, the closed loop path constitutes the boundary between stable and unstable regions. For a stable limit cycle, trajectories outside and in the vicinity of a stable limit cycle will converge to it, while those trajectories inside will diverge to the limit cycle. Limit cycle behavior is anticipated in pulse width modulated position control of the brushless dc motor due to inherent time delay in switching and is further investigated in this chapter.

A simulated trajectory represents actual system performance over time when specific initial conditions are imposed. While the trajectory offers tremendous insight into the system's dynamics for specific response conditions, the unique value of phase plane analysis lies in the general state space solution presented in the phase plane portrait. Figure 6.9 illustrates the reduced order
model configured for closed loop fin position control. The block structure is expanded to identify necessary states of the second order system.

Figure 6.9 Closed Loop Reduced Order Model State Diagram

The system's state equations are developed for the purpose of defining instantaneous slope vectors ($\eta$) as functions of error ($e$) and error rate ($\dot{e}$).

$$\eta = f(e, \dot{e}) = \frac{d \dot{e}(t)}{d e(t)}$$  \hspace{1cm} (6.12)
Fin position \( (\theta_{\text{FIN}}) \) in degrees is assigned the state variable \( x_1 \). Its derivative state \( (\omega_{\text{FIN}}) \) in fin-deg/sec is \( x_2 \).

\[
\dot{x}_1 = x_2 \quad (6.13)
\]

To describe higher order state variables, it is necessary to introduce and apply a scalar transmission factor \( (\Omega^\circ) \) which converts units from Motor Radians to Fin Degrees.

\[
\Omega^\circ = \frac{n \times 180^\circ}{\pi} \quad (6.14)
\]

The variable "n" of Equation 6.14 is the motor to fin mechanical transmission gear ratio and is assumed to be 0.1 in this study. Angular acceleration of the fin in deg/sec\(^2\) is given by Equation 6.15.

\[
\dot{x}_2 = \frac{\Omega^\circ K_T \text{VIN}(e, \dot{e}) - (K_T K_B + fR) x_2}{JR} \quad (6.15)
\]

Closed loop system error \( (e) \) is defined via position and velocity feedback circuits and is given by Equation 6.16. The dynamic nature of the missile guidance and flight control result in the fin controller behaving more as a position tracking vice positioning device. Therefore, while step response analysis is important to determine the
general behavior of the motor, it is solely inadequate to characterize the overall behavior of the system. Ramp response analysis more completely describes the motor's dynamic positioning performance. Using the phase plane simulation developed for this study, a ramp input whose time dependence is scaled by its slope (r) is used to evaluate system tracking performance. Step response analysis is performed by describing initial perturbation conditions (e(0), e(0)) and zeroing the ramp slope.

\[ e = \dot{r} t - K_p x_1 \]  \hspace{1cm} (6.16)

Differentiating Equation 6.16 yields the system's error rate, \( \dot{e} \).

\[ \dot{e} = \dot{r} - K_p x_2 \]  \hspace{1cm} (6.17)

Manipulation of these equations of state is required to determine expressions for \( x_2 \) and \( \dot{x}_2 \) solely in terms of \( e \) and \( \dot{e} \). Equations 6.17 may be rewritten as Equation 6.18.

\[ x_2 = \frac{\dot{r} - \dot{e}}{K_p} \]  \hspace{1cm} (6.18)

\[ \dot{x}_2 = \frac{\partial K_T K_P \text{VIN}(e, \dot{e}) - (K_T K_B + f R)(\dot{r} - \dot{e})}{J_R K_P} \] (6.19)

Finally, the instantaneous slope function \( \eta \) is given by Equation 6.20, and is composed of natural and forced components.

\[ \eta = f(e, \dot{e}) = \frac{d \dot{e}(t)}{d e(t)} = - \frac{K_P \dot{x}_2}{r - K_P x_2} = - \frac{K_p \dot{x}_2}{\dot{e}} \]

\[ \eta = -\frac{\partial K_T K_P \text{VIN}(e, \dot{e}) + (\dot{e} - \dot{r})(K_T K_B + f R)}{J_R \dot{e}} \] (6.20)

The slope function \( \eta \) is linear in terms of \( e \) and \( \dot{e} \), except for the \( \text{VIN}(e, \dot{e}) \) term. The nonlinearity arises from transfer function characteristics of the pulse width modulator. Because of the distinct segregation of linear and nonlinear elements, however, the model may be approximated as a piecewise linear system. The phase plane is subdivided into several regions whose commonalty is local linear operation. These regions are graphically bounded by lines of discontinuity on the phase plane portrait. Table IV describes the five piecewise continuous regions of operation associated with saturation amplifier and PWM servo control. The use of relay control involves regions 1, 4, and 5 only.
### TABLE IV

**PIECEWISE CONTINUOUS APPROXIMATION**

<table>
<thead>
<tr>
<th>Region#</th>
<th>Condition</th>
<th>VIN(e, \hat{e})</th>
<th>Slope</th>
</tr>
</thead>
</table>
| 1       | |\[|P_1| \leq DBAND\]| 0 | \[
(K_TK_B+fR)(\hat{r}-e)\]
|         | (Dead Zone) |                |       | \[
JR\hat{e}\]
|         |            |                |       | \[
\cdots \text{where} \quad P_1 = e + \frac{eK_v + e_{ss}}{K_p}\]
| 2       | 0 < P_2 < 1 | 150P_2         | \[-(150P)\Omega K_TK_p+(\hat{r}-e)(K_TK_B+fR)\] | \[
JR\hat{e}\]
|         | (Positive Linear) |                |       | \[
\cdots \text{where} \quad P_2 = K_{PWM}(e + \frac{eK_v - DBAND - e_{ss}}{K_p}\]
| 3       | -1 < P_3 < 0 | 150P_3         | \[-(150P)\Omega K_TK_p+(\hat{r}-e)(K_TK_B+fR)\] | \[
JR\hat{e}\]
|         | (Negative Linear) |                |       | \[
\cdots \text{where} \quad P_3 = K_{PWM}(e + \frac{eK_v + DBAND - e_{ss}}{K_p}\]
| 4       | P_4 \geq 1 | 150P_4         | \[-(150P)\Omega K_TK_p+(\hat{r}-e)(K_TK_B+fR)\] | \[
JR\hat{e}\]
|         | (Positive Saturation) |                |       | \[
\cdots \text{where} \quad P_4 = K_{PWM}(e + \frac{eK_v - DBAND - e_{ss}}{K_p}\]
| 5       | P_5 \leq -1 | 150P_5         | \[-(150P)\Omega K_TK_p+(\hat{r}-e)(K_TK_B+fR)\] | \[
JR\hat{e}\]
|         | (Negative Saturation) |                |       | \[
\cdots \text{where} \quad P_5 = K_{PWM}(e + \frac{eK_v + DBAND - e_{ss}}{K_p}\]

103
Excluding dead zone error, the steady state error, $e_{ss}$, of this Type 1 system is of finite value for ramp response and zero when stepped. $e_{ss}$ is determined by Equation 6.21 and permits correct ramp response prediction using state space solution techniques.

$$K_v = \lim_{s \to 0} \frac{s \{ \hat{r}(s) \}}{1+G(s)H(s)}$$

$$e_{ss} = \frac{\hat{r}}{K_v} = \frac{\hat{r}(fR+KT_Kp)}{nKTKpKpKpKpKp} + \frac{\hat{r}}{Kp} + \text{Dead Zone} \quad (6.21)$$

$K_a$ is the amplifier gain associated with the nonlinear controller. $K_{PWM}$, as defined in Chapter V, is substituted for $K_a$ when pulse width modulation is employed.

Interpretation of Equation 6.21 describes the effects of amplifier gain on $e_{ss}$. If $K_a$ is infinitely large, such as in the case of an ideal relay, the steady state error is minimized and is approximated by Equation 6.22.

$$e_{ss}(\text{relay}) = \frac{\hat{r}}{Kp} + \text{Dead Zone} \quad (6.22)$$

However, as $K_a$ is reduced, the first term of Equation 6.21 bears more significance and the steady state error is increased. Thus, system accuracy is readily ascertained using analytical methods and is presented for comparison.
with the simulated trajectory as an asterisk ("*") on the phase plane portrait.

As discussed previously, the output of the PWM is either 0 or 150v. However, the pulse duration is a nonlinear function of error. Since the pulse width modulator is described in terms of e and \( \dot{e} \), as well as by time, difficulty exists in trying to illustrate three dimensional characteristics on a two dimensional phase plane. However, the nonlinear dependency on time may be alleviated through linear approximation methods. This is accomplished by time averaging the response of the PWM. The result of such a linearization closely approximates the characteristics of a saturating amplifier. Specifically, the function which describes the average output of these two nonlinear elements is the same. The average dc voltage of the PWM is defined by Equation 6.23.

\[
\overline{V_{DC}} = \frac{1}{T} \int_{0}^{T} V_{out} \, dt = \text{Pulse Duty Cycle} \times 150v \quad (6.23)
\]

The variable "T" represents the period of the PWM's pulse cycle.

The instantaneous error of the system is nonlinearly scaled by the PWM amplifier gain \( K_{PWM} \) and was discussed in detail in Chapter V where essentially, error undergoes normalized saturation amplification. The normalized value
of amplified error is the same as the pulsed voltage output
duty cycle of the PWM.

This one-to-one relationship between input error and
output voltage circumvents the otherwise required
dependency on the time variable to describe pulse duration.
The phase plane portrait of a saturating amplifier will be
used to predict the simulated trajectory of the pulse width
modulated reduced order model. It must be kept in mind,
however, that while the piecewise continuous saturating
amplifier provides a good average approximation of the
PWM's discontinuous characteristics, instantaneous
deviations will exist.

D. SYSTEM PERFORMANCE

Four principal design parameters are realistically
available for optimizing system performance. The lumped
parameters of the reduced order model preclude variation of
actual component characteristics of the brushless dc motor.
Coefficients for position and velocity feedback (K_p and
K_v), pulse width modulation amplifier gain (K_{PWM}), and PWM
pulse frequency (f_{PWM}) are considered free parameters in
this design and are the focus of study in this section.

Since the pulse width modulator's performance can be
most clearly evaluated relative to ideal conditions,
analysis of the system using a similarly configured
saturating amplifier and ideal relay are initially
considered. The continuous time saturating amplifier describes the PWM with very high amplifier gain. Identical simulation conditions are imposed on each system configuration. Small scale step response is attained using an initial perturbation of 3°. The pulse width modulator is configured with intentionally low values of amplifier gain and pulsing frequency to accentuate their effects on system performance. For similar reasons, velocity feedback is not used in this illustrative trial. A relatively large dead zone of ±0.25° is simulated to facilitate observation of the system’s natural response. Phase plane characteristics of PWM, relay, and saturating amplifier controlled systems for a step input are shown in Figures 6.10 through 6.12, respectively.

Each of the three controller configurations display identical response characteristics in the dead zone and saturation regions. This is anticipated since, in these regions, all three nonlinear controllers are characterized by either 0 or 150 volts of continuous output. It is interesting to note that the PWM model is unaffected by pulsing frequency in these regions. This is due to the system error being outside of the linear range of modulation and thus resulting in continuous modulation duty cycles of 0 or 100%. The linear region of operation is unique to the saturating amplifier and PWM controlled
Figure 6.10  Phase Plane / Typical PWM Controller
Figure 6.11 Phase Plane / Equivalent Relay Controller
<table>
<thead>
<tr>
<th>Type of Input</th>
<th>STEP</th>
<th>Start Time</th>
<th>Stop Time</th>
<th>KP</th>
<th>KV</th>
<th>Ess</th>
<th>GAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.00000</td>
<td>0.04000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>0.25000</td>
<td>0.50000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.12 Phase Plane / Equivalent Saturation Amplifier Controller
systems. Although the phase plane portraits representing these two configurations are defined identically, the relatively low pulsing frequency exaggerates the effects of the discontinuous PWM. Specifically, the pulsing period used in the response of Figure 6.10 is sufficiently large so as to cause undesirably long durations of forced and natural response for a determined duty cycle. The increased significance of the time dimension in the phase plane results in time averaging being a less valid approximation of actual performance. If the trajectory in the linear region of the PWM of Figure 6.10 was time averaged, it would describe the same response as the saturating amplifier shown in Figure 6.12. However, the instantaneous and erratic trajectory of Figure 6.10 follows the path of the relay of Figure 6.11 while being forced with 150 volts and, during the relaxed mode, the path of natural response of the dead zone. Since the continuous amplifier provides ideally smooth response behavior, a maximum value of pulsing frequency is desired. Frequency considerations are discussed in subsequent trials.

The relative "softness" of the PWM's settling behavior is due to its relatively low value of amplifier gain. Small error values receive inadequate gain to provide the stiff control typically desired in high performance
applications. Increasing amplifier gain approximates the stiff response of the relay shown in Figure 6.11.

The first design parameter to be independently examined in the phase plane was PWM pulse frequency. Although a maximum pulsing frequency of 10KHz was selected in Chapter V based on hardware limitations, the effect of pulse duration on system response is significant and deserves attention. Figures 6.13 through 6.16 illustrate the effects of varying $f_{PWM}$ with values of 1, 5, 10, & 20KHz. Amplifier gain remains small so as to expand the linear region where pulsing frequency affects the response. All observations made during the trial describe the simulated trajectory in the linear region of operation.

Figure 6.13 shows the unacceptable consequences of using an insufficient $f_{PWM}$. Poor correlation between portrait and trajectory characteristics implies the invalid time averaging is attributed to excessive pulse duration. The response is smoothed with higher pulsing frequencies. However, the difference between the 10 KHz and 20 KHz responses is negligible, indicating that very high frequencies are not required for the linear approximation of continuous saturation amplification to be valid. Therefore, 10 KHz was selected as optimum for PWM pulsing frequency.
Figure 6.13 Phase Plane Response of 1KHz PWM Controller
Figure 6.14 Phase Plane Response of 5KHz PWM Controller
Figure 6.15  Phase Plane Response of 10KHz PWM Controller
Figure 6.16  Phase Plane Response of 20KHz PWM Controller
Amplifier gain determines to a large extent the system's responsiveness. Figures 6.17 through 6.19 illustrate system behavior ranging from "soft" to "stiff". Relatively soft behavior shown in Figure 6.17 is due to the use of an amplifier gain of 0.1 and is characterized by a slow oscillatory transient. Conversely, stiff conditions are shown in Figure 6.19 and are attained using a $K_{PWM}$ of 10.0. The trajectory resembles the response of the relay shown in Figure 6.11 and is described by very rapid "chattering" of the motor about steady state. Finally, acceptable responsiveness is achieved using a gain of 3.0 as illustrated in Figure 6.18 and, therefore, was selected as the optimum value for gain.

A pulsing frequency of 10KHz and amplifier gain of 3 were independently selected by observing their effects on phase plane response behavior. However, frequency and gain exhibit coupled effects which result in limit cycle behavior. While limit cycles may not be eliminated solely through the selection of $K_{PWM}$ and $f_{PWM}$, they may be minimized. Steady state oscillations are small in amplitude for the parameter selection described. However, when gain is increased and frequency decreased, unacceptable magnification of limit cycle behavior is
Figure 6.17 Phase Plane Response of $K_{PWM} = .1$ PWM Controller
Figure 6.18 Phase Plane Response of $K_{PWM}=3$. PWM Controller
Figure 6.19 Phase Plane Response of $K_{PWM}=10$ PWM Controller
realized. Figure 6.20 illustrates the effect of increasing amplifier gain to 50 while maintaining the pulsing frequency at 10KHz. Figure 6.21 shows exaggerated effects achieved by reducing frequency to 1KHz with gain fixed at 3. The region shown by the circle is discussed below.

The extent of limit cycle behavior is dependent on the amount of electromotive energy provided to the motor during the delay associated with the PWM pulsing period. For example, consider the low $f_{PWM}$ trajectory of Figure 6.21. The delay observed in actual controller switching caused by the finite pulse duration which extends the state space forcing vector from one region into another. The delay highlighted by the circular region in Figure 6.21 may be graphically observed in Figure 6.22 by examining the pulse width modulator's time response in the the proximity of switching. Once a pulse's duty cycle is set, the controller is temporarily insensitized to operating conditions for the remainder of the pulse period. Interpretation of Figure 6.22 confirms that limit cycle behavior may be reduced by increasing pulsing frequency or decreasing gain.
Figure 6.20 Stable Limit Cycle Due to $K_{PWM}=50$.

Figure 6.21 Stable Limit Cycle Due to $f_{PWM}=1$KHz
CLOSED LOOP PERFORMANCE (PWM)
GAIN = 3.00    DEAD ZONE = 0.00

Figure 6.22 PWM Time Response Describing Switching Delay
The final phase of this preliminary design involves selection of feedback gain coefficients $K_p$ and $K_v$. For analytical simplicity, unity feedback ($K_p = 1$) is chosen. Velocity feedback is selected relative to $K_p$ and determines the extent of system damping. The feedback design performed in the phase plane does not incorporate dead zone in the controller for the sole purpose of enhancing the illustration of switching line behavior. Specifically, reticence, or "chattering", is not desired and might be undetected if dead zone is applied in this preliminary stage of the design.

Lines of discontinuity acquire a finite slope in the phase plane when velocity feedback is used. The slope is the feedback ratio, $-K_p/K_v$, and applies to all region boundaries. Trajectory interaction with the switching line provides the definition for damping criteria. For continuous regulators, sufficiently large velocity feedback results in overdamped behavior characterized by no transient oscillation. The system trajectory should approach, but not intersect, the switching line. Overdamping discontinuous controllers, such as the PWM, is characterized by excessive reticence and slow responsiveness. If velocity feedback gain is small enough, then the system trajectory will intersect the switching
line and yield an underdamped oscillatory transient. The intersection of the \( \dot{e} \)-axis by the trajectory defines transient overshoot. Critical damping is described by the condition where the trajectory coincides with the switching line. The phase plane provides a graphic method of selection of damping characteristics. It is observed that slope markers, which represent system response characteristics unique to specific operating conditions, are independent of \( K_v \). Furthermore, region #1 of Table IV describes the system's natural response found in dead zone operation. Alignment of the \( K_v \) dependent switching line (dashed line) with slope markers in the dead zone is shown in the phase plane portrait of Figure 6.23 and results in critical damping. The switching line slope may be increased by decreasing \( K_v \), resulting in underdamped conditions. Conversely, increasing \( K_v \) tends the switching line towards horizontal and yields overdamped behavior.

Figure 6.24 illustrates overdamped step response behavior attained with a velocity feedback gain coefficient of \( K_v=0.01 \). The trajectory is characterized by excessive "chattering" while it slowly traverses the switching line and is associated with large \( K_v \). Near critical damping is shown in Figure 6.25 where \( K_v=0.005 \) and is characterized by slow transient behavior along the switching line. This
Figure 6.23 Phase Plane Portrait Showing Natural Response
**Figure 6.24 Phase Plane / Overdamped Step Response**
PHASE PLANE CHARACTERISTICS IPULSE WIDTH MODULATORI

SUMMARY

Type of Input  STEP
Start Time     0.00000
Stop Time      0.04000
K_P            1.00000
K_V            0.00500
E_s            0.00000
GAIN           3.00000
PWM Freq (Hz)  10000.0
system configuration provides optimal position control in terms of energy requirements, but is not well suited to this high performance tracking application. Figure 6.26 illustrates underdamped response behavior associated with a velocity feedback gain coefficient of $K_v=0.0005$. The trajectory is oscillatory and exhibits overshoot, but provides highly responsive transient behavior. Characteristics observed in the phase plane of Figure 6.27 describe good step response performance of fin position for the linearly approximated system when $K_v=0.001$ and support the selection of this velocity feedback gain coefficient as an optimal design parameter value. The slightly underdamped response shows no overshoot and has a fast settling time. Figures 6.28 and 6.29 show agreement between the reduced (2nd) order and third order systems' step time response, thereby validating results attained via the phase plane.

The phase plane described by $e$ and $\dot{e}$ for step response conditions actually describes $-\theta_{\text{fin}}$ and $-\omega_{\text{fin}}$. The plane may readily be transformed into an energy plane, where potential energy is a function of $e$ and kinetic energy is a function of $\dot{e}$. The switching line, which passes through the steady state point describes conditions of locally maximized kinetic energy, whereas the crossing of the $e$-axis describes localized minimum kinetic energy. The
Figure 6.26 Phase Plane / Underdamped Step Response
Figure 6.27 Phase Plane / Preliminary Optimal Design Step

### SUMMARY

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Input</td>
<td>STEP</td>
</tr>
<tr>
<td>Start Time</td>
<td>0.00000</td>
</tr>
<tr>
<td>Stop Time</td>
<td>0.04000</td>
</tr>
<tr>
<td>KP</td>
<td>1.00000</td>
</tr>
<tr>
<td>KV</td>
<td>0.00100</td>
</tr>
<tr>
<td>Ess</td>
<td>0.00000</td>
</tr>
<tr>
<td>GAIN</td>
<td>3.00000</td>
</tr>
<tr>
<td>PWM Freq (Hz)</td>
<td>10000.0</td>
</tr>
</tbody>
</table>
Figure 6.28 Reduced Order Model Fin Position Step Response

Figure 6.29 Third Order Model Fin Position Step Response
degree of response stiffness of the controller is determined by the amount of electromotive energy it provides to the motor. Increased amplifier gain results in greater pulse duty cycles for given system conditions and, hence, greater electromotive energy transmission. The forced harmonic oscillation frequency also increases with amplifier gain and results in greater numbers of rotational reversals, a condition to be avoided whenever possible.

Ramp response analysis is also supported by phase plane methods and will be briefly discussed. The natural response component of the slope function (η) described by Equation 6.20 is dependent on the ramp slope, \( \dot{\tau} \). When \( \dot{\tau} \) is non-zero, the coordinates (e,\( \dot{e} \)) do not linearly transform to \((-\theta_{\text{fin}},-\omega_{\text{fin}}))\) and the energy plane correlation is not valid. Additionally, the phase plane is no longer symmetric about the e-axis. Consequently, the ramp response portrait is significantly different than that of the step response.

The underdamped controller whose step response was determined in Figure 6.27 was subjected to a ramp input of slope \( \dot{\tau}=3000 \text{ deg/sec} \). The phase plane trajectory is illustrated in Figure 6.30, characterized by similarly acceptable response behavior. Steady state error is analytically determined to be \( e_{\text{ss}}=3.14 \) by equation 6.21. Figures 6.31 and 6.32 provide the ramp time responses of
Figure 6.30 Phase Plane / Preliminary Optimal Design Ramp Response
Figure 6.31  Reduced Order Model Fin Position Ramp Response

Figure 6.32  Third Order Model Fin Position Ramp Response
the second and third order lumped parameter models. Transient behavior is rapid and steady state tracking accuracy of this demanding input is tolerable.

Of particular interest is the phase plane representation of the system's dynamic limitations which describe the motor's torque capabilities. Figure 6.33 illustrates the case where the system's maximum position tracking rate precludes stable convergence when a ramp input of \( \dot{r} = 10000 \text{ deg/sec} \) is used. At steady state, the trajectory becomes asymptotic to the horizontal line given by Equation 6.24. Steady state error is infinite, indicating ramp response instability.

\[
\dot{e}_{ss} = \dot{r} - \omega_{fss}(0.L.) = 10^4 - 7277 = 2623 \text{ fin-deg sec}^{-1} \quad (6.24)
\]

The phase plane has been used extensively in this chapter as a method for gaining insight of the controller's dynamics and for selecting preliminary design parameters. However, two important final design considerations remain unaddressed and are the focus of attention in Chapter VII. Proper design of this electromechanical system requires compromise between system performance and reliability of electronic components. Additionally, thorough simulation
Figure 6.33 Phase Plane / Unstable Ramp Response

<table>
<thead>
<tr>
<th>SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Input</td>
</tr>
<tr>
<td>Start Time</td>
</tr>
<tr>
<td>Stop Time</td>
</tr>
<tr>
<td>KP</td>
</tr>
<tr>
<td>KV</td>
</tr>
<tr>
<td>Ess</td>
</tr>
<tr>
<td>GAIN</td>
</tr>
<tr>
<td>PWM Freq (Hz)</td>
</tr>
</tbody>
</table>
of the final design must be accomplished using the detailed model of the brushless dc motor.
VII. PARAMETER OPTIMIZATION BY SIMULATION METHOD

A. BACKGROUND

Proper design of a position controller for an electronically commutated cruise missile fin actuator demands consideration of the system's mechanical response characteristics and the associated transient environment imposed on its solid state components. Preliminary design based solely on mechanical performance was presented in the preceding chapter. Electrical transient behavior of the system was developed and studied in Chapters III and IV. The central theme of this chapter is investigation of electromechanical design considerations, where compromise between mechanical performance and electronic reliability is required. The application of pulse width modulation to position control, studied in Chapters V and VI, was shown to provide the broad performance range necessary to accommodate electrical-mechanical design compromises and, therefore, was selected as the means of fin actuation control for this design.

Design parameters independently determined in preceding chapters are summarized in Table V and yield good electrical and mechanical performance properties.
These values serve as the starting point in the final design.

TABLE V
SUMMARY OF PRELIMINARY DESIGN PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM Amplifier Gain ($K_{PWM}$)</td>
<td>3.0</td>
</tr>
<tr>
<td>PWM Pulsing Frequency ($f_{PWM}$)</td>
<td>10KHz</td>
</tr>
<tr>
<td>PWM Controller Dead Zone (DBAND)</td>
<td>0.0°</td>
</tr>
<tr>
<td>Velocity Feedback Gain Coefficient ($K_v$)</td>
<td>0.001</td>
</tr>
<tr>
<td>Position Feedback Gain Coefficient ($K_p$)</td>
<td>1.0</td>
</tr>
<tr>
<td>Bias Threshold Voltage Divider %</td>
<td>50%</td>
</tr>
<tr>
<td>Commutation Offset (THADV)</td>
<td>0.0°</td>
</tr>
<tr>
<td>Plugging Delay (TDPLUG)</td>
<td>500 μsec</td>
</tr>
<tr>
<td>Decay Rate Resistance (RDADJ)</td>
<td>3 Ω</td>
</tr>
</tbody>
</table>

System performance design conducted in Chapter VI was based primarily on small scale step response analysis. Using a 3° fin perturbation, phase plane trajectories of the reduced order model were characterized by slightly underdamped behavior. No overshoot and very little transient oscillation were noted. A relatively large scale
ramp response (3000°/sec) was also studied using the same design criteria, and was observed to behave very acceptably. The validity of this design based on lumped parameter modeling is initially ascertained through simulation using the detailed brushless dc motor model listed in Appendix A. Subsequent to fine adjustment of design parameters, large scale step, ramp, and sinusoidal response analyses are performed to determine the controller's robustness.

B. SIMULATION

Validation of the small scale step response analysis performed in the phase plane is readily accomplished through simulation using the detailed model listed in Appendix A. While small scale step analysis does not accurately represent the tracking dynamics encountered in missile flight control, it does provide a general and convenient standard for selection of controller design parameters.

1. Small Scale Step Response

Simulation using the preliminary design parameters summarized in Table V duplicate the system configuration selected in Chapter VI. In addition to system response verification, observation of the corresponding electrical transient behavior is afforded. Figures 7.1 through 7.4 illustrate the electromechanical performance of the
Figure 7.1 Fin Position Step Response (Preliminary Design)
Figure 7.2 Motor Current Step Response (Preliminary Design)
Figure 7.3 Node 0 Voltage Step Response (Preliminary Design)
Figure 7.4 Motor Torque Step Response (Preliminary Design)
preliminary design when simulated with the brushless dc motor model. Figure 7.1 depicts the fin position response to a 3° step input. Of particular interest is the presence of a 25% peak overshoot, which was not predicted by the lumped parameter model simulation. Associated with the oscillatory transient are a relatively large current demand exhibited in Figure 7.2 and a 400 volt spike observed in the center point trajectory of Figure 7.3. Torque response characteristics are given in Figure 7.4 and vividly display the relatively long duration of natural response due to the use of a plugging delay time (TDPLUG) of 500μsec. System relaxation of such duration invites unacceptably large degrees of ballistic overshoot. However, plugging delay of some extent is required due to the inevitable requirement for current reversal associated with forced braking of the motor. The closed loop transient analysis performed in Chapter III assumes maximum stiffness in the controller's operation and employs the characteristics of an ideal relay to simulate such behavior. When maximum current flow is reversed, there is an understandable need for dissipation of excess coil energy. However, a soft controller was determined to be best suited in the preliminary design and pulse width modulation using an amplifier gain (KPWM) of 3 was selected. The resulting buffering action provided by the controller eliminates the need for a lengthy current decay time during plugging. Figures 7.5 through 7.8
illustrate the small scale step response behavior using a reduced TDPLUG of 10 µsec. No overshoot is observed in the fin position step response shown in Figure 7.5 and system behavior is virtually identical with the results attained in lumped parameter modeling. Hence, the linear approximations required by phase plane design methods are validated and shown to be an effective means of predicting system performance. Figure 7.6 indicates substantial reduction in current demand when the plugging delay is decreased since system recovery from ballistic overshoot is not required. A peak voltage surge of only 100 volts is experienced at the center connection and is shown in Figure 7.7. Figure 7.8 illustrates the absence of any detectable period of natural motor torque response associated with plugging delay.

The erratic current response behavior illustrated in Figure 7.6 between .003 and .004 seconds is attributed to the periodic forcing nature of the PWM controller as the fin position settles. Near steady state conditions, the PWM pulse duty cycle is small and does not receive effective filtering by the system's electrical time constant. Therefore, the electrically developed torque shown in Figure 7.8 is also affected by the pulsation. It is observed, however, that the dominant mechanical low pass characteristics of the motor filters the oscillation and results in the smooth position response shown in
Figure 7.5 Fin Position Step Response (TDPLUG=10 μsec)
Figure 7.6 Motor Current Step Response (TDPLUG=10 μsec)
Figure 7.7 Node 0 Voltage Response (TDPLUG=10 μsec)
BRUSHLESS DC MOTOR TORQUE STEP RESPONSE

TDPLUG = 0.00001 SECONDS
Figure 7.5. The inclusion of a ±.5° dead zone decreases the extent of this forced oscillation and is described in Figures 7.9 through 7.11. Figure 7.9 portrays the fin position step response with the inclusion of dead zone. Steady state accuracy is compromised in a manner proportionate to the extent of dead zone employed. The fin position passes through its intended steady state position of 0° and settles at .5°. Once again, the relatively large region allowed for natural response supports ballistic overshoot of the fin position and causes additional undesired forced braking action. Because the motor undergoes commutation switching at 0°, the commutation logic is reverse sequenced and results in current reversal. Figure 7.10 indicates reduced susceptibility of the motor current to the pulsing action of the PWM due to dead zone, but is also characterized by rapid decay of current associated with the interaction of plugging and commutation switching. The rapid change in current flow is responsible for the 180 volt spike observed in the center connection voltage response of Figure 7.11. Analysis of small scale step response indicates that inclusion of dead zone not.
Figure 7.9  Fin Position Step Response (Dead Zone = ±.5°)
Figure 7.10 Motor Current Step Response (Dead Zone = ±.5°)
Figure 7.11 Node O Voltage Response (Dead Zone = ±.5°)
only degrades system accuracy, but also accentuates electrical transients due to the increased need for forced braking.

2. Large Scale Step Response

Stability of nonlinear systems often is largely dependent on the initial conditions imposed. Simulation of a large scale step response, for example, might be expected to display different response characteristics of the brushless dc motor fin position controller than have been observed using a small step. In application, it is highly unlikely that the missile command guidance would command a 45° fin step. However, because of its excessive nature, such a simulation was conducted for purposes of evaluating benchmark performance. Figures 7.12 through 7.14 illustrate the electromechanical performance of the brushless dc motor subjected to a 45° step input. The fin position response shown in Figure 7.12 indicates a 15% peak overshoot and a comparatively large settling time of 0.02 seconds. Both of these response characteristics are highly acceptable and indicate performance robustness. Figure 7.13 illustrates motor current large scale step response behavior. Although average current demand is relatively high due to the nature of the input, the current response is very well behaved. Figure 7.14 shows reasonable Node 0 voltage transients in the motor, indicative of proper commutation operation of
Figure 7.12  Fin Position Large Scale Step Response
Figure 7.13 Motor Current Large Scale Step Response

Brushless DC Motor Current Step Response

Large step amplitude = 45 degrees

TIME (seconds) $\times 10^{-1}$
Figure 7.14 Node 0 Voltage Large Scale Step Response
the power conditioner. A maximum center connection voltage surge of 120 volts was observed.

3. **Ramp Response**

Since the cruise missile fin actuator behaves as a tracking device, a dynamic range of input values permit more realistic analysis of the controller. A stiff ramp input signal of slope 3000 deg/sec is used to attain the dynamic response behavior of the actuator. Similar analysis was previously conducted in the phase plane using the lumped parameter model. Although ramp response analysis specifically evaluates the positioning device's ability to perform as a constant velocity device, it provides rudimentary dynamic response behavior unavailable in step response analysis. Figures 7.15 through 7.19 display the tracker's dynamic performance characteristics simulated using a relatively demanding ramp input of 3000deg/sec. Tracking performance is illustrated in Figure 7.15 and very similar to the behavior observed using lumped parameter modeling. Steady state error, as defined in Chapter VI, is observed and is considered to be acceptable for this design application. Although the system is underdamped, transient oscillation is negligible and forced braking is kept to a minimum. Figure 7.16 gives the motor current ramp response and is characterized by a relatively high current demand. Additionally, significant harmonic
Figure 7.15  Fin Position Ramp Response
Figure 7.16 Motor Current Ramp Response
Figure 7.17 Motor Torque Ramp Response
Figure 7.18 Node 0 Voltage Ramp Response
Figure 7.19 Transistor #5 Voltage Ramp Response
oscillation caused by PWM pulsing is exhibited near constant velocity. PWM pulsing is more clearly shown in the torque response of Figure 7.17, where transient and steady state oscillation also are exhibited. A center connection peak voltage surge of 150 volts is observed in Figure 7.18. The voltage response of Power Transistor #5 is illustrated in Figure 7.19. The pulsing effects of the PWM operating at low duty cycle near steady state are readily observed as moderate spiking across the transistor. Although this ragged forced response could be smoothed with use of higher pulsing frequencies, hardware constraints preclude such implementation. The corresponding voltage responses shown in Figures 7.18 and 7.19 are characterized by high frequency voltage transients of moderate magnitude. Therefore, despite the undesirable pulsing effects of the PWM, the ramp response characteristics are determined to be acceptable.

4. Sinusoidal Response

The final and most realistic simulation performed evaluates the dynamic tracking performance of the fin actuator by subjecting the controller to a sinusoidal input. The ability of the controller to track sinusoidally varying position commands demonstrates true tracking robustness. To simulate typical cruise missile command guidance flight correction commands, a sine wave of
amplitude $3^\circ$ and frequency of 100Hz was used as a reference input. Figures 7.20 through 7.23 illustrate electromechanical sine wave response characteristics of the fin actuator. Good tracking characteristics are exhibited in the fin position response of Figure 7.20. Although steady state error varies with the instantaneous slope of the input, it is acceptably small and is approximated at $0.3^\circ$ at the sine's point of inflection. Figure 7.21 gives the motor current sine wave response and indicates considerable pulsing activity at near steady state conditions. These pulses are realized as voltage spikes at solid state components and the representative conditions at power transistor #5 are illustrated in Figure 7.22. However, the spikes are sufficiently small (170 volts) so as not to degrade solid state reliability. Figure 7.23 shows the center point trajectory when the controller is subjected to small-amplitude sinusoidal inputs. A peak voltage surge of 15 volts is highly acceptable.
Figure 7.20 Fin Position Sinusoidal Response
Figure 7.21 Motor Current Sinusoidal Response
Figure 7.22 Transistor #5 Voltage Sinusoidal Response
Figure 7.23 Node 0 Voltage Sinusoidal Response
C. FINAL DESIGN

It has been demonstrated through simulation that the position controller final design parameters specified in Table VI provide robust operational performance. Demanding mechanical performance is achieved with an electronically sensitive brushless dc motor through the adaption of pulse width modulation. The transient waveshaping incorporated in this design appeared to significantly modify the electrical response characteristics when independently studied. However, the coupled electrical-mechanical design performed in this chapter diminishes the impact of waveshaping and relies most heavily on the soft response characteristics of the PWM. Oscillation due to low duty cycle pulse width modulation is apparent during near-steady state conditions, but its effects are filtered by the low pass characteristics of the brushless dc motor.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM Amplifier Gain ($K_{PWM}$)</td>
<td>3.0</td>
</tr>
<tr>
<td>PWM Pulsing Frequency ($f_{PWM}$)</td>
<td>10KHz</td>
</tr>
<tr>
<td>PWM Controller Dead Zone (DBAND)</td>
<td>0.0°</td>
</tr>
<tr>
<td>Velocity Feedback Gain Coefficient ($K_v$)</td>
<td>0.001</td>
</tr>
<tr>
<td>Position Feedback Gain Coefficient ($K_p$)</td>
<td>1.0</td>
</tr>
<tr>
<td>Bias Threshold Voltage Divider %</td>
<td>50%</td>
</tr>
<tr>
<td>Commutation Offset (THADV)</td>
<td>0.0°</td>
</tr>
<tr>
<td>Plugging Delay (TDPLUG)</td>
<td>10 μsec</td>
</tr>
<tr>
<td>Decay Rate Resistance (RDADJ)</td>
<td>3 Ω</td>
</tr>
</tbody>
</table>
VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

It has been demonstrated through computer simulation that the brushless dc motor has viable application in position control. The electronically demanding environment associated with forced braking and bi-directional motor operation requires that considerable attention be given to the power conditioner's electrical transient response characteristics. The central theme presented in this study involves the design compromises required to exploit the intrinsic high performance of a brushless dc motor without sacrificing operational reliability of its solid state components. This study advances an on-going comprehensive development effort of cruise missile fin control using electromechanical actuation.

Reduced peak voltage conditions associated with commutation switching were attained through transient current waveshaping techniques involving modification of power conditioner circuitry. Pulse width modulation in closed loop position control was observed to provide electrically advantageous response buffering during rotational reversals of the motor.

System performance was initially evaluated in the phase plane and preliminary design parameters based on a reduced
order lumped parameter model were identified. Composite analysis of the electromechanical actuator was conducted through computer simulation using the detailed model of the brushless dc motor listed in Appendix A. Following minor power conditioning parameter adjustments, simulated results were highly consistent with system performance observed using linear approximation methods and electrical transient behavior was observed to be very satisfactory. The controller was found to be functionally robust based on response evaluation using small and large step, ramp, and sinusoidal inputs.

The four pole brushless dc motor modeled in this study is based on typical motor specifications and limited physical benchmark measurements of this motor conducted at NWC, China Lake. Consequently, validation of the model is limited to the extent of available performance data. Ongoing cruise missile fin actuator design efforts using brushless dc motors at NWC, China Lake promise a more complete and reliable performance data base for future modeling improvements. The interactive nature of the simulation program permits easy access to motor and controller parameters and facilitates future parameter modification to accommodate updated performance measurements. Validation of the computer simulation was performed not only with comparison of predicted and physically observed behavior, but also with analytical
methods. Lumped parameter modeling and state space solutions determined in Chapter VI were used to accurately predict the detailed simulation results of Chapter VII.

B. RECOMMENDATIONS FOR FURTHER STUDY

All analyses performed in this study were based on linear fin actuator load conditions. Mechanical linkage design consideration with inclusion of loading and power transmission effects, such as modeled in Wright's thesis [Ref. 12], should be investigated for torsional and stall torque operating conditions.

The numerical accuracy of all computer simulations conducted in this work is limited by the nature of small step trapezoidal integration and should be considered as an area for future improvement. Since convergence criteria is constantly monitored, modification to the integration method may be readily evaluated through observation of numerical convergence behavior.

A study of power dissipation and heat sinking requirements of solid state components is warranted based on the high current demands observed during simulation of position control.

Feasibility of electrical power regeneration during dynamic braking action of the motor was suggested and deserves more extensive evaluation in future studies. Because of the anticipated lengthy duration of missile
flight control, adaption of such an energy conservation measure would reduce the demand on the missile power supply and conceivably support size and weight reduction.

Improved modeling of diode response characteristics will enhance the simulation of freewheeling action encountered during commutation switching. Presently, diodes are modeled as either conducting or not conducting, depending on bias conditions. The corresponding diode resistance characteristics, therefore, are discontinuously modeled and require artificial programming techniques to simulate their physical behavior.
APPENDIX A

BRUSHLESS DC MOTOR SIMULATION PROGRAM

SNOfloatcalls
SNOdebug
C ... MAIN PROGRAM ...
C
C | ROSSITTO, VS THESIS PROF GERBA 03/26/87 |
C | ___________________BRUSHLESS_DC_MOTOR______________________ |
C
C
C

SYSTEM REQUIREMENTS
C
C
C This simulation is written for compilation using MS FORTRAN77 V3.31.
C Three programs: BLDCM1.FOR, BLDCM2.FOR, & BLDCM3.FOR must be
C compiled and linked. This version uses PLOTWORKS PLOT88 for graphic
C output support; the library must be included during linking.
C
C - Hardware Reqmts --> INTEL 8088/8086 family (PC/XT)
C or INTEL 80286 family (AT)
C ({ INTEL 8087/80287 math coprocessor highly recommended})
C - Memory Reqmts --> 640K (RAM resident utility programs may have
C to be stripped.)
C - Storage --> Hard Disk preferred, Minimum 2 Floppy Drive System
C For use with floppy drives, virtual memory scheme
C requires 360KB data disk. The selection of VM default
C drive is menu selectable.
C - DOS Configuration --> CONFIG.SYS file must contain:
C    device=ANSI.SYS
C    files=15
C - Support Files --> BLDCM.INP (formatted input data which is
C    accessible during execution of program)
C    BLDCM1.VIR |
C    BLDCM2.VIR | These 4 files are generated by
C    BLDCM3.VIR | the simulation program and are
C    BLDCM4.VIR | used as virtual memory.
C
C
C PROGRAM DESCRIPTION
C
C This program simulates the detailed operation of a bipolar,
C three phase brushless dc motor. Console interaction by the user
C provides the ability to simulate under a broad range of operating
C parameters as well as initial conditions. Since the simulation
C is intended to be used primarily for design purposes, repetitive
C iterations with saved data are possible.
C
C The method of simulation used in this program emulates
C processing through the use of two nested systems of non-linear
differential equations. The inner loop serves to describe the
electrical system response, while the outer loop describes the
electro-mechanical system response. Both systems of non-linear
differential equations are solved using Newton-like iterative
convergence schemes. The inner loop is numerically stiff in that
the non-linear nature of the diode equivalent resistance allows
numerical convergence to two solutions when the diode cutoff
threshold is approached. This is dealt with by using a fixed,
but relatively small step size and a cautious modified Newton’s
method of solution.

Trapezoidal integration is used for computational speed.
Comparison to results attained using Runge-Kutta methods on IBM-
370 based CSMP indicate good accuracy. The first order
differential equation solver, TCONST, is an algebraic solver and
provides a rapid means of solving an otherwise iterative problem.

Variable Descriptors are explained throughout the program and,
therefore, will not be described here.

IMPLICIT REAL*4 (A-Z)
COMMON BEGTIM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,PLEN,FLEN,
+XTITLE,YTITLE,PTITLE,PTITLE
REAL*4 XTIME(1010)
INTEGER*2 NTIM,NTERMS,IOPORT,MODEL,NUMIT,NUMIN,XLEN,YLEN,N,
+NTIM1,NTIM11,NTIM12,NTIM13,NTIM14,
+NTIM15,NTIM12,NTIM13,NTIM14,NTIM15,NTIM16,
+ELEMT,SIM2FL,POSIT,DIR,
+NTIM01,NTIM02,NTIM03,NTIM04,NTIM05,NTIM06,HALF,
+NTIM07,NTIM08,PMODEL,NCTR,NPTS,AHALF,BHALF,CHALF,
+TRIG1,TRIG2,TRIG3,TRIG4,TRIG5,TRIG6,DIRM1,PLUG,
logical*2 TOGGLE
character*3 DISOPT,DRIVE,DRIVE1
character*6 ANSI1,ANS11,ANS21,ANS27,PRTSEL,NONLIN,TYPE,TYPE1
character*25 XTITLE,YTITLE,NLCHAR
character*51 PRTCHR,PTITLE,PTITLE

... Introductory Page (1 time good deal!)
call CLRSCR
write(*,4)
PAUSE

*** Open/Read/Close input data file ***
13 open(2,FILE='BLDCM.INP',STATUS='OLD',ACCESS='SEQUENTIAL')
read(2,1000) PMODEL,FRICHER,RIN,ROUT,RADJ
read(2,1020) BEGTIM,FINTIM,PTSPLT,MAXITS,SIM2PL
read(2,1022) KPWM,KV,E0,EDOT0,TTIME,XORG,YORG
read(2,1024) RS,RSAT,JL,BL,KTM
read(2,1026) JM,RA,LA,KEM,THADV,TDPLUG
read(2,1028) BM,KP,PERIOD,DBAND,KA
read(2,1030) TYPE,RCUT,STPMAG,RSLOPE
read(2,1032) SINAMP,SINFREQ,SINPHA,DRIVE

179
close(2,STATUS=’KEEP’)

C

******************************************************************************

C

*** DISPLAY MAIN MENU SELECTIONS ***

******************************************************************************

C

1 call CLRSCR
     call GOTOXY(8,1)
     write(*,5)
     read(*,’(A)’) ANS1

C ... Hardware Options ...
      if ( (ANS1 .eq. ’h’) .or. (ANS1 .eq. ’H’)) then

C

101 call CLRSCR
     call GOTOXY(17,24)
     write(*,‘*** CURRENT PRINTER SELECTION ***‘)
     call GOTOXY(20,20)
     write(*,‘*’ ) PRTCHR
     call GOTOXYU(1,1)
     write(*,105)
     read(*,’(A)’) ANS1

C

C ... Printer Options ...
      if ( (ANS1 .eq. ’p’) .or. (ANS1 .eq. ’P’)) then
          call PRNOPT(PMODEL,PRTSEL,PRTCHR)
          go to 101

C

C ... Quit the Hardware Menu ...
      elseif((ANS1 .eq. ’q’) .or. (ANS1 .eq. ’Q’)) then
          go to 1
      else
          go to 101
      endif

C

C ... Motor Parameters ...
      elseif ((ANS1 .eq. ’m’) .or. (ANS1 .eq. ’M’)) then

C

call MOTPAR(KTM,KEM,RIN,ROUT,RA,LA,EM,SL,JM,JL,
+ RSAT,RCUT,TTIME,RS,TADV,IDPLUG)
      go to 1

C

C ... NON-LINEAR ELEMENT SELECTION MENU ...

C

elseif ((ANS1 .eq. ’n’) .or. (ANS1 .eq. ’N’)) then

14 call CLRSCR
     call GOTOXY(21,1)
     write(*,272) NLCHAR
     call GOTOXY(1,1)
     write(*,270) KV,KP
     read(*,’(A)’) ANS27
C ... RELAY AS NON-LINEAR ELEMENT ...
   if ((ANS27 .eq. 'r') .or. (ANS27 .eq. 'R')) then
      ELEMNT=1
      NLCHAR='RELAY'
   endif
   call CLRSCR
   NONLIN='R'
   write(*,290) DBAND
   read(*,'(A)') ANS21
   if (ANS21 .eq. 'DBAND') then
      call GOTOXY(24,1)
      write(*,'(A)')'Enter a REAL value for DBAND-->
      read(*,*) DBAND
      go to 291
   elseif ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then
      go to 14
   else
      go to 291
   endif
   elseif ((ANS27 .eq. 'KV') .or. (ANS27 .eq. 'kv')) then
      call GOTOXY(24,1)
      181
write(*,'(A)') Enter a REAL value for KV -->
read(*,*) KV
go to 14

C
elseif ((ANS27 .eq. 'KP') .or. (ANS27 .eq. 'kp')) then
  call GOTOXY(24,1)
  write(*,'(A)') Enter a REAL value for KP -->
  read(*,*) KP
go to 14
C
elseif ((ANS27 .eq. 'q') .or. (ANS27 .eq. 'Q')) then
  go to 1
else
  go to 14
endif
C
... Simulation Options ...
elseif ((ANS1 .eq. 'o') .or. (ANS1 .eq. 'O')) then
C
C 202 call CLRSCR
C
DELTIM=(FINTIM-BEGTIM)/(PTSPLT*SIM2PL)
if (FINTIM/DELTIM .gt. MAXITS) DELTIM = FINTIM/MAXITS
NTERMS=IFIX(FINTIM/DELTIM)+1
C
write(*,240) BEGTIM,FINTIM,PTSPLT,MAXITS,SIM2PL,EO,EDOTO,+
  XORG,YORG,DRIVE,DELTIM,NTERMS
read(*,'(A)') ANS21
C
if (ANS21 .eq. 'BEGTIM') then
  call GOTOXY(24,1)
  write(*,'(A)') Enter a REAL value for BEGTIM -->
  read(*,*) BEGTIM
go to 202
elseif (ANS21 .eq. 'FINTIM') then
  call GOTOXY(24,1)
  write(*,'(A)') Enter a REAL value for FINTIM -->
  read(*,*) FTIM
  if (FTIME .le. BEGTIM) then
    call CLRSCR
    call GOTOXY(10,10)
    write(*,*) 'INVALID FINTIM ... FINTIM must be > BEGTIM'
go to 202
  endif
  FINTIM=FTIME
go to 202
elseif (ANS21 .eq. 'PTSPLT') then
  call GOTOXY(24,1)
  write(*,'(A)') Enter a REAL value for PTSPLT -->
  read(*,*) PTSPLT
go to 202

182
elseif (ANS21 .eq. 'MAXITS') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for MAXITS --> '
  read(*,*) MAXITS
  go to 202
elseif (ANS21 .eq. 'SIM2PL') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter an INTEGER value for SIM2PL -->'
  read(*,*) SIM2PL
  go to 202
elseif (ANS21 .eq. 'E0') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for E0 (degrees) -->'
  read(*,*) E0
  go to 202
elseif (ANS21 .eq. 'EDOT0') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for EDOT0 (deg/sec) -->'
  read(*,*) EDOT0
  go to 202
elseif (ANS21 .eq. 'XORG') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for XORG -->'
  read(*,*) XORG
  go to 202
elseif (ANS21 .eq. 'YORG') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for YORG -->'
  read(*,*) YORG
  go to 202
elseif (ANS21 .eq. 'DRIVE') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter Drive (A,B,C) for data storage -->'
  read(*,'(A)') DRIVE1
  if ((DRIVE1 .eq. 'A') .or. (DRIVE1 .eq. 'a')) then
    DRIVE='A'
  elseif ((DRIVE1 .eq. 'B') .or. (DRIVE1 .eq. 'b')) then
    DRIVE='B'
  elseif ((DRIVE1 .eq. 'C') .or. (DRIVE1 .eq. 'c')) then
    DRIVE='C'
  else
    call CLRSRCR
    call GOTOXY(10,20)
    write(*,'(A)') 'Invalid Drive Specified - Enter A, B, or C'
    PAUSE
  endif
  go to 202
C ... Quit Simulation Options Menu ...
elseif (ANS21 .eq. 'Q') then
  go to 1
else
go to 202
end if

C

... Command Input Selection ...

elseif ((ANS1 .eq. 'c') .or. (ANS1 .eq. 'C')) then

203 call CLRSCR

write(*,245) TYPE,STPMAG,RSLOPE,SINAMP,SINFRQ,SINPHA
read(*,'(A)') ANS21

if (ANS21 .eq. 'TYPE') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a STEP, RAMP, or SINE--'
  read(*,'(A)') TYPE1
  if ((TYPE1 .ne. 'STEP') .and. (TYPE1 .ne. 'RAMP') .and.
      (TYPE1 .ne. 'SINE')) then
    call CLRSCR
    call GOTOXY(10,10)
    write(*,*) 'Invalid Selection !!!!'
    call GOTOXY(20,1)
    PAUSE
    go to 203
  else
    TYPE=TYPE1
  endif

  go to 203
elseif (ANS21 .eq. 'STPMAG') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for STPMAG--'
  read(*,*) STPMAG
  go to 203
elseif (ANS21 .eq. 'RSLOPE') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for RSLOPE--'
  read(*,*) RSLOPE
  go to 203
elseif (ANS21 .eq. 'SINAMP') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for SINAMP--'
  read(*,*) SINAMP
  go to 203
elseif (ANS21 .eq. 'SINFRQ') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for SINFRQ--'
  read(*,*) SINFRQ
  go to 203
elseif (ANS21 .eq. 'SINPHA') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for SINPHA--'

read(*,*) SINPHA
   go to 203
C ... Quit Simulation Options Menu ...
   elseif (ANS21 .eq. 'Q') then
      go to 1
   else
      go to 203
   endif
C ... Save Options to File ...
   elseif ((ANS1 .eq. 's') .or. (ANS1 .eq. 'S')) then

******************************************************************************
*** OPEN/WRITE/CLOSE INPUT DATA FILE ***
******************************************************************************

open(2, FILE='BLDCM.INP', STATUS='OLD', ACCESS='SEQUENTIAL')
write(2,1000) PMODEL, PRTCIR, RIN, ROUT, RDAD
write(2,1020) BEGTIM, FINTIM, FTSFLT, MAXITS, SIM2PL
write(2,1022) KFMM, KV, E0, EDTO, TTIME, XORG, YORG
write(2,1024) RS, RSAT, JL, BL, KTM
write(2,1026) JM, RA, LA, KRM, THADV, TDPLUG
write(2,1028) BM, KP, PERIOD, DBAND, KA
write(2,1030) TYPE, RCUT, STPMAG, RSLOPE
write(2,1032) SINAMP, SINFRQ, SINPHA, DRIVE
close(2, STATUS='KEEP')
C
   go to 1
C
C ... Run the Program ...
   elseif ((ANS1 .eq. 'r') .or. (ANS1 .eq. 'R')) then
C
C ... Quit the Program ...
   elseif ((ANS1 .eq. 'q') .or. (ANS1 .eq. 'Q')) then
      stop
   else
      go to 1
   endif
C
C ... Open an output data file ...
   continue
   if (DRIVE .eq. 'A') then
      OPEN(4, FILE='a:BLDCM1.VIR', STATUS='NEW')
      OPEN(5, FILE='a:BLDCM2.VIR', STATUS='NEW')
      OPEN(6, FILE='a:BLDCM3.VIR', STATUS='NEW')
      OPEN(7, FILE='a:BLDCM4.VIR', STATUS='NEW')
   elseif (DRIVE .eq. 'B') then
      OPEN(4, FILE='b:BLDCM1.VIR', STATUS='NEW')
      OPEN(5, FILE='b:BLDCM2.VIR', STATUS='NEW')
      OPEN(6, FILE='b:BLDCM3.VIR', STATUS='NEW')
      OPEN(7, FILE='b:BLDCM4.VIR', STATUS='NEW')
elseif (DRIVE .eq. 'C') then
    OPEN(4,FILE='c:BLDCM1.VIR',STATUS='NEW')
    OPEN(5,FILE='c:BLDCM2.VIR',STATUS='NEW')
    OPEN(6,FILE='c:BLDCM3.VIR',STATUS='NEW')
    OPEN(7,FILE='c:BLDCM4.VIR',STATUS='NEW')
endif

C *** Listing of constants ***
C
PI = 3.14159265
KADJ=0.63
KK3 = 3.590
TN2=.1
N=30.0
TCIMC=0.01
TLL=0.0
VSAT=.4
REVTIM=0.1
VN1=0.
VN2=0.
VSGREF=30.
KINT=10000.
KRAMP=18000.
VSG=0.
VINHF1=75.0
REDRAT=.1
C
C Initial Conditions
TRIG1=0
TRIG2=0
TRIG3=0
TRIG4=0
TRIG5=0
TRIG6=0
WM=0.
THETA=0.
PTHETA=0.0
THETA1=0.
IMA=0.0
IMB=0.0
IMC=0.0
IMAB=0.0
IMBC=0.0
IMCA=0.0
NCTR=0
NPTS=0
NTIM1=0
NTIM01=0
NTIM02=0
NTIM03=0
NTIM04=0
NTIM05=0
NTIM06=0
NTIM11=0
NTIM12=0
NTIM13=0
NTIM14=0
NTIM21=0
NTIM22=0
NTIM01=0
NTIM02=0
NTIM03=0
NTIM04=0
NTIM05=0
NTIM06=0
NTIM32=0
NTIM33=0
NTIM34=0
NTIM35=0
ICQ1=.75
ICQ2=.75
ICQ3=.75
ICQ4=.75
ICQ5=.75
ICQ6=.75
ICIMAB=0.
ICIMBC=0.
ICIMCA=0.
ICTMF=0.
ICM4=0.
X1=0.0
F1=0.0
F2=0.0
X3=0.0
ZA3=0.0
ZB3=0.0
ZC3=0.0
TSTART=0.0
TOGGLE.=true.
VIN=0.
PREV5=0.
CURV5=0.
HOLD1=0.
PREV2=0.
PREV2=0.
PREV3=0.
YY2=0.
YY3=0.
YY4=0.
YDM12=0.
YDM13=0.
YDM14=0.

YD2=0.
YD3=0.
YD4=0.
YDDM12=0.
YDDM13=0.
YDDM14=0.
YDD2=0.
YDD3=0.
YDD4=0.
PREV7=0.
CURV7=0.
HOLD3=0.
PLUG=0
TFPLUG=0.
DIRM1=0

C

C RELATIONSHIPS

KT=KTM*KADJ
KB=KBM*KADJ
BLP=BL/(N**2)
JLP=JL/(N**2)
J=JM+JLP
B=BM+BLP
A1=LA/RA
A2=J/B
A3=LA/(RA+RSAT)
RS1=RS/2.
RS2=RS/2.
VINIC=15.0/KINT
VINF=VINF1

C ... Initial conditions for THETA(fin, degrees) and Omega(fin, degrees).

EORAD=E0*PI/(180.*REDRAT)
EDOT0R=EDOT0*PI/(180.*REDRAT)
if (KP .ne. 0.) then
  WM0=-EDOTOR/KP
  THETA0=EORAD/KP
elseif (KP .eq. 0.) then
  WM0=-EDOTOR
  THETA0=-EORAD
endif

WM=WM0
X3=THETA0

C

C ... Output Simulation Options to Output Data File ...

C

C *************************************

C ***** Time Generator ... 0-50ms *****

C *************************************

DELTIM=(FINTIM-BEGTIM)/(PTSFLT*SIM2PL)
if (FINTIM/DELTIM .gt. MAXITS) DELTIM =FINTIM/MAXITS
NTERMS=IFIX(FINTIM/DELTIM)+1

188
SPACE=DELTIM/2.

C

*** Clear Screen & Home Cursor ***
call CLESCR
call GOTOXY(10,29)
write(*,*) 'Elapsed Simulation Time'
call GOTOXY(12,41)
write(*,*) 'seconds'
call GOTOXY(15,1)
write(*,15) BEGTIM,FINTIM,DELTIM,NTERMS

format(22X,'Simulation Start Time --> ','F8.7,/',
+ 23X,'Simulation Stop Time --> ','F8.7,/',
+ 23X,'Simulation Step Size --> ','F8.7,/',
+ 23X,'Total Number of Steps --> ',I4)
call GOTOXY(21,1)

if ((NONLIN .eq. 'r') .or. (NONLIN .eq. 'R')) then
  write(*,*) '*** NON-LINEAR ELEMENT IS RELAY ***'
elseif ((NONLIN .eq. 'p') .or. (NONLIN .eq. 'P')) then
  write(*,*) '*** NON-LINEAR ELEMENT IS PWM ***'
else
  write(*,*) '*** NON-LINEAR ELEMENT NOT SELECTED ***'
  write(*,*) '... Return to Main Menu ...'
  PAUSE
go to 1
endif

DO 100 NTIME = 1,NTERMS
  TIME = (NTIME-1)*DELTIM

*** Output Elapsed Simulation Time to Display ***
call GOTOXY(12,33)
write(*,93) TIME
93 format(F7.6)

C

... Command Input Signal Generator ...
if (TYPE .eq. 'STEP') then
  ORDER=STPMAG
elseif (TYPE .eq. 'RAMP') then
  ORDER=RSLOPE*TIME
elseif (TYPE .eq. 'SINE') then
  ORDER=SINAMP*sin(SINFRQ*TIME+SINPHA)
endif

VSGDEL=(VSGREF-VSG)/2.
P1=-.5
P2=.5
call DEADSP(P1,P2,VSGDEL,VSGERR)
call INTGRL(NTIME,NTIM21,DELTIM,VINIC,PREV5,VSGERR,
CURV5,HOLD1,VININT)

VINHFG=KINT*VININT

**********************************************************************
***** INITIAL ITERATIVE CONDITIONS *****
**********************************************************************

if (NTIME .eq. 2) then
WM=WM+(ORDER-X3)*1.E-3
X3=X3+(ORDER-X3)*1.E-3
ZA3=1.E-6
ZB3=-1.E-4
ZC3=1.E-4
endif

HALF=0

**********************************************************************
**** ITERATIVE LOOP ********************
**********************************************************************
do 500 NUMIT=1,20
THETA=X3

*** Phase angles ***
BEMFA=(3.0*SIN(2.0*THETA+(11.0*PI/6.0))+.590*SIN(10.0*THETA+(1.0*PI/6.0)))
BEMFB=(3.0*SIN(2.0*THETA+(7.0*PI/6.0))+.590*SIN(10.0*THETA+(5.0*PI/6.0)))
BEMFC=(3.0*SIN(2.0*THETA+(3.0*PI/6.0))+.590*SIN(10.0*THETA+(9.0*PI/6.0)))

*** Normalize leg EMF ***
VEMFA = BEMFA * KB/KK3 * WM
VEMFB = BEMFB * KB/KK3 * WM
VEMFC = BEMFC * KB/KK3 * WM
VEMFAC = VEMFA - VEMFC
VEMFBA = VEMFB - VEMFA
VEMFCB = VEMFC - VEMFB

call COMADV(THETA,PI,float(DIR)*THADV,THCON,THRST)
call HALL(THC0N,SE1,SE2,SE3)

THETAF=REDRAT*THETA*180./PI
OMEGAF=REDRAT*WM*180./PI
POSERR=ORDER-KP*THETAF-KV*OMEGAF

... Utilization of Non-Linear Element ...

... PULSE WIDTH MODULATOR ...
if ((NONLIN .eq. 'p') .or. (NONLIN .eq. 'P')) then
call PWMOD(TIME,NUMIT,TSTART,PERIOD,TOGGLE,POSERR,+
DBAND,VIF,VIB,DIR,VREF,THRESH,KPWM)
elseif ((NONLIN .eq. 'r') .or. (NONLIN .eq. 'R')) then
call RELAY(POSERR,DBAND,VIF,VIB,DIR)
else

C  HALF=0
C   ***********************************************************
C   ******************* ITERATIVE LOOP ********************
C   ***********************************************************
C   do 500 NUMIT=1,20
C   THETA=X3
C   C *** Phase angles ***
C   BEMFA=(3.0*SIN(2.0*THETA+(11.0*PI/6.0))+.590*SIN(10.0*THETA+(1.0*PI/6.0)))
C   + THETA+(1.0*PI/6.0)))
C   BEMFB=(3.0*SIN(2.0*THETA+(7.0*PI/6.0))+.590*SIN(10.0*THETA+(5.0*PI/6.0)))
C   + THETA+(5.0*PI/6.0)))
C   BEMFC=(3.0*SIN(2.0*THETA+(3.0*PI/6.0))+.590*SIN(10.0*THETA+(9.0*PI/6.0)))
C   + THETA+(9.0*PI/6.0)))
C *** Normalize leg EMF ***
C VEMFA = BEMFA * KB/KK3 * WM
C VEMFB = BEMFB * KB/KK3 * WM
C VEMFC = BEMFC * KB/KK3 * WM
C VEMFAC = VEMFA - VEMFC
C VEMFBA = VEMFB - VEMFA
C VEMFCB = VEMFC - VEMFB
C
C call COMADV(THETA,PI,float(DIR)*THADV,THCON,THRST)
call HALL(THC0N,SE1,SE2,SE3)
C
C THETAF=REDRAT*THETA*180./PI
C OMEGAF=REDRAT*WM*180./PI
C POSERR=ORDER-KP*THETAF-KV*OMEGAF
C
C ... Utilization of Non-Linear Element ...
C ...

call GOTOXY(21,1)
write(*,*)
write(*,*), *** NON-LINEAR ELEMENT NOT SELECTED ***
... Return to Main Menu ...
PAUSE
go to 1
endif
VIN=VIF-VIB
IL=VIN/(2.0*RCUT)

**************************************************************
c***** INITIAL ITERATIVE CONDITIONS FOR INNER ITERATION *****
**************************************************************

TRIG1 = 0
TRIG2 = 0
TRIG3 = 0
TRIG4 = 0
TRIG5 = 0
TRIG6 = 0

if ((TIME .ge. TFPLUG) .or. (TIME .eq. TOPLUG)) then
  if (float(DIRM1)*float(DIR) .lt. 0.) then
    TOPLUG=TIME
    TFPLUG=TOPLUG+TDPLUG
    PLUG=1
  else
    TOPLUG=0.
    TFPLUG=0.
    PLUG=0
  endif
endif

call TRANSW(TIME,REVIM,BEMFA,BEMFB,BEMFC,BEMFT,VEMFA,
+ VEMFB,VEMFC,VIB,VIF,IM,IMA,IMB,IMC,VN1,VN2,SW1,
+ SW2,SW3,SW4,SW5,SW6,THCON,THCON1,RSAT,POSIT,DIR,
+ VD1D,VD2D,VD3D,VD4D,VD5D,VD6D,RIN,ROUT,RS,
+ REQ1,REQ2,REQB1,REQB2,REQC1,REQC2,LAB,IBC,ICA,
+ FVOA,FVBO,FVCO,NODE,TRIG1,TRIG2,TRIG3,TRIG4,
+ TRIG5,TRIG6,RA,PLUG,VAIND,VBIND,VCIND)
call TCONST(ICQ1,.75-SW1,TTIME,NTIME,NTIM01,DELTIM,Q1EXP)
call TCONST(ICQ2,.75-SW2,TTIME,NTIME,NTIM02,DELTIM,Q2EXP)
call TCONST(ICQ3,.75-SW3,TTIME,NTIME,NTIM03,DELTIM,Q3EXP)
call TCONST(ICQ4,.75-SW4,TTIME,NTIME,NTIM04,DELTIM,Q4EXP)
call TCONST(ICQ5,.75-SW5,TTIME,NTIME,NTIM05,DELTIM,Q5EXP)
call TCONST(ICQ6,.75-SW6,TTIME,NTIME,NTIM06,DELTIM,Q6EXP)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q1EXP,RQ1)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q2EXP,RQ2)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q3EXP,RQ3)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q4EXP,RQ4)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q5EXP,RQ5)
call LIMIT(RSAT,RCUT,2.0*RCUT*Q6EXP,RQ6)

REQA1=RQ1*RD1/(RQ1+RD1)
REQA2=RQ2*RD2/(RQ2+RD2)
REQB1=RQ3*RD3/(RQ3+RD3)
REQB2=RQ4*RD4/(RQ4+RD4)
REQC1=RQ5*RD5/(RQ5+RD5)
REQC2=RQ6*RD6/(RQ6+RD6)

ASIGN=ZA3
BSIGN=ZB3
CSIGN=ZC3

AHALF=0
BHALF=0
CHALF=0

---------------------------------------------------------------
***** START INNER LOOP *****
---------------------------------------------------------------

do 650 NUMIN=1,20
IMA=ZA3
IMB=ZB3
IMC=ZC3

CALL DERIV(DELTIM,NTIME,NTIME12,0.0,PREV2,IMA,YY2,YDM12,YD2,
          + YDDM12,YDD2,DIADT)
CALL DERIV(DELTIM,NTIME,NTIME13,0.0,PREV3,IMB,YY3,YDM13,YD3,
          + YDDM13,YDD3,DIBDT)
CALL DERIV(DELTIM,NTIME,NTIME14,0.0,PREV4,IMC,YY4,YDM14,YD4,
          + YDDM14,YDD4,DICDT)
VAIND=LA*DIADT
VBIND=LA*DIBDT
VCIND=LA*DICDT

FVAO=IMA*(RA+RIN/2.)+VAIND
FVBO=IMB*(RA+RIN/2.)+VBIND
FVCO=IMC*(RA+RIN/2.)+VCIND

VAO=FVAO+VEMFA
VBO=FVBO+VEMFB
VCO=FVCO+VEMFC

call TRANS(Time,REVTIME,BEFA,BEMFB,BECFA,BEFT,VEFMA,
+ VEMFB,VEFMC,VIB,VIF,IM,IMA,IMB,IMC,VN1,VN2,SW1,
+ SW2,SW3,SW4,SW5,SW6,TCON,THCON1,RSAT,POSIT,DIR,
+ VD1D,VD2D,VD3D,VD4D,VD5D,VD6D,RIN,ROUT,RS,
+ REQA1,REQA2,REQB1,REQB2,REQC1,REQC2,IB,IBC,ICA,
+ FVAO,FVBO,FVCO,NODE,TRIG1,TRIG2,TRIG3,TRIG4,

192
call LIMIT(-.7, 800.0, VD1D, VD1D)
call LIMIT(-.7, 800.0, VD2D, VD2D)
call LIMIT(-.7, 800.0, VD3D, VD3D)
call LIMIT(-.7, 800.0, VD4D, VD4D)
call LIMIT(-.7, 800.0, VD5D, VD5D)
call LIMIT(-.7, 800.0, VD6D, VD6D)
call FCNSW(VD1D+.6, RSAT, RSAT, RCUT, RDX1)
call FCNSW(VD2D+.6, RSAT, RSAT, RCUT, RDX2)
call FCNSW(VD3D+.6, RSAT, RSAT, RCUT, RDX3)
call FCNSW(VD4D+.6, RSAT, RSAT, RCUT, RDX4)
call FCNSW(VD5D+.6, RSAT, RSAT, RCUT, RDX5)
call FCNSW(VD6D+.6, RSAT, RSAT, RCUT, RDX6)

RD1=RDX1+RDADJ
RD2=RDX2+RDADJ
RD3=RDX3+RDADJ
RD4=RDX4+RDADJ
RD5=RDX5+RDADJ
RD6=RDX6+RDADJ

REQA1=REQA1+RS1+ROUT/2.
REQAS2=REQA2+RS2+ROUT/2.
REQBS1=REQB1+RS1+ROUT/2.
REQBS2=REQB2+RS2+ROUT/2.
REQCS1=REQC1+RS1+ROUT/2.
REQCS2=REQC2+RS2+ROUT/2.

VANTH=VN1*REQAS2/(REQAS1+REQAS2)+VN2*REQAS1/(REQAS1+REQAS2)
VBTH=VN1*REQBS2/(REQBS1+REQBS2)+VN2*REQBS1/(REQBS1+REQBS2)
VCNTH=VN1*REQCS2/(REQCS1+REQCS2)+VN2*REQCS1/(REQCS1+REQCS2)

REQA=REQAS1*REQAS2/(REQAS1+REQAS2)
REQB=REQBS1*REQBS2/(REQBS1+REQBS2)
REQC=REQCS1*REQCS2/(REQCS1+REQCS2)

REQAB=2.0*RA+1.0*RIN+REQA+REQB
REQBC=2.0*RA+1.0*RIN+REQB+REQC
REQCA=2.0*RA+1.0*RIN+REQC+REQA

IAB=(VANTH-VBTH)/REQAB
IBC=(VBTH-VCNTH)/REQBC
ICA = (VCNTH - VANTH) / REQCA

ABTAU = 2.0 * LA / REQAB
BCTAU = 2.0 * LA / REQBC
CATAU = 2.0 * LA / REQCA

*** DISCHARGE OF PHASE CURRENTS (FREEWHEELING DIODES) ***

if (PLUG .eq. 1) then
  IAB = 0.
  IBC = 0.
  ICA = 0.
elseif ((POSIT .eq. 1) .or. (POSIT .eq. 4)) then
  IAB = 0.
  ICA = 0.
elseif ((POSIT .eq. 2) .or. (POSIT .eq. 5)) then
  IBC = 0.
  ICA = 0.
elseif ((POSIT .eq. 3) .or. (POSIT .eq. 6)) then
  IAB = 0.
  IBC = 0.
endif

call TCONST (ICIMAB, IAB, ABTAU, NTIME, NTIM32, DELTIM, IMAB)
call TCONST (ICIMBC, IBC, BCTAU, NTIME, NTIM33, DELTIM, IMBC)
call TCONST (ICIMCA, ICA, CATAU, NTIME, NTIM34, DELTIM, IMCA)

FA3 = IMAB - IMCA - ZA3
FB3 = IMBC - IMAB - ZB3
FC3 = IMCA - IMBC - ZC3

IMA = IMAB - IMCA
IMB = IMBC - IMAB
IMC = IMCA - IMBC

IM = (abs(IMA) + abs(IMB) + abs(IMC)) / 2.

*****************************************
if (NTIME .eq. 1) go to 550

if (NUMIN .eq. 1) then
  ZA1 = ZA3
  ZB1 = ZB3
  ZC1 = ZC3
  FA1 = FA3
  FB1 = FB3
  FC1 = FC3
  ZA3 = IMA
  ZB3 = IMB
  ZC3 = IMC

194
go to 650

elseif (NUMIN .eq. 2) then

ZA2=ZA3
ZB2=ZB3
ZC2=ZC3
FA2=FA3
FB2=FB3
FC2=FC3
ASIGN=FA3
BSIGN=FB3
CSIGN=FC3

if (abs(FA1-FA2) .eq. 0.) FA2=.5*FA2+1.E-6
if (abs(FB1-FB2) .eq. 0.) FB2=.5*FB2+1.E-6
if (abs(FC1-FC2) .eq. 0.) FC2=.5*FC2+1.E-6
if (abs(FA1-FA2) .ne. 0.) ZA3=ZA2-FA2*(ZA1-ZA2)/(FA1-FA2)
if (abs(FB1-FB2) .ne. 0.) ZB3=ZB2-FB2*(ZB1-ZB2)/(FB1-FB2)
if (abs(FC1-FC2) .ne. 0.) ZC3=ZC2-FC2*(ZC1-ZC2)/(FC1-FC2)
go to 650

else

C ... "A" Leg ...

if (FA3 .ne. 0.) then

if (FA1*FA2 .lt. 0.) then

if (FA1*FA3 .lt. 0.) then

ZA2=ZA3
FA2=FA3
if (FA3*ASIGN .gt. 0.) FA1=FA1/2.
ZA3=ZA2-FA2*(ZA2-ZA1)/(FA2-FA1)
elseif (FA1*FA3 .gt. 0.) then

ZA1=ZA3
FA1=FA3
if (FA3*ASIGN .gt. 0.) FA2=FA2/2.
ZA3=ZA2-FA2*(ZA2-ZA1)/(FA2-FA1)
endif

elseif (FA1*FA2 .gt. 0.) then

C ... Cautious forward iteration using halving and checking scheme in

C unbracketed region.

if (AHALF .eq. 0) then

ZA1=ZA2
ZA2=ZA3
FA1=FA2
FA2=FA3
ZA3=ZA1+(ZA2-ZA1)/2.
AHALF=1
elseif (AHALF .eq. 1) then

if (FA1*FA3 .lt. 0.) then

AHALF=0
ZA2=ZA3
FA2=FA3
ZA3=ZA2-FA2*(ZA2-ZA1)/(FA2-FA1)
elseif (FA1*FA3 .gt. 0.) then

AHALF=0

195
ZA1=ZA2
ZA2=ZA3
FA1=FA2
FA2=FA3
if (abs(FA2-FA1).eq. 0.) FA2=FA2/2.
if (abs(FA2-FA1).ne.0.) ZA3=ZA2-FA2*(ZA2-ZA1)/(FA2-FA1)
endif
endif
elseif (FA1*FA2 .eq. 0.) then
ZA1=ZA2
ZA2=ZA3
FA1=FA2
FA2=FA3
if (abs(FA2-FA1).eq. 0.) FA2=FA2/2.
if (abs(FA2-FA1).ne.0.) ZA3=ZA2-FA2*(ZA2-ZA1)/(FA2-FA1)
endif
endif
C ... "B" Leg ...
if (FB3 .ne. 0.) then
if (FB1*FB2 .lt. 0. ) then
if (FB1*FB3 .lt. 0.) then
ZB2=ZB3
FB2=FB3
if (FB3*BSIGN .gt. 0.) FB1=FB1/2.
ZB3=ZB2-2*FB2*(ZB2-ZB1)/(FB2-2)
elseif (FB1*FB3 .gt. 0.) then
ZB1=ZB3
FB1=FB3
if (FB3*BSIGN .gt. 0.) FB2=FB2/2.
ZB3=ZB2-2*FB2*(ZB2-ZB1)/(FB2-2)
endif
elseif (FB1*FB2 .gt. 0.) then
C ... Cautious forward iteration using halving and checking scheme in C unbracketed region.
if (BHALSE .eq. 0.) then
ZB1=ZB2
ZB2=ZB3
FB1=FB2
FB2=FB3
ZB3=ZB1+(ZB2-ZB1)/2.
BHALSE=1
elseif (BHALSE .eq. 1) then
if (FB1*FB3 .lt. 0.) then
BHALSE=0
ZB2=ZB3
FB2=FB3
ZB3=ZB2-2*FB2*(ZB2-ZB1)/(FB2-2)
elseif (FB1*FB3 .gt. 0.) then
BHALSE=0
ZB1=ZB2
ZB2=ZB3
FB1 = FB2
FB2 = FB3
if (abs(FB2 - FB1) .eq. 0.) FB2 = FB2/2.
if (abs(FB2 - FB1) .ne. 0.) ZB3 = ZB2 * FB2 * (ZB2 - ZB1) / (FB2 - FB1)
endif
endif
elseif (FB1 * FB2 .eq. 0.) then
  ZB1 = ZB2
  ZB2 = ZB3
  FB1 = FB2
  FB2 = FB3
  if (abs(FB2 - FB1) .eq. 0.) FB2 = FB2/2.
  if (abs(FB2 - FB1) .ne. 0.) ZB3 = ZB2 * FB2 * (ZB2 - ZB1) / (FB2 - FB1)
endif
endif

C ... "C" Leg ...
if (FC3 .ne. 0.) then
  if (FC1 * FC2 .lt. 0.) then
    if (FC1 * FC3 .lt. 0.) then
      ZC2 = ZC3
      FC2 = FC3
      if (FC3 * CSIGN .gt. 0.) FC1 = FC1/2.
      ZC3 = ZC2 - FC2 * (ZC2 - ZC1) / (FC2 - FC1)
    elseif (FC1 * FC3 .gt. 0.) then
      ZC1 = ZC3
      FC1 = FC3
      if (FC3 * CSIGN .gt. 0.) FC2 = FC2/2.
      ZC3 = ZC2 - FC2 * (ZC2 - ZC1) / (FC2 - FC1)
    endif
  elseif (FC1 * FC2 .gt. 0.) then
    C ... Cautious forward iteration using halving and checking scheme in
    C unbracketed region.
    if (CHALF .eq. 0) then
      ZC1 = ZC2
      ZC2 = ZC3
      FC1 = FC2
      FC2 = FC3
      ZC3 = ZC1 + (ZC2 - ZC1)/2.
      CHALF = 1
    elseif (CHALF .eq. 1) then
      if (FC1 * FC3 .lt. 0.) then
        CHALF = 0
        ZC2 = ZC3
        FC2 = FC3
        ZC3 = ZC2 - FC2 * (ZC2 - ZC1) / (FC2 - FC1)
      elseif (FC1 * FC3 .gt. 0.) then
        CHALF = 0
        ZC1 = ZC2
        ZC2 = ZC3
        FC1 = FC2
        FC2 = FC3
      endif
    endif
  endif
endif
if (abs(FC2-FC1).eq.0.) FC2=FC2/2.
if (abs(FC2-FC1).ne.0.) ZC3=ZC2-FC2*(ZC2-ZC1)/(FC2-FC1)
endif
endif
elseif (FC1*FC2.eq.0.) then
  ZC1=ZC2
  ZC2=ZC3
  FC1=FC2
  FC2=FC3
  if (abs(FC2-FC1).eq.0.) FC2=FC2/2.
  if (abs(FC2-FC1).ne.0.) ZC3=ZC2-FC2*(ZC2-ZC1)/(FC2-FC1)
endif
endif
endif
C
ASIGN=FA3
BSIGN=FB3
CSIGN=FC3
C
if (abs(ZA3).gt.1.E-4) RERRZA=abs(FA3/ZA3)
if (abs(ZA3).le.1.E-4) RERRZA=abs(FA3)
if (abs(ZB3).gt.1.E-4) RERRZB=abs(FB3/ZB3)
if (abs(ZB3).le.1.E-4) RERRZB=abs(FB3)
if (abs(ZC3).gt.1.E-4) RERRZC=abs(FC3/ZC3)
if (abs(ZC3).le.1.E-4) RERRZC=abs(FC3)
C
if ((RERRZA.lt.1.E-6).and. (RERRZB.lt.1.E-6).and. + (RERRZC.lt.1.E-6)) go to 550
C
650 continue
C ****************************
C ***** End of inner loop *****
C ****************************
C
call GOTOXY(23,12)
write(*,651) TIME
651 format(1x,'Stiff Phase Current Characteristics at ',F8.7, + ' seconds')
C
550  TA=IMA*KT*BEMFA/KK3
    TB=IMB*KT*BEMFB/KK3
    TC=IMC*KT*BEMFC/KK3
    TM=TA+TB+TC
call TCONST(ICTMF,TM,.005,NTIME,NTIM47,DELTIM,TMF)
    TM=KT*IM
    TB=BM*WM
    VKB=MB*WM
C
if (WM.lt.0.0) go to 180
  TN1=TM-TL
  go to 185
180  IN1=IN+1
185  continue

IN2=IN1/B

C
WM1=WM
THETA1=THETA
call TCONST(WM0, IN2, A2, NTIME, NTIM48, DELTIM, WM)
call INTGRL(NTIME, NTIM23, DELTIM, THETA0, PREV7, WM, CURV7, 

+ HOLD3, THETA)

C
THETA=PTHETA
F3=THETA-THETA1

C
****** MODIFIED LINEAR INTERPOLATION *****
C

C
if (NTIME .eq. 1) go to 601
C

if (NUMIT .eq. 1) then
  X1=X3
  F1=F3
  X3=THETA
  go to 500
elseif (NUMIT .eq. 2) then
  X2=X3
  F2=F3
  if (abs(F2-F1) .eq. 0.) F2=.5*F2+1.E-6
  if (abs(F2-F1) .ne. 0.) X3=X2-F2*(X2-X1)/(F2-F1)
  TSIGN=F3
  go to 500
elseif (F3 .ne. 0.) then
  if (F1*F2 .lt. 0.) then
    if (F1*F3 .lt. 0.) then
      X2=X3
      F2=F3
      if (F3*TSIGN .gt. 0.) F1=F1/2.
      X3=X2-F2*(X2-X1)/(F2-F1)
    elseif (F1*F3 .gt. 0.) then
      X1=X3
      F1=F3
      if (F3*TSIGN .gt. 0.) F2=F2/2.
      X3=X2-F2*(X2-X1)/(F2-F1)
    endif
  elseif (F1*F2 .gt. 0.) then
    C...Cautious forward iteration using halving and checking scheme in 
    C  unbracketed region.
    if (HALF .eq. 0) then
      X1=X2
      X2=X3
      F1=F2
      F2=F3
  endif

199
X3=X1+(X2-X1)/2.
HALF=1
elseif (HALF .eq. 1) then
  if (F1*F3 .lt. 0.) then
    HALF=0
    X2=X3
    F2=F3
    X3=X2-F2*(X2-X1)/(F2-F1)
  elseif (F1*F3 .gt. 0.) then
    HALF=0
    X1=X2
    X2=X3
    F1=F2
    F2=F3
    if (abs(F2-F1) .eq. 0.) F2=F2/2.
    if (abs(F2-F1) .ne. 0.) X3=X2-F2*(X2-X1)/(F2-F1)
  endif
endif
elseif (F1*F2 .eq. 0.) then
  X1=X2
  X2=X3
  F1=F2
  F2=F3
  if (abs(F2-F1) .eq. 0.) F2=F2/2.
  if (abs(F2-F1) .ne. 0.) X3=X2-F2*(X2-X1)/(F2-F1)
endif
endif
C
if (abs(X3) .gt. 1.E-4) RELERR=abs(F3/X3)
if (abs(X3) .le. 1.E-4) RELERR=abs(F3)
C
if (RELERR .lt. 1.E-6) go to 601
500 continue
  call GOTOXY(24,15)
  write(*,652) TIME
652 format (ix,'FAILED Convergence of Outer Loop at ',F8.7, ' seconds')
C
601 WMREW=WM*30./PI
  THDEG=THETA*180.0/PI
  X3=THETA+WM*DELTIM
  THETAF=REDRAT*THETA*180./PI
  OMEGAF=REDRAT*WM*180./PI
  FREIMA=ZA3+.5*DIADT*DELTIM
  FREIMB=ZB3+.5*DIBDT*DELTIM
  FREIMC=ZC3+.5*DICDT*DELTIM
  ZA3=FREIMA
  ZB3=FREIMB
  ZC3=FREIMC
  DIRM1=DIR
C
C **************************************************
C  ***** End outer loop *****
C  ********************************************
C
C  ... Generate Plotting Arrays ...
    if (TIME .ge. BEGTIM) then
        NCTR=NCTR+1
    if ((mod(NCTR,SIM2PL) .eq. 0) .or. (NCTR .eq. 1)) then
        NPTS=NPTS+1
C  ... Output data to file for future retrieval and plotting ...
C
    XTIME(NPTS)=TIME

    write(4,1200) TIME,THETAORDER,OMEGAORDER,NODE,PLUG
    write(5,1210) IM,IMA,IMB,IMC,IMD
    write(6,1220) VD1V,VD2V,VD3V,VD4V,VD5V,VD6V
    write(7,1230) SV1,SV2,SV3,SV4,SV5,SV6

    1200 format(5F15.7,3X,I3)
    1210 format(6F15.7)
    1220 format(6F15.7)
    1230 format(6F5.2)
    endif
    endif
C
100 CONTINUE
    close(4,STATUS='KEEP')
    close(5,STATUS='KEEP')
    close(6,STATUS='KEEP')
    close(7,STATUS='KEEP')
C
C  ********************************************
C  ***** Plotting selection *****
C  ********************************************

C  *** Clear Screen & Home Cursor ***

400 call CLRSCR
C
    write(*,1305)
    read(*,'(A)') DISOPT
C
    if ((DISOPT .eq. 'm') .or. (DISOPT .eq. 'M')) then
        MODEL=99
        IPORT=99
    elseif ((DISOPT .eq. 'p') .or. (DISOPT .eq. 'P')) then
        MODEL=MODEL
        IPORT=1
    C  ... IPORT=9600 is COM1 ...
    C  ... IPORT=9650 is COM2 ...
        if ((MODEL .eq. 20) .or. (MODEL .eq. 30)) IPORT=9650
    elseif ((DISOPT .eq. 'r') .or. (DISOPT .eq. 'R')) then
        GO TO 13
    elseif ((DISOPT .eq. 'w') .or. (DISOPT .eq. 'W')) then

201
call PRNOUT(BEGTIM, FINTIM, PTSPLT, MAXITS, SIM2PL, DELTIM, NTERMS, + E0, EDOTO, TYPE, STPMAG, RSLOPE, SINAMP, SINFRQ, SINPHA, + NLCHAR, DBAND, PERIOD, KPWM, RIN, ROUT, Rdadj, KTM, KBM, + KV, KP, RA, LA, BM, BL, JM, JL, RSAT, RCUT, ITIME, RS, THADV, + TDPLUG)

go to 400

elseif (DISOPT .eq. 's') .or. (DISOPT .eq. 'S')) then

*** OPEN/WRITE/CLOSE INPUT DATA FILE ***

open(2,FILE='MOTOR.INP',STATUS='OLD',ACCESS='SEQUENTIAL')
write(2,1000) PMODEL, PRCHR, RIN, ROUT, Rdadj
write(2,1020) BEGTIM, FINTIM, PTSPLT, MAXITS, SIM2PL
write(2,1022) KPWM, KV, E0, EDOTO, ITIME, XORG, YORG
write(2,1024) RS, RSAT, JL, BL, KTM
write(2,1026) JM, RA, LA, KBM, THADV, TDPLUG
write(2,1028) BM, KP, PERIOD, DBAND, KA
write(2,1030) TYPE, RCUT, STPMAG, RSLOPE
write(2,1032) SINAMP, SINFRQ, SINPHA, DRIVE

close(2,STATUS='KEEP')

elseif ((DISOPT .eq. 'Q') .or. (DISOPT .eq. 'q')) then

GO TO 460

else

end if

call DISPLA(XORG, YORG, DISOPT, PRCHR, ELEMENT, DRIVE, XTIME)

go to 400

460 continue

*** I/O Statements ***


5 format(32X, '*** MAIN MENU ***', / / , + 20X, '[H] ----> HARDWARE Configuration Menu', / ,
+ 17X, '[P] Pulse Width Modulator',//,
+ 1X,F15.7,1X,'[KV] Velocity Feedback Constant',//,
+ 1X,F15.7,1X,'[XP] Position Feedback Constant',//,
+ 17X, '[Q] QUIT THIS MENU/RETURN TO MAIN MENU',//,
+ 1X,'Enter Selection --- > ',")
C
272 format(10X,'CURRENT SELECTION -- > ',A30)
C
280 format(///,8X,'*** SATURATING AMPLIFIER SPECIFICATIONS ***',///,
+ 1X,F15.7,1X,'[DBAND] Deadband Applied to System Feedback',///,
+ 1X,F15.7,1X,'[KA] Amplifier Gain',///,
+ 17X,'[Q] QUIT THIS MENU',///,
+ 1X,'Enter the selection (UPPERCASE) --- > ',")
C
290 format(///,15X,'*** RELAY SPECIFICATIONS ***',///,
+ 1X,F15.7,1X,'[DBAND] Deadband Applied to System Feedback',///,
+ 17X,'[Q] QUIT THIS MENU',///,
+ 1X,'Enter the selection (UPPERCASE) --- > ',")
C
300 format(///,5X,'*** PULSE WIDTH MODULATOR SPECIFICATIONS ***',///,
+ 1X,F15.7,1X,'[DBAND] Deadband Applied to System Feedback',///,
+ 1X,F15.7,1X,'[PERIOD] Period of PWM Reference Cycle(sec)',///,
+ 1X,F15.7,1X,'[KPWM] PWM Amplifier Gain',///,
+ 17X,'[Q] QUIT THIS MENU',///,
+ 1X,'Enter the selection (UPPERCASE) --- > ',")
C
1305 format(///,2X,'Display options:',///,
+ 5X,'[M] MONITOR',///,
+ 5X,'[P] PRINTER',///,
+ 5X,'[R] RETURN TO START-UP MENU (RE-INITIALIZE)',///,
+ 5X,'[W] WRITE SIMULATION SPECIFICATIONS TO PRINTER',///,
+ 5X,'[S] SAVE SIMULATION SPECIFICATIONS TO DISK',///,
+ 5X,'[Q] QUIT THE PROGRAM',///,
C
1500 format(1X,I3,2X,A50)
STOP
END

------------------------------------------
C BLDCM2A.FOR (subroutines PRNOPT,MOTPAR,DISPLA,PRNOUT)
C These subroutines are compiled as a group, but separately from
C BLDCM1 & BLDCM2A due to size limitations. Subroutines must be
C linked together.
C
Last Revision --> 04 April 1987
LT Vincent S. Rossitto, USN
C
******************************
C *** PRINTER/ PLOTTER SELECTION MENU ***
C ------------------------------------------

204
C Provides user interface for selection of Printer for output device.

C subroutine PRNOPT(PMODEL, PRTSEL, PRTCHR)

implicit REAL*4 (A-Z)
integer*2 PMODEL
character*6 PRTSEL
character*51 PRTCHR

call CLRSCR
write(*,130)
read(*,'(A)' PRTSEL

if (PRTSEL .eq. '0') then
  PRTCHR='Epson FX-80 Printer, single density'
  PMODEL=0
elseif (PRTSEL .eq. '1') then
  PRTCHR='Epson FX-80 Printer, double density'
  PMODEL=1
elseif (PRTSEL .eq. '2') then
  PRTCHR='Epson FX-80 Printer, dble spd,dual density'
  PMODEL=2
elseif (PRTSEL .eq. '3') then
  PRTCHR='Epson FX-80 Printer, quad density'
  PMODEL=3
elseif (PRTSEL .eq. '4') then
  PRTCHR='Epson FX-80 Printer, CRT Graphics I'
  PMODEL=4
elseif (PRTSEL .eq. '5') then
  PRTCHR='Epson FX-80 Printer, plotter graphics'
  PMODEL=5
elseif (PRTSEL .eq. '6') then
  PRTCHR='Epson FX-80 Printer, CRT Graphics II'
  PMODEL=6
elseif (PRTSEL .eq. '10') then
  PRTCHR='Epson FX-100 Printer, single density'
  PMODEL=7
elseif (PRTSEL .eq. '11') then
  PRTCHR='Epson FX-100 Printer, double density'
  PMODEL=11
elseif (PRTSEL .eq. '12') then
  PRTCHR='Epson FX-100 Printer, dble spd,dual density'
  PMODEL=12
elseif (PRTSEL .eq. '13') then
  PRTCHR='Epson FX-100 Printer, quad density'
  PMODEL=13
elseif (PRTSEL .eq. '14') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics I'
  PMODEL=14
elseif (PRTSEL .eq. '15') then
  PRTCHR='Epson FX-100 Printer, plotter graphics'

205
PMODEL=15
elseif (PRTSEL .eq. '16') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics II'
  PMODEL=16
elseif (PRTSEL .eq. '20') then
  PRTCHR='HP 7470A Graphics Plotter'
  PMODEL=20
elseif (PRTSEL .eq. '30') then
  PRTCHR='HP 7475A Graphics Plotter'
  PMODEL=30
elseif (PRTSEL .eq. '60') then
  PRTCHR='HP 2686A Laser Jet Printer'
  PMODEL=60
C
C ... Quit the Printer Menu ... 
elseif ((PRTSEL .eq. 'Q') .or. (PRTSEL .eq. 'q')) then
go to 101
else
  go to 131
dendif
101  continue
C
C 130 FORMAT(24X,*** PRINTER OPTIONS MENU ***,,//,
+ 15X,'[0] --> Epson FX-80 Printer, single density',//,
+ 15X,'[1] --> Epson FX-80 Printer, double density',//,
+ 15X,'[2] --> Epson FX-80 Printer, dble spd,dual density',//,
+ 15X,'[3] --> Epson FX-80 Printer, quad density',//,
+ 15X,'[4] --> Epson FX-80 Printer, CRT Graphics I',//,
+ 15X,'[5] --> Epson FX-80 Printer, plotter graphics',//,
+ 15X,'[6] --> Epson FX-80 Printer, CRT Graphics II',//,
+ 15X,'[10] --> Epson FX-100 Printer, single density',//,
+ 15X,'[11] --> Epson FX-100 Printer, double density',//,
+ 15X,'[12] --> Epson FX-100 Printer, dble spd,dual density',//,
+ 15X,'[13] --> Epson FX-100 Printer, quad density',//,
+ 15X,'[14] --> Epson FX-100 Printer, CRT Graphics I',//,
+ 15X,'[15] --> Epson FX-100 Printer, plotter graphics',//,
+ 15X,'[16] --> Epson FX-100 Printer, CRT Graphics II',//,
+ 15X,'[20] --> HP 7470A Graphics Plotter',//,
+ 15X,'[30] --> HP 7475A Graphics Plotter',//,
+ 15X,'[60] --> HP 2686A Laser Jet Printer (NPS installation)',//,
+ 15X,'[Q] --> QUIT THIS MENU',//,
+ 3X,'Enter Printer Selection Integer or Q to QUIT ---> ',//)
return
C
C ******************************************************
C **** Motor Parameter Input Subroutine ****
C ******************************************************
C subroutine MOTPAR(KTM,KBM,RIN,ROUT,RDADJ,RA,LA,BM,BL,
+ RSAT,RCUT,TTIME,RS,THADV,TDPLUG)
implicit REAL(A-Z)
character*6 ANS21
C
201 call CLRSCR
write(*,230) KTM,KBM,RIN,ROUT,RDADJ,RA,BM,BL,JM,JL,
+ RSAT,RCUT,TTIME,RS,THADV,TDPLUG
read(*,'(A)') ANS21
C
if (ANS21 .eq. 'KTM') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for KTM-->
 read(*,*) KTM
 go to 201
elseif (ANS21 .eq. 'KBM') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for KBM-->
 read(*,*) KBM
 go to 201
elseif (ANS21 .eq. 'RIN') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for RIN-->
 read(*,*) RIN
 go to 201
elseif (ANS21 .eq. 'ROUT') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for ROUT-->
 read(*,*) ROUT
 go to 201
elseif (ANS21 .eq. 'RDADJ') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for RDADJ-->
 read(*,*) RDADJ
 go to 201
elseif (ANS21 .eq. 'RA') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for RA-->
 read(*,*) RA
 go to 201
elseif (ANS21 .eq. 'LA') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for LA-->
 read(*,*) LA
 go to 201
elseif (ANS21 .eq. 'BM') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for BM-->
 read(*,*) BM
 go to 201
elseif (ANS21 .eq. 'BL') then
 call GOTOXY(24,1)
 write(*,'(A)') 'Enter a REAL value for BL-->
}
read(*,*) BL
go to 201
elseif (ANS21 .eq. 'JM') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for JM-->
    read(*,*) JM
    go to 201
elseif (ANS21 .eq. 'JL') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for JL-->
    read(*,*) JL
    go to 201
elseif (ANS21 .eq. 'RSAT') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for RSAT-->
    read(*,*) RSAT
    go to 201
elseif (ANS21 .eq. 'RCUT') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for RCUT-->
    read(*,*) RCUT
    go to 201
elseif (ANS21 .eq. 'TTIME') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for TTIME-->
    read(*,*) TTIME
    go to 201
elseif (ANS21 .eq. 'RS') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for RS-->
    read(*,*) RS
    go to 201
elseif (ANS21 .eq. 'THADV') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for THADV-->
    read(*,*) THADV
    go to 201
elseif (ANS21 .eq. 'TDPLUG') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for TDPLUG-->
    read(*,*) TDPLUG
    go to 201
C  ... Quit Motor Parameters Menu ...
elseif (ANS21 .eq. 'Q') then
    go to 1
else
    go to 201
end if
C  continue
C
230 format(10X,'MOTOR PARAMETER SELECTION MENU',///,
**GRAPHIC OUTPUT SELECTION MENU**

```
+ 1X,F15.7,1X, 'KTM'  Motor Torque Constant',/,
+ 1X,F15.7,1X, 'KBM'  Motor Back EMF Constant',/,
+ 1X,F15.7,1X, 'RIN'   Current Limiting Resistance (inside')/,
+ 1X,F15.7,1X, 'ROUT'  Current Limiting Resistance (outside')/,
+ 1X,F15.7,1X, 'RADJ'  Series R Added to Freewheeling Diodes',/,
+ 1X,F15.7,1X, 'RA'    Motor Phase Resistance(Ohm')/,
+ 1X,F15.7,1X, 'LA'    Motor Phase Inductance(Henries')/,
+ 1X,F15.7,1X, 'BM'    Motor Viscous Friction Coeff',/,
+ 1X,F15.7,1X, 'BL'    Load Viscous Friction Coeff',/,
+ 1X,F15.7,1X, 'JM'    Motor Inertia(oz-in/s^2')/,
+ 1X,F15.7,1X, 'JL'    Load Inertia(oz-in/s^2')/,
+ 1X,F15.7,1X, 'RSAT'  Equiv Ckt Resistance of Trans Satur',/,
+ 1X,F15.7,1X, 'RCUT'  Equiv Ckt Resistance of Trans Cutoff',/,
+ 1X,F15.7,1X, 'TTIME' Transistor Switching Time',/,
+ 1X,F15.7,1X, 'RS'    Internal Resist of Supply Voltage',/,
+ 1X,F15.7,1X, 'THADV' Commutation Advance (Electrical Deg')/,
+ 1X,F15.7,1X, 'TDPLUG' Commutation Dead Time while Plugging',/,
+ 10X,'(Q)' QUIT THIS MENU',/,
+ 1X,'Enter Variable Name(UPPERCASE) or Q to QUIT ---> ',
```

C
```
return
end
```

C
```
*** GRAPHIC OUTPUT SELECTION MENU ***
```

C
```
Provides user selection for graphic output of data.
```

C
```
subroutine DISPLA(XORG,YORG,DISOPT,PRTCHR,ELEMNT,DRIVE,XTIME)
implicit real*(A-Z)
common BEGTIM,FINTIM,NPTS,IOPORT,XLEN,YLEN,PLEN,PLEN1,
+         XTITLE,YTITLE,PTITLE,PTIT1
real*4 XTIME(1010),Y1(1010),Y2(1010)
integer*2 NPTS,PCTR,XLEN,YLEN,IOPORT,ELEMNT
character*3 DISOPT,PLOPT,DRIVE
character*25 XTITLE,YTITLE
character*51 PRTCHR,PTITLE,PTIT1
```

C
```
*** Clear Screen & Home Cursor ***
```
410 call CLRSCR
```
```
if ((DISOPT .eq. 'p') .or. (DISOPT .eq. 'P')) then
  call GOTOXY(24,20)
  write(*,*) PRTCHR
  call GOTOXY(1,1)
endif
write(*,1300)
read(*,'(A)') FLOPT
```
```
if ((FLOPT .eq. 'Q') .or. (FLOPT .eq. 'q')) then
```
go to 400
elseif (PLOPT .eq. '1') then
    XTITLE='TIME (seconds)'
    XLEN=15
    YTITLE='FIN POSITION (deg)'
    YLEN=19
    if (ELEMENT .eq. 1) then
        PTITLE='FIN POSITION RESPONSE WITH RELAY CONTROLLER'
        PLEN=44.
    elseif (ELEMENT .eq. 3) then
        PTITLE='FIN POSITION RESPONSE WITH PWM CONTROLLER'
        PLEN=42.
    endif
    PTITLE='BRUSHLESS DC MOTOR'
    PLEN1=20.
C
C ... Data Retrieval ... 
if (DRIVE .eq. 'A') then
    open(4, file='a:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
    open(4, file='b:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
    open(4, file='c:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
endif
    do 830 PCTR=1,NPTS
        read(4, 930) THETAF, ORDER
    format(15x, 2F15.7)
    Y1(PCTR)=THETAF
    Y2(PCTR)=ORDER
    continue
close(4, status='KEEP')
C
    call MGRAPH(XTIME, Y1, Y2, XORG, YORG, DISOPT)
C
elseif (PLOPT .eq. '2') then
    XTITLE='TIME (seconds)'
    XLEN=15
    YTITLE='FIN VELOCITY (deg/sec)'
    YLEN=23
    if (ELEMENT .eq. 1) then
        PTITLE='FIN VELOCITY RESPONSE WITH RELAY CONTROLLER'
        PLEN=44.
    elseif (ELEMENT .eq. 3) then
        PTITLE='FIN VELOCITY RESPONSE WITH PWM CONTROLLER'
        PLEN=42.
    endif
    PTITLE='BRUSHLESS DC MOTOR'
    PLEN1=20.
C
C ... Data Retrieval ... 
if (DRIVE .eq. 'A') then

open(4, file='a:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
  open(4, file='b:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
  open(4, file='c:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
endif

do 832 PCTR=1,NPTS
  read(4,932) OMEGAF
  Y1(PCTR)=OMEGAF
  continue
932  format(45X,F15.7)
close(4, status='KEEP')

C
call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (FLOPT .eq. '3') then
  XTITLE='TIME (seconds)'
  XLEN=15
  YTITLE='IM (amps)'
  YLEN=10
  if (ELEMNT .eq. 1) then
    FTITLE='MOTOR SOURCE CURRENT WITH RELAY CONTROLLER'
    FLEN=43.
  elseif (ELEMNT .eq. 3) then
    FTITLE='MOTOR SOURCE CURRENT WITH PWM CONTROLLER'
    FLEN=41.
  endif
  FTTL='BRUSHLESS DC MOTOR'
  FLEN=20.
C
C ... Data Retrieval ...
if (DRIVE .eq. 'A') then
  open(5, file='a:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
  open(5, file='b:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
  open(5, file='c:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
endif

do 833 PCTR=1,NPTS
  read(5,933) IM
  Y1(PCTR)=IM
  continue
933  format(F15.7)
close(5, status='KEEP')
C
 call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (FLOPT .eq. '4') then
  XTITLE='TIME (seconds)'
  XLEN=15
  YTITLE='IMA (amps)'
  
211
if (ELEMNT .eq. 1) then
   PTITLE='MOTOR PHASE A CURRENT WITH RELAY CONTROLLER'
   PLEN=44.
elseif (ELEMNT .eq. 3) then
   PTITLE='MOTOR PHASE A CURRENT WITH PWM CONTROLLER'
   PLEN=42.
endif
PTITl='BRUSHLESS DC MOTOR'
PLEN1=20.

C ... Data Retrieval ...
if (DRIVE .eq. 'A') then
   open(5,file='a:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
   open(5,file='b:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
   open(5,file='c:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
endif
   do 834 PCTR=1,NPTS
      read(5,934) IMA
      Y1(PCTR)=IMA
   continue
934 format(15X,F15.7)
close(5,status='KEEP')
C
   call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (FLOPT .eq. '5') then
   XTITLE='TIME (seconds)'
   XLEN=-15
   YTITLE='IMB (amps)'
   YLEN=11.
if (ELEMNT .eq. 1) then
   PTITLE='MOTOR PHASE B CURRENT WITH RELAY CONTROLLER'
   PLEN=44.
elseif (ELEMNT .eq. 3) then
   PTITLE='MOTOR PHASE B CURRENT WITH PWM CONTROLLER'
   PLEN=42.
endif
PTITl='BRUSHLESS DC MOTOR'
PLEN1=20.
C ... Data Retrieval ...
if (DRIVE .eq. 'A') then
   open(5,file='a:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
   open(5,file='b:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
   open(5,file='c:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
endif
do 835 PCTR=1,NPTS
   read(5,935) IMB
   Y1(PCTR)=IMB
835   continue
935   format(30X,F15.7)
   close(5,status='KEEP')
C
   call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (PLOPT .eq. '6') then
   XTITLE='TIME (seconds)'
   XLEN=15
   YTITLE='IMC (amps)'
   YLEN=11
   if (ELEMNT .eq. 1) then
      PTITLE='MOTOR PHASE C CURRENT WITH RELAY CONTROLLER'
   elseif (ELEMNT .eq. 3) then
      PTITLE='MOTOR PHASE C CURRENT WITH PWM CONTROLLER'
   endif
   PTIT1='BRUSHLESS DC MOTOR'
   PLEN1=20.
C
C ... Data Retrieval ...
   if (DRIVE .eq. 'A') then
      open(5,file='a:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
   elseif (DRIVE .eq. 'B') then
      open(5,file='b:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
   elseif (DRIVE .eq. 'C') then
      open(5,file='c:BLDCM2.VIR',status='OLD',access='SEQUENTIAL')
   endif
   do 836 PCTR=1,NPTS
      read(5,936) IMB
      Y1(PCTR)=IMB
836   continue
936   format(45X,F15.7)
   close(5,status='KEEP')
C
   call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (PLOPT .eq. '7') then
   XTITLE='TIME (seconds)'
   XLEN=15
   YTITLE='TM (oz-in)'
   YLEN=11
   if (ELEMNT .eq. 1) then
      PTITLE='MOTOR TORQUE (oz-in) WITH RELAY CONTROLLER'
   elseif (ELEMNT .eq. 3) then
      PTITLE='MOTOR TORQUE (oz-in) WITH PWM CONTROLLER'

213
PLEN=41.
endif
PTITL='BRUSHLESS DC MOTOR'
PLEN=20.

C ...
Data Retrieval ...
if (DRIVE.eq. 'A') then
  open(5, file='a:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE.eq. 'B') then
  open(5, file='b:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE.eq. 'C') then
  open(5, file='c:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
endif
  do 837 PCTR=1,NPTS
     read(5,937) TM
     Y1(PCTR)=TM
     continue
  enddo
  close(5, status='KEEP')

C call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (FLOPT.eq. '8') then
  XTITLE='TIME (seconds)'
  XLEN=15
  YTITLE='TMF (oz-in)'
  YLEN=12
  if (ELEMNT.eq. 1) then
    PTITLE='FILTERED MOTOR TORQUE (oz-in) WITH RELAY CONTROLLER'
    PLEN=51.
  elseif (ELEMNT.eq. 3) then
    PTITLE='FILTERED MOTOR TORQUE (oz-in) WITH PWM CONTROLLER'
    PLEN=50.
  endif
  PTITL='BRUSHLESS DC MOTOR'
  PLEN=20.
C ...
Data Retrieval ...
if (DRIVE.eq. 'A') then
  open(5, file='a:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE.eq. 'B') then
  open(5, file='b:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
elseif (DRIVE.eq. 'C') then
  open(5, file='c:BLDCM2.VIR', status='OLD', access='SEQUENTIAL')
endif
  do 838 PCTR=1,NPTS
     read(5,938) TMF
     Y1(PCTR)=TMF
     continue
  enddo
  close(5, status='KEEP')
call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)

elseif (PLOPT .eq. '9') then
    XTITLE='TIME (seconds)'
    XLEN=15
    YTITLE='VD1D (volts)'
    YLEN=13
    if (ELEMNT .eq. 1) then
        PTITLE='POWER TRANSISTOR #1 VOLTAGE DROP (RELAY)'
        PLEN=41.
    elseif (ELEMNT .eq. 3) then
        PTITLE='POWER TRANSISTOR #1 VOLTAGE DROP (PWM)'
        PLEN=39.
    endif
    PTITLE='BRUSHLESS DC MOTOR'
    PLEN=20.
endif

... Data Retrieval ...

if (DRIVE .eq. 'A') then
    open(6,file='a:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
    open(6,file='b:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
    open(6,file='c:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
endif

do 839 PCTR=1,NPTS
    read(6,939) VD1D
    Y1(PCTR)=VD1D
  continue

  format(F15.7)
  close(6,status='KEEP')

call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)

elseif (PLOPT .eq. '10') then
    XTITLE='TIME (seconds)'
    XLEN=15
    YTITLE='VD2D (volts)'
    YLEN=13
    if (ELEMNT .eq. 1) then
        PTITLE='POWER TRANSISTOR #2 VOLTAGE DROP (RELAY)'
        PLEN=41.
    elseif (ELEMNT .eq. 3) then
        PTITLE='POWER TRANSISTOR #2 VOLTAGE DROP (PWM)'
        PLEN=39.
    endif
    PTITLE='BRUSHLESS DC MOTOR'
    PLEN=20.
endif

... Data Retrieval ...

215
if (DRIVE .eq. 'A') then
   open(6, file='a:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
else if (DRIVE .eq. 'B') then
   open(6, file='b:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
else if (DRIVE .eq. 'C') then
   open(6, file='c:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
endif

   do 840 PCTR=1,NPTS
      read(6,940) VD2D
      Y1(PCTR)=VD2D
   continue
940 format(15X,F15.7)
   close(6, status='KEEP')
C
   call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
   elseif (PLOPT .eq. '11') then
      XTITLE='TIME (seconds)'
      XLEN=15
      YTITLE='VD3D (volts)'
      YLEN=13
      if (ELEMNT .eq. 1) then
         PTITLE='POWER TRANSISTOR #3 VOLTAGE DROP (RELAY)'
         PLEN=41.
      elseif (ELEMNT .eq. 3) then
         PTITLE='POWER TRANSISTOR #3 VOLTAGE DROP (PWM)'
         PLEN=39.
      endif
      PTIT1='BRUSHLESS DC MOTOR'
      PLEN1=20.
C
   C ... Data Retrieval ...
   if (DRIVE .eq. 'A') then
      open(6, file='a:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
   elseif (DRIVE .eq. 'B') then
      open(6, file='b:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
   elseif (DRIVE .eq. 'C') then
      open(6, file='c:BLDCM3.VIR', status='OLD', access='SEQUENTIAL')
   endif
   do 841 PCTR=1,NPTS
      read(6,941) VD3D
      Y1(PCTR)=VD3D
   continue
941 format(30X,F15.7)
   close(6, status='KEEP')
C
   call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
   elseif (PLOPT .eq. '12') then
      XTITLE='TIME (seconds)'
      XLEN=-15
YTITLE='VD4D (volts)'
YLEN=13
if (ELEMNT .eq. 1) then
  PTITLE='POWER TRANSISTOR #4 VOLTAGE DROP (RELAY)'
  PLEN=41.
elseif (ELEMNT .eq. 3) then
  PTITLE='POWER TRANSISTOR #4 VOLTAGE DROP (PWM)'
  PLEN=39.
endif
PTITLE='BRUSHLESS DC MOTOR'
PLEN1=20.

C

... Data Retrieval ...

if (DRIVE .eq. 'A') then
  open(6,file='a:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
  open(6,file='b:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
  open(6,file='c:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
endif
do 842 PCTR=1,NPTS
  read(6,942) VD4D
  Y1(PCTR)=VD4D
842 continue
942 format(45X,F15.7)
close(6,status='KEEP')

C
call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)

C

elseif (FLOPT .eq. '13') then
  XTITLE='TIME (seconds)'
  XLEN=15
  YTITLE='VD5D (volts)'
  YLEN=13
if (ELEMNT .eq. 1) then
  PTITLE='POWER TRANSISTOR #5 VOLTAGE DROP (RELAY)'
  PLEN=41.
elseif (ELEMNT .eq. 3) then
  PTITLE='POWER TRANSISTOR #5 VOLTAGE DROP (PWM)'
  PLEN=39.
endif
PTITLE='BRUSHLESS DC MOTOR'
PLEN1=20.

C

... Data Retrieval ...

if (DRIVE .eq. 'A') then
  open(6,file='a:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'B') then
  open(6,file='b:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE .eq. 'C') then
  open(6,file='c:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
endif

do 843 PCTR=1,NPTS
   read(6,943) VD5D
   Y1(PCTR)=VD5D
943 format(60X,F15.7)
close(6,status='KEEP')
C
call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (PLOPT.eq.'14') then
   XTITLE='TIME (seconds)'
   XLEN=15
   YTITLE='VD6D (volts)'
   YLEN=13
   if (ELEMNT.eq.1) then
      PTITLE='POWER TRANSISTOR #6 VOLTAGE DROP (RELAY)'
      PLEN=41.
   elseif (ELEMNT .eq. 3) then
      PTITLE='POWER TRANSISTOR #6 VOLTAGE DROP (PWM)'
      PLEN=39.
   endif
   PTIT1='BRUSHLESS DC MOTOR'
   PLEN1=20.
C
... Data Retrieval ...
if (DRIVE.eq.'A') then
   open(6,file='a:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE.eq.'B') then
   open(6,file='b:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
elseif (DRIVE.eq.'C') then
   open(6,file='c:BLDCM3.VIR',status='OLD',access='SEQUENTIAL')
endif

do 844 PCTR=1,NPTS
   read(6,944) VD6D
   Y1(PCTR)=VD6D
944 format(75X,F15.7)
close(6,status='KEEP')
C
call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)
C
elseif (PLOPT.eq.'15') then
   XTITLE='TIME (seconds)'
   XLEN=15
   YTITLE='NODE 0 VOLTAGE (VDC)'
   YLEN=21
   PTITLE='CENTER CONNECTION VOLTAGE RESPONSE'
   PLEN=35.
   PTIT1='BRUSHLESS DC MOTOR'
   PLEN1=20.
C ...

\[ \text{Data Retrieval ...} \]

if (DRIVE .eq. 'A') then
  open(4, file='a:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
else if (DRIVE .eq. 'B') then
  open(4, file='b:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
else if (DRIVE .eq. 'C') then
  open(4, file='c:BLDCM1.VIR', status='OLD', access='SEQUENTIAL')
endif

do 845 PCTR=1,NPTS
  read(4,945) NODE
  Yl(PCTR)=NODE
  continue
945 format(60X,F15.7)
close(4, status='KEEP')

C

call GRAPH(XTIME,Y1,XORG,YORG,DISOPT)

C

C

else
  go to 410
endif

C

400 continue
C


C

return
end

C

....................................................
C

***** Output Simulation Specs to Printer ***
C

....................................................
C Dumps simulation specifications to printer.
C
C subroutine PRNOUT(BEGTIM,FINTIM,PTSPLT,MAXITS,SIM2PL,DELTIM,
+ NTERMS,E0,EDOTO,TYPE,STPMAG,RSLOPE,SINAMP,SINFRQ,
+ SINPHA,NLCHAR,DBAND,PERIOD,KPWM,RIN,ROUT,RADJ,
+ KTM,KBM,KV,KP,RA,LA,BM,BL,JM,JL,RSAT,RCUT,TTIME,
+ RS,THADV,TDPLUG)

implicit REAL*4 (A-Z)
integer*2 SIM2PL,NTERMS
character*6 TYPE
character*25 NLCHAR

C ... Printer ready?? ...

call CLRSCR
call GOTOXY(12,25)
write(*,*) 'Please ensure PRINTER is ready'
call GOTOXY(20,1)
PAUSE

C ... Output Simulation Options to Printer ...

open(8,file='prn',status='new')

C write(8,1600) BEGTIM,FINTIM,PTSPLT,MAXITS,SIM2PL,DELTIM,
+ NTERMS,E0,EDOTO
write(8,1601) TYPE,STPMAG,RSLOPE,SINAMP,SINFRQ,SINPHA,
+ NLCHAR,DBAND,PERIOD,KPWM,RIN,ROUT,RADJ
write(8,1602) KTM,KBM,KV,KP,RA,LA,BM,BL,JM,JL,RSAT,
+ RCUT,TTIME,RS,THADV,TDPLUG
write(8,1603)

C 1600 format(/,18X,'*** BRUSHLESS DC MOTOR SIMULATION SPECS ***',/,
+ 1X,F15.7,1X,'[BEGTIM] Start Time of Plotting Window',/,
+ 1X,F15.7,1X,'[FINTIM] Stop Time of Plotting Window',/,
+ 1X,F15.7,1X,'[PTSPLT] Points to be Plotted per Curve',/,
+ 1X,F15.7,1X,'[MAXITS] Max Number of Simulation Iterations',/,
+ 13X,I3,1X,  ' [SIM2PL] Ratio: Points Simulated/Plotted',/,
+ 1X,F15.7,1X,'[DELTIM] Simulation Step Size (seconds)',/,
+ 10X,I6,1X, ' [NTERMS] Total Number of Simulation Steps',/,
+ 1X,F15.7,1X, '[E0] Initial Fin Position ERROR (deg)',/,
+ 1X,F15.7,1X, '[EDOTO] Initial Fin Velocity ERROR (deg/s)',/)

C 1601 format(25X,'*** COMMAND INPUT SELECTION ***',/,
+ 1X,A15,1X, ' [TYPE] STEP, RAMP, or SINE Response',/,
+ 1X,F15.7,1X, ' [STPMAG] Commanded Position for STEP Response',/,
+ 1X,F15.7,1X, ' [RSLOPE] Slope of RAMP Function',/,
+ 1X,F15.7,1X, ' [SINAMP] Amplitude of SINE Function',/,
+ 1X,F15.7,1X, ' [SINFRQ] Frequency (deg/sec) of SINE Function',/,
+ 1X,F15.7,1X, ' [SINPHA] Phase Angle (deg) of SINE Function',/,
+ 27X,'*** CONTROLLER SELECTION ***',/,
+ 10X,'[CONTROLLER SELECTION] -> ',A30,/, 
+ 1X,F15.7,1X, '[DBAND] Deadband Applied to System Feedback',/,
+ 1X,F15.7,1X, '[PERIOD] Period of PWM Reference Cycle(sec)',/,
**Motors**

**Motor Amplifier Gain**, \( [\text{PWM Amplifier Gain}] \), \( [\text{RIN}] \)

**Motor Current Limiting Resistance (inside)**, \( [\text{Current Limiting Resistance (inside)}] \), \( [\text{ROADJ}] \)

**Motor Series R Added to Freewheeling Diodes**, \( [\text{Series R Added to Freewheeling Diodes}] \), \( [\text{PWM}] \)

**Motor Current Limiting Resistance (outside)**, \( [\text{Current Limiting Resistance (outside)}] \), \( [\text{ROUT}] \)

**Motor Current Limiting Resistance (inside)**, \( [\text{Current Limiting Resistance (inside)}] \), \( [\text{RDADJ}] \)

**Motor Input Current Limiting Resistance (inside)**, \( [\text{Current Limiting Resistance (inside)}] \), \( [\text{RDADJ}] \)

**Motor Input Current Limiting Resistance (outside)**, \( [\text{Current Limiting Resistance (outside)}] \), \( [\text{RIN}] \)

**Motor Phase Resistance (Ohm)**, \( [\text{Motor Phase Resistance (Ohm)}] \), \( [\text{RA}] \)

**Motor Phase Resistance (Ohm)**, \( [\text{Motor Phase Resistance (Ohm)}] \), \( [\text{RA}] \)

**Motor Phase Inductance (Henries)**, \( [\text{Motor Phase Inductance (Henries)}] \), \( [\text{LA}] \)

**Motor Viscous Friction Coeff**.**, [\text{Motor Viscous Friction Coeff}].**, \( [\text{BM}] \)

**Motor Viscous Friction Coeff**.**, [\text{Motor Viscous Friction Coeff}].**, \( [\text{BL}] \)

**Motor Inertia (oz-in/s^2)**, \( [\text{Motor Inertia (oz-in/s^2)}] \), \( [\text{JM}] \)

**Motor Inertia (oz-in/s^2)**, \( [\text{Motor Inertia (oz-in/s^2)}] \), \( [\text{JM}] \)

**Equiv Ckt Resistance of Trans Satur**.**, [\text{Equiv Ckt Resistance of Trans Satur}].**, \( [\text{RSAT}] \)

**Equiv Ckt Resistance of Trans Satur**.**, [\text{Equiv Ckt Resistance of Trans Satur}].**, \( [\text{RSAT}] \)

**Equiv Ckt Resistance of Trans Satur**.**, [\text{Equiv Ckt Resistance of Trans Satur}].**, \( [\text{RSAT}] \)

**Equiv Ckt Resistance of Trans Cutoff**.**, [\text{Equiv Ckt Resistance of Trans Cutoff}].**, \( [\text{RCUT}] \)

**Equiv Ckt Resistance of Trans Cutoff**.**, [\text{Equiv Ckt Resistance of Trans Cutoff}].**, \( [\text{RCUT}] \)

**Equiv Ckt Resistance of Trans Cutoff**.**, [\text{Equiv Ckt Resistance of Trans Cutoff}].**, \( [\text{RCUT}] \)

**Transistor Switching Time**.**, [\text{Transistor Switching Time}].**, \( [\text{TTIME}] \)

**Transistor Switching Time**.**, [\text{Transistor Switching Time}].**, \( [\text{TTIME}] \)

**Transistor Switching Time**.**, [\text{Transistor Switching Time}].**, \( [\text{TTIME}] \)

**Load Viscous Friction Coeff**.**, [\text{Load Viscous Friction Coeff}].**, \( [\text{BM}] \)

**Load Viscous Friction Coeff**.**, [\text{Load Viscous Friction Coeff}].**, \( [\text{BL}] \)

**Load Viscous Friction Coeff**.**, [\text{Load Viscous Friction Coeff}].**, \( [\text{BL}] \)

**Load Inertia (oz-in/s^2)**, \( [\text{Load Inertia (oz-in/s^2)}] \), \( [\text{JL}] \)

**Load Inertia (oz-in/s^2)**, \( [\text{Load Inertia (oz-in/s^2)}] \), \( [\text{JL}] \)

**Load Inertia (oz-in/s^2)**, \( [\text{Load Inertia (oz-in/s^2)}] \), \( [\text{JL}] \)

**Internal Resist of Supply Voltage**.**, [\text{Internal Resist of Supply Voltage}].**, \( [\text{RS}] \)

**Internal Resist of Supply Voltage**.**, [\text{Internal Resist of Supply Voltage}].**, \( [\text{RS}] \)

**Internal Resist of Supply Voltage**.**, [\text{Internal Resist of Supply Voltage}].**, \( [\text{RS}] \)

**Commutation Advance (Electrical Deg)**.**, [\text{Commutation Advance (Electrical Deg)}].**, \( [\text{THADV}] \)

**Commutation Advance (Electrical Deg)**.**, [\text{Commutation Advance (Electrical Deg)}].**, \( [\text{THADV}] \)

**Commutation Advance (Electrical Deg)**.**, [\text{Commutation Advance (Electrical Deg)}].**, \( [\text{THADV}] \)

**Commutation Dead Time while Plugging**.**, [\text{Commutation Dead Time while Plugging}].**, \( [\text{TDPLUG}] \)

**Commutation Dead Time while Plugging**.**, [\text{Commutation Dead Time while Plugging}].**, \( [\text{TDPLUG}] \)

**Commutation Dead Time while Plugging**.**, [\text{Commutation Dead Time while Plugging}].**, \( [\text{TDPLUG}] \)

**Commutation Dead Time while Plugging**.**, [\text{Commutation Dead Time while Plugging}].**, \( [\text{TDPLUG}] \)

---

**C**

**Last Revision --> 04 April 1987**

**LT Vincent S. Rositto, USN**

**C**

**PLOTTING SUBROUTINES**

**C**

**Subroutine GRAPH(X,Y,XORG,YORG,DISOPT)**

**C**

**Implicit REAL*4 (A-Z)**

**COMMON BEGTM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,PLEN,PLEN1, XTITLE,YTITLE,PTITLE,PTIT1**

**REAL*4 X(1010),Y(1010)**

**INTEGER*2 NPTS,IOPORT,MODEL,XLEN,YLEN,NCHAR,NCHAR1**

**CHARACTER*3 DISOPT,ANS**

**CHARACTER*25 XTITLE,YTITLE**
C... Make a new title...

5 call CLRSCR
if (((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
    call GOTOXY(8,30)
    write(*,*), 'Current Title is:'
    call GOTOXY(10,20)
    write(*,*), PTITLE
    call GOTOXY(11,20)
    write(*,*), PTIT1
    call GOTOXY(15,25)
    write(*,'(A)') 'Do you want to change the title?'
    read(*,'(A)') ANS
    call CLRSCR
if (((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
    call GOTOXY(5,10)
    write(*,*), 'Enter titles in left justified format'
    call GOTOXY(12,10)
    write(*,*), '1234567890123456789012345678901234567890'
    call GOTOXY(13,10)
    write(*,*), '1234567890123456789012345678901234567890'
    call GOTOXY(21,10)
    write(*,*), '1234567890123456789012345678901234567890'
    call GOTOXY(23,25)
    write(*,*), 'Number of characters -->'
    call GOTOXY(15,46)
    read(*,'(A51)') PTITLE
    call GOTOXY(15,46)
    read(*,*), PLEN
    call GOTOXY(19,11)
    read(*,'(A51)') PTIT1
    call GOTOXY(23,46)
    read(*,*), PLEN1
elseif ((ANS .eq. 'N') .or. (ANS .eq. 'n')) then
    go to 10
endif
10 go to 5
endif

C

10 call GOTOXY(10,25)
write(*,*), 'Calculating Plotting Data'

C

ASPRAT=.65
CHARHT=.22
PTX=.5+(6.-PLEN*ASPRAT*CHARHT)/2.
PTY=4.5
PTX1 = .5*(6.-PLEN1*ASPRAT*CHARHT)/2.
PTY1 = 4.1
NCHAR = ifix(PLEN)
NCHAR1 = ifix(PLEN1)

C

call PLOTS(0, IPORT, MODEL)
call FACTOR(1.00)
call ASPECT(ASPRAT)
C ... Draw a Border ...
if ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
  call PLOT(XORG, YORG, -13)
call PLOT(8.0, 0.0, 0.2)
call PLOT(8.0, 6.0, 0.2)
call PLOT(0.0, 6.0, 0.2)
call PLOT(0.0, 0.0, 0.2)
end if

C
C call SCALE(X, 6., NPTS, 1)
call SCALE(Y, 4., NPTS, 1)
call STAXIS(.15, .22, 0.12, 0.080, 3)
C ... This scaling applies when the X axis represents Time...
X(NPTS+1) = BEGTIM
FIRSTX = X(NPTS+1)
X(NPTS+2) = (X(NPTS) - X(NPTS+1))/6.
DELTAX = X(NPTS+2)
FIRSTY = Y(NPTS+1)
DELTAY = Y(NPTS+2)
call PLOT(1.25, 1.0, -13)
call AXIS(0.0, 0.0, XTITLE, XLEN, 6.0, 0.0, FIRSTX, DELTAX)
call STAXIS(.15, .22, 12.0, .080, 3)
call AXIS(0.0, , YTITLE, YLEN, 4.90, 0.0, FIRSTY, DELTAY)
call SYMBOL(PTX, PTY, CHARHT, PTITLE, 0.0, NCHAR)
call SYMBOL(PTX1, PTY1, CHARHT, PTITLE1, 0.0, NCHAR1)
call LINE(X, Y, NPTS, 1, 0, 0)
call PLOT(0.0, 0.999)

C

MODEL = 99
IPORT = 99
C

return
end

C

**********************************************************************

C ***** PLOTTING SUBROUTINES *****
C ***** (multi-function plot)*****
C **********************************************************************

Subroutine MGRAPH(X, Y, Z, XORG, YORG, DISOPT)
C
implicit REAL*4 (A-Z)
COMMON BEGTIM, FINTIM, NPTS, IPORT, MODEL, XLEN, YLEN, PLEN, PLEN1,
     +    XTITLE, YTITLE, PTITLE, PTITLE1

223
real*4 X(1010),Y(1010),Z(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,NCHAR,NCHAR1
character*3 DISOPT,ANS
character*25 XTITLE,YTITLE
character*51 PTITLE,PTIT1

C
ASPRAT=.65
CHARHT=.22
CHARH1=.20
C
... Make a new title...
5 call CLRSCR
if (((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
   call GOTOXY(8,30)
   write(*,*) 'Current Title is:'
   call GOTOXY(10,20)
   write(*,*) PTITLE
   call GOTOXY(11,20)
   write(*,*) PTIT1
   call GOTOXY(15,25)
   write(*,'(A)') 'Do you want to change the title?'
   read(*,'(A)') ANS
   call CLRSCR
   if ((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
      call GOTOXY(5,10)
      write(*,*) 'Enter titles in left justified format'
      call GOTOXY(12,10)
      write(*,*) '1234567890123456789012345678901234567890'
      call GOTOXY(13,10)
      write(*,*) '1 2 3 4 5'
      call GOTOXY(15,25)
      write(*,*) '# of characters ->'
      call GOTOXY(20,10)
      write(*,*) '1234567890123456789012345678901234567890'
      call GOTOXY(21,10)
      write(*,*) '1 2 3 4 5'
      call GOTOXY(23,25)
      write(*,*) '# of characters ->'
      call GOTOXY(11,11)
      read(*,'(A51)') PTITLE
      call GOTOXY(15,46)
      read(*,*) PLEN
      call GOTOXY(19,11)
      read(*,'(A51)') PTIT1
      call GOTOXY(23,46)
      read(*,*) PLEN1
   elseif ((ANS .eq. 'N') .or. (ANS .eq. 'n')) then
      go to 10
   endif
endif
  go to 5
endif
10 call GOTOXY(10,25)
write(*,*) 'Calculating Plotting Data'

PTX=.5+(6.-PLEN*ASPRAT*CHARHT)/2.
PTY=6.5
PTX1=.5+(6.-PLEN1*ASPRAT*CHARHT1)/2.
PTY1=4.1
NCHAR=ifix(PLEN)
NCHAR1=ifix(PLEN1)

call PLOTS(0,IOPORT,MODEL)
call FACTOR(1.00)
call ASPECT(ASPRAT)

... Draw a Border ...
if ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
   call PLOT(XORG,YORG,-13)
   call PLOT(8.0,0.0,2)
   call PLOT(8.0,6.0,2)
   call PLOT(0.0,6.0,2)
   call PLOT(0.0,0.0,2)
endif

... This scaling applies when the X axis represents Time...
X(NPTS+1)=BEGTIM
FIRSTX = X(NPTS+1)
X(NPTS+2)=(X(NPTS)-X(NPTS+1))/6.
DELTAX = X(NPTS+2)

call SCALE(Y,4.,NPTS,1)
call SCALE(Z,4.,NPTS,1)
if (Z(NPTS+2) .gt. Y(NPTS+2)) then
   Y(NPTS+2)=Z(NPTS+2)
else
   Z(NPTS+2)=Y(NPTS+2)
endif
FIRSTY=Y(NPTS+1)
DELTAY = Y(NPTS+2)

call PLOT(1.25,1.,-13)
call STAXIS(.15,.22,.12,.080,3)
call AXIS(0.0,0.0,0,XTITLE,XLEN,6.,0.,FIRSTX,DELTAX)
call STAXIS(.15,.22,.12,.080,2)
call AXIS(0.0,0.0,0,YTITLE,YLEN,4.,90.,FIRSTY,DELTAY)
call SYMBOL(PTX,PTY,CHARHT,PTITLE,0.,NCHAR)
call SYMBOL(PTX1,PTY1,CHARHT1,PTITLE1,0.,NCHAR1)
call LINE(X,Y,NPTS,1,0,0)
call CURVE(X,Z,NPTS,-.1)
call PLOT(0.0,999)
MODEL=99
IOPORT=99

C
return
end

C
*******************************************************************************
C
***** PULSE WIDTH MODULATOR MODULE *****
*******************************************************************************

C
Subroutine PWMOD(TIME,NUMIT,TSTART,PERIOD,TOGGLE,POSERR,
+ DBAND,VIF,VIB,DIR,VREF,THRESH,KPWM)

C
IMPLICIT REAL*4 (A-Z)
INTEGER*2 NUMIT,DIR
LOGICAL*2 WAITNG,TOGGLE

C... Reset the saw-tooth reference signal ...
    if (TIME .ge, TSTART+PERIOD) then
      TSTART=TSTART+PERIOD
      TOGGLE=.true.
    endif

C
    if (POSERR .gt. (0.+DBAND)) then
      DIR=1
      ERROR=abs(KPWM*(POSERR-DBAND))
    elseif (POSERR .lt. (0.-DBAND)) then
      DIR=-1
      ERROR=abs(KPWM*(POSERR+DBAND))
    else
      ERROR=0.
    endif
    call LIMIT(0.,1.,ERROR,ERROR)
    call RAMP(TIME,TSTART,NREF)
    VREF=NREF/PERIOD
    THRESH=1.-ERROR

C... "WAITNG" is a logical variable indicating whether or not a new pulse may be generated ...
    if (NUMIT .eq. 1) WAITNG=TOGGLE

C
    if (WAITNG) then
      if (VREF .gt. THRESH) then
        if (DIR .eq. 1) then
          VIF=75.
          VIB=-75.
          TOGGLE=.false.
        elseif (DIR .eq. -1) then
          VIF=75.
          VIB=-75.
          TOGGLE=.false.
      endif
endif
elseif (VREF lt. THRESH) then
  VIF=0.
  VIB=0.
  TOGGLE=.true.
endif

return
end

***************************************************************************
***** RELAY MODULE *****
***************************************************************************

Subroutine RELAY(POSERR, DBAND, VIF, VIB, DIR)

implicit REAL*4 (A-Z)
integer*2 DIR

if (POSERR .gt. DBAND) then
  VIF=75.
  VIB=-75.
  DIR=1
elseif (POSERR .lt. -DBAND) then
  VIF=75.
  VIB=-75.
  DIR=1
elseif (abs(POSERR) .le. DBAND) then
  VIF=0.
  VIB=0.
endif

return
end

***************************************************************************
***** Commutation Advance Subroutine *****
***************************************************************************

Subroutine COMADV(THETA, PI, THADV, THCON, THRST)

implicit REAL*4 (A-Z)
THDEG=THETA*(180.0/PI)
THRST=AMOD(THDEG,360.0)
if (THRST .LT. 0.0) THRST=THRST+360.
THCON=THRST+THADV

return
end

***************************************************************************
***** Hall Sensor Positioning Subroutine *****
Subroutine HALL(THCON,SE1,SE2,SE3)

implicit REAL*4 (A-Z)

if (THCON .ge. 180.0) THCON=THCON-180.
if ((THCON .ge. 0.0) .and. (THCON .lt. 30.0)) go to 10
if ((THCON .ge. 30.0) .and. (THCON .lt. 60.0)) go to 11
if ((THCON .ge. 60.0) .and. (THCON .lt. 90.0)) go to 12
if ((THCON .ge. 90.0) .and. (THCON .lt. 120.0)) go to 13
if ((THCON .ge. 120.0) .and. (THCON .lt. 150.0)) go to 14
if ((THCON .ge. 150.0) .and. (THCON .lt. 180.0)) go to 15
10 SE1=1.
    SE2=0.
    SE3=1.
    go to 20
11 SE1=1.
    SE2=0.
    SE3=0.
    go to 20
12 SE1=1.
    SE2=1.
    SE3=0.
    go to 20
13 SE1=0.
    SE2=1.
    SE3=0.
    go to 20
14 SE1=0.
    SE2=1.
    SE3=1.
    go to 20
15 SE1=0.
    SE2=0.
    SE3=1.
20   continue
return
end

***********************************************************************
***** General Commutation Subroutine *****
***********************************************************************

Subroutine TRANS(TIME,REVTIM,BEMFA,BEMFB,BEMFC,BEMFT,VEMFA,
+ VEMFB,VEMFC,VIB,VIF,IM,IMA,IMB,IMC,VN1,VN2,SW1,
+ SW2,SW3,SW4,SW5,SW6,THCON,THCON1,RSAT,POSIT,DIR,
+ VD1D,VD2D,VD3D,VD4D,VD5D,VD6D,RIN,ROUT,RS,
+ REQA1,REQA2,REQB1,REQB2,REQC1,REQC2,REQD,RA,IBC,
+ ICA,PVAO,PVBO,PVCO,NODE,TRIG1,TRIG2,TRIG3,
+ TRIG4,TRIG5,TRIG6,PLUG,VAIND,VBIND,VCIND)

implicit REAL*4 (A-Z)

integer*2 POSIT,DIR,TRIG1,TRIG2,TRIG3,TRIG4,TRIG5,TRIG6,PLUG

THCON1=amod(THCON,180.0)
if (THCON1 < 0.0) THCON1 = THCON1 + 180.

IM = \frac{(\text{abs}(IMA) + \text{abs}(IMB) + \text{abs}(IMC))}{2}.

if (TRIG1 = 1) then
  FVACX = \text{abs}(VAIND) + IMA \times (RA + RIN/2.)
elseif (TRIG2 = 1) then
  FVACX = -\text{abs}(VAIND) + IMA \times (RA + RIN/2.)
else
  FVACX = VAIND + IMA \times (RA + RIN/2.)
endif

if (TRIG3 = 1) then
  PVBOX = \text{abs}(VBIND) + IMB \times (RA + RIN/2.)
elseif (TRIG4 = 1) then
  PVBOX = -\text{abs}(VBIND) + IMB \times (RA + RIN/2.)
else
  PVBOX = VBIND + IMB \times (RA + RIN/2.)
endif

if (TRIG5 = 1) then
  FVCOX = \text{abs}(VCIND) + IMC \times (RA + RIN/2.)
elseif (TRIG6 = 1) then
  FVCOX = -\text{abs}(VCIND) + IMC \times (RA + RIN/2.)
else
  FVCOX = VCOIND + IMC \times (RA + RIN/2.)
endif

if ((DIR = 1) .and. (PLUG = 0)) then
  if ((THCON1 > 0.0) .and. (THCON1 < 30.0)) POSIT = 1
  if ((THCON1 > 30.0) .and. (THCON1 < 60.0)) POSIT = 2
  if ((THCON1 > 60.0) .and. (THCON1 < 90.0)) POSIT = 3
  if ((THCON1 > 90.0) .and. (THCON1 < 120.0)) POSIT = 4
  if ((THCON1 > 120.0) .and. (THCON1 < 150.0)) POSIT = 5
  if ((THCON1 > 150.0) .and. (THCON1 < 180.0)) POSIT = 6
elseif ((DIR = -1) .and. (PLUG = 0)) then
  if ((THCON1 > 0.0) .and. (THCON1 < 30.0)) POSIT = 4
  if ((THCON1 > 30.0) .and. (THCON1 < 60.0)) POSIT = 5
  if ((THCON1 > 60.0) .and. (THCON1 < 90.0)) POSIT = 6
  if ((THCON1 > 90.0) .and. (THCON1 < 120.0)) POSIT = 1
  if ((THCON1 > 120.0) .and. (THCON1 < 150.0)) POSIT = 2
  if ((THCON1 > 150.0) .and. (THCON1 < 180.0)) POSIT = 3
endif

if (POSIT = 1) then

C
C ( NODE is the WYE center point floating voltage )
NODE = -(FVCOX + PVBOX) + float(DIR) \times (VEMFC - VEMFB)
if (PLUG = 0) then
  SW1 = 0.
  SW2 = 0.
  SW3 = 0.

229
SW4=1.
SW5=1.
SW6=0.

IAB=0.
ICA=0.
BEMFT=BEMFC-BEMFB
VN1=VIF-VEMFC
VN2=VIB-VEMFB

VD1D=(VIF-IM*(RS+ROUT)/2.)-(FVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

VD2D=(FVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

VD4D=-IMB*REQB2
if (VD4D .le. -.6) TRIG4 = 1

VD5D=IMC*REQC1
if (VD5D .le. -.6) TRIG5 = 1

VD6D=(FVCOX+VEMFC+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1

elseif (PLUG .eq. 1) then
SW1=0.
SW2=0.
SW3=0.
SW4=1.
SW5=0.
SW6=0.

IAB=0.
IBC=0.
ICA=0.
BEMFT=BEMFC-BEMFB
VN1=VIF-VEMFC
VN2=VIB-VEMFB

VD1D=(VIF-IM*(RS+ROUT)/2.)-(FVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

VD2D=(FVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1
C

VD4D=(PVBOX+VEMFB+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1
C

VD5D=(VIF-IM*(RS+ROUT)/2.)-(PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1
C

VD6D=(PVCOX+VEMFC+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1
C

endif
C

elseif (POSIT .eq. 2) then
C

NODE=-(PVAOX+PVBOX)+float(DIR)*(VEMFB-VEMFA)
if (PLUG .eq. 0) then
SW1=1.
SW2=0.
SW3=0.
SW4=1.
SW5=0.
SW6=0.
C

IBC=0.
ICA=0.
BEMFT=BEMFA-BEMFB
VN1=VIF-VEMFA
VN2=VIB-VEMFB
C

VD1D=IMA*REQA1
if (VD1D .le. -.6) TRIG1 = 1
C

VD2D=(PVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1
C

VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1
C

VD4D=IMB*REQB2
if (VD4D .le. -.6) TRIG4 = 1
C

VD5D=(VIF-IM*(RS+ROUT)/2.)-(PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1
C

VD6D=(PVCOX+VEMFC+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1
C

elseif (PLUG .eq. 1) then
SW1=0.
SW2=0.
SW3=0.
SW4 = 1.
SW5 = 0.
SW6 = 0.

C

IAB = 0.
IBC = 0.
ICA = 0.
BEMFT = BEMFA - BEMFB
VN1 = VIF - VEMFA
VN2 = VIB - VEMFB

C

VD1D = (VIF - IM*(RS + ROUT)/2.) - (PVAOX + VEMFA + NODE)
if (VD1D .le. -.6) TRIG1 = 1

C

VD2D = (PVAOX + VEMFA + NODE) - (VIB + IM*(RS + ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

C

VD3D = (VIF - IM*(RS + ROUT)/2.) - (PVCOX + VEMFB + NODE)
if (VD3D .le. -.6) TRIG3 = 1

C

VD4D = (PVCOX + VEMFB + NODE) - (VIB + IM*(RS + ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1

C

VD5D = (VIF - IM*(RS + ROUT)/2.) - (PVCOX + VEMFC + NODE)
if (VD5D .le. -.6) TRIG5 = 1

C

VD6D = (PVCOX + VEMFC + NODE) - (VIB + IM*(RS + ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1

endif

C

elseif (POSIT .eq. 3) then

NODE = -(PVAOX + PVCOX) + float(DIR) * (VEMFA - VEMFC)
if (PLUG .eq. 0) then

SW1 = 1.
SW2 = 0.
SW3 = 0.
SW4 = 0.
SW5 = 0.
SW6 = 1.
IAB = 0.
IBC = 0.
BEMFT = BEMFA - BEMFB
VN1 = VIF - VEMFA
VN2 = VIB - VEMFB

C

VD1D = IMA*REQA1
if (VD1D .le. -.6) TRIG1 = 1

C

VD2D = (PVAOX + VEMFA + NODE) - (VIB + IM*(RS + ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

C
VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

C
VD4D=(PVBOX+VEMFB+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1

C
VD5D=(VIF-IM*(RS+ROUT)/2.)-(FVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1

C
VD6D=IMC*REQC2
if (VD6D .le. -.6) TRIG6 = 1

C
elseif (PLUG .eq. 1) then
  SW1=0.
  SW2=0.
  SW3=0.
  SW4=0.
  SW5=0.
  SW6=1.
  IAB=0.
  IBC=0.
  ICA=0.
  BEMFT=BEMFA-BEMFC
  VN1=VIF-VEMFA
  VN2=VIB-VEMFC

C
VD1D=(VIF-IM*(RS+ROUT)/2.)-(PVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

C
VD2D=(PVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

C
VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

C
VD4D=(PVBOX+VEMFB+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1

C
VD5D=(VIF-IM*(RS+ROUT)/2.)-(PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1

C
VD6D=(PVCOX+VEMFC+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1

C
endif

C
elseif (POSIT .eq. 4) then

C
NODE=-(PVBOX+FVCOX)+float(DIR)*(VEMFC-VEMFB)
if (PLUG .eq. 0) then
SW1=0.
SW2=0.
SW3=1.
SW4=0.
SW5=0.
SW6=1.
C
IAB=0.
ICA=0.
BEMFT=BEMFB-BEMFC
VN1=VIF-VEMFB
VN2=VIB-VEMFC
C
VD1D=(VIF-IM*(RS+ROUT)/2.)-(FVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1
C
VD2D=(FVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1
C
VD3D=IMB*REQB1
if (VD3D .le. -.6) TRIG3 = 1
C
VD4D=(FVBAX+VEMFB+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1
C
VD5D=(VIF-IM*(RS+ROUT)/2.)-(PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1
C
VD6D=IMC*REQC2
if (VD6D .le. -.6) TRIG6 = 1
C
elseif (PLUG .eq. 1) then
SW1=0.
SW2=0.
SW3=0.
SW4=0.
SW5=0.
SW6=1.
C
IAB=0.
ICA=0.
BEMFT=BEMFB-BEMFC
VN1=VIF-VEMFB
VN2=VIB-VEMFC
C
VD1D=(VIF-IM*(RS+ROUT)/2.)-(FVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1
C
VD2D=(FVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
if (VD2D .le. -.6) TRIG2 = 1

234
if (VD2D .le. -.6) TRIG2 = 1

VD3D = (VIF - IM*(RS+ROUT)/2.) - (PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

VD4D = (PVBOX+VEMFB+NODE) - (VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1

VD5D = (VIF - IM*(RS+ROUT)/2.) - (PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1

VD6D = (PVCOX+VEMFC+NODE) - (VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1

endif

elseif (POSIT .eq. 5) then

NODE = (PVBOX+PVCOX)+float(DIR)*(VEMFB-VEMFA)
if (PLUG .eq. 0) then

SW1 = 0.
SW2 = 1.
SW3 = 1.
SW4 = 0.
SW5 = 0.
SW6 = 0.

IBC = 0.
ICA = 0.
BEMFT = BEMFB-BEMFA
VN1 = VIF - VEMFB
VN2 = VIB - VEMFA

VD1D = (VIF - IM*(RS+ROUT)/2.) - (PVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

VD2D = -IMA*REQA2
if (VD2D .le. -.6) TRIG2 = 1

VD3D = IMB*REQB1
if (VD3D .le. -.6) TRIG3 = 1

VD4D = (PVBOX+VEMFB+NODE) - (VIB+IM*(RS+ROUT)/2.)
if (VD4D .le. -.6) TRIG4 = 1

VD5D = (VIF - IM*(RS+ROUT)/2.) - (PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1

VD6D = (PVCOX+VEMFC+NODE) - (VIB+IM*(RS+ROUT)/2.)
if (VD6D .le. -.6) TRIG6 = 1

235
elseif (PLUG .eq. 1) then
    SW1=0.
    SW2=1.
    SW3=0.
    SW4=0.
    SW5=0.
    SW6=0.
    IAB=0.
    IBC=0.
    ICA=0.
    BEMFT=BEMFB-BEMFA
    VN1=VIF-VEMFB
    VN2=VIB-VEMFA
    
    VD1D=(VIF-IM*(RS+ROUT)/2.)-(PVAOX+VEMFA+NODE)
    if (VD1D .le. -.6) TRIG1 = 1
    
    VD2D=(PVAOX+VEMFA+NODE)-(VIB+IM*(RS+ROUT)/2.)
    if (VD2D .le. -.6) TRIG2 = 1
    
    VD3D=(VIF-IM*(RS+ROUT)/2.)-(PVBOX+VEMFB+NODE)
    if (VD3D .le. -.6) TRIG3 = 1
    
    VD4D=(PVBOX+VEMFB+NODE)-(VIB+IM*(RS+ROUT)/2.)
    if (VD4D .le. -.6) TRIG4 = 1
    
    VD5D=(VIF-IM*(RS+ROUT)/2.)-(PVCOX+VEMFC+NODE)
    if (VD5D .le. -.6) TRIG5 = 1
    
    VD6D=(PVCOX+VEMFC+NODE)-(VIB+IM*(RS+ROUT)/2.)
    if (VD6D .le. -.6) TRIG6 = 1

endif

elseif (POSIT .eq. 6) then

    NODE=(PVCOX+PVAOX)+float(DIR)*(VEMFA-VEMFC)
    if (PLUG .eq. 0) then
        SW1=0.
        SW2=1.
        SW3=0.
        SW4=0.
        SW5=1.
        SW6=0.
        IAB=0.
        IBC=0.
        BEMFT=BEMFC-BEMFA
        VN1=VIF-VEMFC
        VN2=VIB-VEMFA

endif
VD = (VIF - IM*(RS+ROUTE)/2.) - (PVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

VD2D = -IMA*REQA2
if (VD2D .le. -.6) TRIG2 = 1

VD3D = (VIF - IM*(RS+ROUTE)/2.) - (PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

VD4D = (PVBOX+VEMFB+NODE) - (VIB+IM*(RS+ROUTE)/2.)
if (VD4D .le. -.6) TRIG4 = 1

VD5D = IMC*REQC1
if (VD5D .le. -.6) TRIG5 = 1

VD6D = (PVCOX+VEMFC+NODE) - (VIB+IM*(RS+ROUTE)/2.)
if (VD6D .le. -.6) TRIG6 = 1

elseif (PLUG .eq. 1) then
SW1 = 0.
SW2 = 1.
SW3 = 0.
SW4 = 0.
SW5 = 0.
SW6 = 0.

IAB = 0.
IBC = 0.
ICA = 0.
BEMFT = BEMFC-BEMFA
VN1 = VIF-VEMFC
VN2 = VIB-VEMFA

VD1D = (VIF - IM*(RS+ROUTE)/2.) - (PVAOX+VEMFA+NODE)
if (VD1D .le. -.6) TRIG1 = 1

VD2D = (PVBOX+VEMFB+NODE) - (VIB+IM*(RS+ROUTE)/2.)
if (VD2D .le. -.6) TRIG2 = 1

VD3D = (VIF - IM*(RS+ROUTE)/2.) - (PVBOX+VEMFB+NODE)
if (VD3D .le. -.6) TRIG3 = 1

VD4D = (PVBOX+VEMFB+NODE) - (VIB+IM*(RS+ROUTE)/2.)
if (VD4D .le. -.6) TRIG4 = 1

VD5D = (VIF - IM*(RS+ROUTE)/2.) - (PVCOX+VEMFC+NODE)
if (VD5D .le. -.6) TRIG5 = 1

VD6D = (PVCOX+VEMFC+NODE) - (VIB+IM*(RS+ROUTE)/2.)
if (VD6D .le. -.6) TRIG6 = 1

237
C
    endif
C
    endif
99 continue
return
end

C
*************************************************
C      ***** Cutoff-Saturation Limiting Subroutine *****
C
*************************************************

Subroutine LIMIT(RSAT,RCUT,INPUT,OUT)
    implicit REAL*4 (A-Z)
    if (INPUT .le. RSAT) then
        OUT=RSAT
    elseif (INPUT .ge. RCUT) then
        OUT=RCUT
    else
        OUT=INPUT
    end if
C
    return
end

C
*************************************************
C      ***** Function Switch Subroutine *****
C
*************************************************

Subroutine FCNSW(X1,X2,X3,X4,OUT)
    implicit REAL*4 (A-Z)
    if (X1 .lt. 0.0) then
        OUT=X2
    elseif (X1 .eq. 0.0) then
        OUT=X3
    else
        OUT=X4
    end if
C
    return
end

C
*************************************************
C      ***** Step function Subroutine *****
C
*************************************************

Subroutine STEP(TIME,TSTEP,OUT)
    implicit REAL*4 (A-Z)
    if (TIME .ge. TSTEP) then
        OUT=1.0
    else
        OUT=0.0
    end if
C
    return
end
C ***** Deadspace Subroutine *****
C
Subroutine DEADSP(P1,P2,VSGDEL,VSGERR)
  implicit REAL*4 (A-Z)
  if (VSGDEL .gt. P2) then
    VSGERR=VSGDEL-P2
  elseif (VSGDEL .lt. P1) then
    VSGERR=VSGDEL-P1
  else
    VSGERR=0.0
  end if
C
  return
end
C
C
C ***** Ramp Subroutine *****
C
Subroutine RAMP(TIME,TRAMP,OUT)
  implicit REAL*4 (A-Z)
  if (TIME .ge. TRAMP) then
    OUT=TIME-TRAMP
  else
    OUT=0.0
  end if
C
  return
end
C
C *** TIME CONSTANT ***
C
Subroutine TCONST(Y0,X,TAU,NTIME,NTIM,DELTIM,Y)
  implicit real*4 (A-Z)
  integer*2 NTIME,NTIM
  if (NTIME .eq. 1) Y=Y0
  if (NTIME .ne. NTIM) Y=0
  DECAY=exp(-DELTIM/TAU)
  Y=Y0+(X-Y0)*(1.-DECAY)
  if (NTIME .eq. 1) Y=Y0
  NTIM-NTIME
C
  return
end
C
C
C ***** First Order Derivative Subroutine *****
C ***** (Central Difference Method) *****
C
Subroutine DERIV(DELTIM,NTIME,NTIM1,IC2,XM1,NOWVAL,XX,XDM1,
  + XD,XDD1,XDD,SLOPE)
  implicit REAL*4 (A-Z)
  integer*2 NTIME,NTIM1
C
239
if (NTIME .eq. NTIM1) then
  XX=NOWVAL
else
  XM1=XX
  XX=NOWVAL
  XDM1=XD
  XDDM1=XDD
end if

XD=(XX-XM1)/DELTIM
if (abs(XD) .lt. 1.E-8) XD=0.
if (NTIME .eq. 1) XD=IC2
XDD=(XD-XDM1)/DELTIM
if (abs(XDD) .lt. 1.E-8) XDD=0.0

NTIM1=NTIME

XPRED=XX+XD*DELTIM+XDD*(DELTIM**2)/2.0
if (abs(XPRED) .lt. 1.E-8) XPRED=0.0
SLOPE=(XPRED-XM1)/(2.0*DELTIM)
if (abs(SLOPE) .lt. 1.E-8) SLOPE=0.0

return
end

***********************************************************************

***** Trapezoidal Integration Subroutine *****

**************************************************************************

Subroutine INTGRL(NTIME,NTIM2,DELTIM,IC3,PREVAL,NOWVAL,
+ CURVAL,OUTOLD,OUTNEW)
implicit REAL*4 (A-Z)
integer*2 NTIME,NTIM2

if ((NTIME .eq. NTIM2) .or. (NTIME .eq. 1)) then
  CURVAL=NOWVAL
else
  PREVAL = CURVAL
  CURVAL = NOWVAL
  OUTOLD = OUTNEW
end if
if (NTIME .eq. 1) OUTOLD=IC3
OUTNEW = OUTOLD+(CURVAL+PREVAL)*DELTIM/2.
NTM2=NTIME

return
end

***********************************************************************

***** CLEAR SCREEN AND HOME CURSOR *****

**************************************************************************

subroutine CLRSCR
character*1 C1,C2,C3,C4
integer*2 IC(4)
equivalence (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
data IC/16#1B,16#5B,16#32,16#4A/
C
C *** Write Escape Code to Display ***
write(*,1) C1,C2,C3,C4
1 format(1X,4A1)
C
return
end
C
C ***** Position Cursor by Row,Column *****
C
******************************************************************************
C subroutine GOTOXY(ROW,COLUMN)
integer*2 IC(4),ROW,COLUMN,L
character*1 C1,C2,C5,C8,LC(5)
character*5 CBUFF
equivalence (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),
+ (CBUFF,LC(1))
data IC/16#1B,16#5B,16#3B,16#66/
C
L=10000+100*ROW+COLUMN
C
C *** Write Escape Codes to a Character Buffer ***
write(CBUFF,2) L
2 format(I5)
C
C *** Write Escape Codes to Display ***
write(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
3 format(1X,8A1,\)
return
end
APPENDIX B

PWM OPEN LOOP TRANSFER CHARACTERISTICS

This version tests the pulse width modulator for logical correctness. POSERR is used as an input and VIN is the output.

IMPLICIT REAL*4 (A-Z)
COMMON BEGTIM,FINTIM,NPORT,MODEL,XLEN,YLEN,PLEN,PPLANE,
+ XTITLE,YTITLE,PTITLE
REAL*4 X(1010),Y1(1010),Y2(1010),Y3(1010),Y4(1010),LINEO(1010)
INTEGER*2 NTERMS,IOPORT,MODEL,XLEN,YLEN,NTIME,NUMIT,NTIM1,
+ NTIM2,NTIM3,NTIM4,DIR,MODEL,NPTS,NCTR,PPLANE,
+ SIM2PL,WAVE
LOGICAL*2 WAITING,TOGGLE
CHARACTER*1 DISOPT,PLOT
CHARACTER*6 ANS1,ANS11,ANS21,PRTSEL
CHARACTER*20 XTITLE,YTITLE
CHARACTER*51 PRTCHR,PTITLE

... Introductory Page (1 time good deal!)
call CLRSCR
write(*,4)
PAUSE

***************************************
*** OPEN/READ/CLOSE INPUT DATA FILE ***
***************************************

13 open(7,FILE='PWM.INP',STATUS='OLD',ACCESS='SEQUENTIAL')
read(7,1000) PMODEL,PRTCHR
read(7,1020) BEGTIM,FINTIM,MAXITS,SIM2PL
read(7,1022) DBAND,PERIOD
read(7,1028) KPWM,X0,Y0,WFACT
close(7,STATUS='KEEP')

***************************************
*** DISPLAY MAIN MENU SELECTIONS ***
***************************************

1 call CLRSCR
call GOTOXY(10,1)
write(*,5)
read(*,'(A)*') ANS1
C ... Hardware Options ...
if ((ANS1.eq.'h') .or. (ANS1.eq.'H')) then
C 101 call CLRSCR
call GOTOXY(17,24)
write(*,'*** CURRENT PRINTER SELECTION ***')
call GOTOXY(20,20)
write(*,' FRTCHR')
call GOTOXY(10,1)
write(*,105)
read(*,'(A)*') ANS11
C ... Printer Options ...
if ((ANS11.eq.'p') .or. (ANS11.eq.'P')) then
C 131 call CLRSCR
write(*,130)
read(*,'(A)*') PRTSEL
C
if (PRTSEL.eq.'0') then
   FRTCHR='Epson FX-80 Printer, single density'
   PMODEL=0
elseif (PRTSEL.eq.'1') then
   FRTCHR='Epson FX-80 Printer, double density'
   PMODEL=1
elseif (PRTSEL.eq.'2') then
   FRTCHR='Epson FX-80 Printer, dble spd,dual density'
   PMODEL=2
elseif (PRTSEL.eq.'3') then
   FRTCHR='Epson FX-80 Printer, quad density'
   PMODEL=3
elseif (PRTSEL.eq.'4') then
   FRTCHR='Epson FX-80 Printer, CRT Graphics I'
   PMODEL=4
elseif (PRTSEL.eq.'5') then
   FRTCHR='Epson FX-80 Printer, plotter graphics'
   PMODEL=5
elseif (PRTSEL.eq.'6') then
   FRTCHR='Epson FX-80 Printer, CRT Graphics II'
   PMODEL=6
elseif (PRTSEL.eq.'10') then
   FRTCHR='Epson FX-100 Printer, single density'
   PMODEL=7
elseif (PRTSEL.eq.'11') then
   FRTCHR='Epson FX-100 Printer, double density'
   PMODEL=11
elseif (PRTSEL.eq.'12') then
   FRTCHR='Epson FX-100 Printer, dble spd,dual density'
   PMODEL=12
elseif (PRTSEL.eq.'13') then
   FRTCHR='Epson FX-100 Printer, quad density'
243
elseif (PRTSEL .eq. '14') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics I'
  PMODEL=14
elseif (PRTSEL .eq. '15') then
  PRTCHR='Epson FX-100 Printer, plotter graphics'
  PMODEL=15
elseif (PRTSEL .eq. '16') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics II'
  PMODEL=16
elseif (PRTSEL .eq. '20') then
  PRTCHR='HP 7470A Graphics Plotter'
  PMODEL=20
elseif (PRTSEL .eq. '30') then
  PRTCHR='HP 7475A Graphics Plotter'
  PMODEL=30
elseif (PRTSEL .eq. '60') then
  PRTCHR='HP 2686A Laser Jet Printer'
  PMODEL=60

C ... Quit the Printer Menu ...
elseif ((PRTSEL .eq. 'Q') .or. (PRTSEL .eq. 'q')) then
  go to 101
else
  go to 131
endif
  go to 101

C ... Quit the Hardware Menu ...
elseif ((ANS11 .eq. 'q') .or. (ANS11 .eq. 'Q')) then
  go to 1
else
  go to 101
endif

C ... PWM Design Menu ...
elseif ((ANS1 .eq. 'p') .or. (ANS1 .eq. 'P')) then

C 203 call CLRSCR
write(*,250) KPWM,DBAND,PERIOD
read(*,'(A)') ANS21

if (ANS21 .eq. 'KPWM') then
  call GOTOXY(24,1)
  write(*,'#(A)')'Enter a REAL value for KPWM--> '
  read(*,*) KPWM
  go to 203
else if (ANS21 .eq. 'PERIOD') then
  call GOTOXY(24,1)
  write(*,'#(A)')'Enter a REAL value for PERIOD--> '
  read(*,*) PERIOD

244
go to 203
elseif (ANS21 .eq. 'DBAND') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for DBAND-->
read(*,*) DBAND
goto 203
end if
C
C ... Quit PWM Design Menu ...
elseif (ANS21 .eq. 'Q') then
go to 1
else
goto 203
end if
C
C ... Simulation Options ...
elif ((ANS1 .eq. 'O') .or. (ANS1 .eq. 'O')) then
C
202 call CLRSCR
C
DELTIM=(FINTIM-BEGTIM)/(1000.*SIM2PL)
if (FINTIM/DELTIM .gt. MAXITS) DELTIM =FINTIM/MAXITS
NTERMS=IFIX(FINTIM/DELTIM)+1
C
write(*,240) BEGTIM,FINTIM,MAXITS,SIM2PL,+
X0,Y0,WFACT,DELTIM,NTERMS
read(*,'(A)') ANS21
C
if (ANS21 .eq. 'BEGTIM') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for BEGTIM-->
read(*,*) BEGTIM
goto 202
elseif (ANS21 .eq. 'FINTIM') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for FINTIM-->
read(*,*) FINTIM
goto 202
elseif (ANS21 .eq. 'MAXITS') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for MAXITS-->
read(*,*) MAXITS
goto 202
elseif (ANS21 .eq. 'SIM2PL') then
call GOTOXY(24,1)
write(*,'(A)')'Enter an INTEGER value for SIM2PL-->
read(*,*) SIM2PL
goto 202
elseif (ANS21 .eq. 'X0') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for X0-->
read(*,*) X0

245
go to 202
elseif (ANS21 .eq. 'Y0') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for Y0-->
read(*,*) Y0
go to 202
endif

C ... Quit Simulation Options Menu ...
elseif (ANS21 .eq. 'Q') then
go to 1
else
go to 202
endif

C ... Input Waveform Options ...
elseif ((ANSI .eq. 'i') .or. (ANSI .eq. 'I')) then
204 call CLRSCR
C
write(*,260)
read(*,'(A)') ANSI
write(*,*)
read(*,'(A)') ANS21
write(*,**)
call CLRSCR
C

if (ANS21 .eq. 'I') then
call GOTOXY(10,1)
WAVE=1
write(*,'(A)') 'Enter a REAL value for K-->
read(*,*) CONST
write(*,'(A)') 'Enter a REAL value for W-->
read(*,*) W
call GOTOXY(17,4)
write(*,141) CONST,W
141 format(1X,'POSERR =',F8.3,'* SIN(',F8.3,'* t)')
PAUSE
go to 204
elseif (ANS21 .eq. '2') then
call GOTOXY(10,1)
WAVE=2
write(*,'(A)') 'Enter a REAL value for K-->
read(*,*) CONST
write(*,'(A)') 'Enter a REAL value for W-->
read(*,*) W
write(*,'(A)') 'Enter a REAL value for TAU-->
read(*,*) TAU
call GOTOXY(17,4)
write(*,142) CONST,TAU,W
142 format(1X,'POSERR =',F8.3,'* SIN(',F8.3,'* t)')
142 format(1X,'POSERR =',F8.3,* EXP(-t/F8.4,')*SIN(F8.3,* t')
   PAUSE
   go to 204
   elseif (ANS21 .eq. '3') then
      call GOTOXY(10,1)
      WAVE=3
      write(*,'(A)'),'Enter a REAL value for A-->
      read(*,*) A
      write(*,'(A)'),' Enter a REAL value for B-->
      read(*,*) B
      call GOTOXY(17,'4)
      write(*,143) A,B
   143 format(1X,'POSERR =',F8.3,* t +',F8.4)
   PAUSE
   go to 204
C ... Quit Simulation Options Menu ...
   elseif (ANS21 .eq. 'Q') then
      go to 1
   else
      go to 204
   endif
C ... Save Options to File ...
   elseif ((ANS1 .eq. 's') .or. (ANS1 .eq. 'S')) then
C
C ****************************************
C *** OPEN/READ/CLOSE INPUT DATA FILE ***
C ****************************************
C
C open(7,FILE='FWM.INP',STATUS='NEW')
write(7,1000)'PMODEL,PRTCDR
write(7,1020)BEGTIM,FINTIM,MAXITS,SIM2PL
write(7,1022)DBAND,PERIOD
write(7,1028)_KP,W,X0,Y0,WFAC
write(7,2000)
close(7,STATUS='KEEP')
C
   go to 1
C
C ... Run the Program ...
   elseif ((ANS1 .eq. 'r') .or. (ANS1 .eq. 'R')) then
   go to 2
C
C ... Quit the Program ...
   elseif ((ANS1 .eq. 'q') .or. (ANS1 .eq. 'Q')) then
      stop
   else
      go to 1
   endif
C
   2 PI=3.14159
C
... Initializations ...

NCTR=0
NPTS=0
TSTART=0.0
TOGGLE=.true.
PPLANE=0

Display the simulation header ...

call CLRSCR
call GOTOXY(10,29)
write(*,*) 'Simulation in Progress'

DELTIM=(FINTIM-BEGTIM)/(1000.*SIM2PL)
if (FINTIM/DELTIM .gt. MAXITS) DELTIM=FINTIM/MAXITS
NTERMS=IFIX(FINTIM/DELTIM)+1
call GOTOXY(14,1)
write(*,15) DELTIM,NTERMS

format(16X,'Simulation Step Size --> ',F8.7,' seconds',/,
+ 17X,'Total Number of Steps--> ',I4)
SPACE=DELTIM/2.0

Open/Write/Close OUTPUT data file ...

open(4,FILE='PWM.OUT',STATUS='NEW')
write(4,1200) BEG TIM,FINTIM,M A XITS,SIM2PL,PERIOD,DBAND,
+ ERRCUT,AGCCUT,ERRSAT,AGCSAT
close(4,STATUS='KEEP')

**********************************
*** START MAIN SIMULATION LOOP ***
**********************************

DO 100 NTIME=1,NTERMS
TIME=(NTIME-1)*DELTIM

Reset the sawtooth reference signal ...

if (TIME .ge. TSTART+PERIOD) then
  TSTART=TSTART+PERIOD
  TOGGLE=.true.
end if

if (WAVE .eq. 1) POSERR=CONST*SIN(W*TIME)
if (WAVE .eq. 2) POSERR=CONST*exp(-TIME/TAU)*SIN(W*TIME)
if (WAVE .eq. 3) POSERR=A*TIME+B

if (POSERR .gt. (0.+DBAND)) then
  ERROR=abs(KP*M*(POSERR-DBAND))
else if (POSERR .lt. (0.-DBAND)) then
  ERROR=abs(KP*M*(POSERR+DBAND))
else
  ERROR=0.
end if
call LIMIT(0.,1.,ERROR,ERROR)
call RAMP(TIME,TSTART,NREF)
VREF=NREF/PERIOD
THRESH=1.-ERROR
C
C ... "WAITNG" is a logical variable indicating whether or not a new
pulse may be generated ...
WAITNG=TOGGLE
C
if (WAITING) then
  if (VREF .gt. THRESH) then
    if (DIR .eq. 1) then
      VIN=150.
      TOGGLE=.false.
    elseif (DIR .eq. -1) then
      VIN=-150.
      TOGGLE=.false.
    endif
  else
    VIN=0.
    TOGGLE=.true.
  endif
endif
C
DIRLOG=VIN/150.
C
C ... Generate Plotting Arrays ...
if (TIME .ge. BEGTIM) then
  NCTR=NCTR+1
  if (mod(NCTR,SIM2PL) .eq. 0) then
    NPTS=NPTS+1
    X(NPTS)=TIME
    Y1(NPTS)=POSERR
    Y2(NPTS)=VREF
    Y3(NPTS)=THRESH
    Y4(NPTS)=DIRLOG
    LINE0(NPTS)=0.
  endif
endif
C
100 CONTINUE
C
C *****************************************
C *** END MAIN SIMULATION LOOP ***
C *****************************************
C
C ************************************************
C **** Plotting selection ****
C ************************************************
C
C *** Clear Screen & Home Cursor ***
400 call CLRSCR
C
write(*,1305)
read(*,'(A)') DISOPT
C
if (DISOPT .eq. '1') then
  MODEL=99
  IOPORT=99
elseif (DISOPT .eq. '2') then
  MODEL=PMODEL
  IOPORT=1
  if ((MODEL .eq. 20) .or. (MODEL .eq. 30)) IOPORT=9600
elseif (DISOPT .eq. '3') then
  GO TO 13
elseif ((DISOPT .eq. 'Q') .or. (DISOPT .eq. 'q')) then
  GO TO 460
else
  write('**') 'Incorrect display option selection, try again!'
goto 400
end if
C
410 call CLRSCR
if (DISOPT .eq. '2') then
  call GOTOXY(20,20)
  write('**') PRTCHR
  call GOTOXY(1,1)
endif
write(*,1300)
read(*,'(A)') PLOPT
C
if ((PLOPT .eq. 'Q') .or. (PLOPT .eq. 'q')) then
go to 400
elseif (PLOPT .eq. '1') then
  XTITLE='TIME (sec)'  
  XLEN=10
  YTITLE='PWM PERFORMANCE'
  YLEN=14
  PPLANE=0
  call MGRAPH(X,Y1,Y2,Y3,Y4,LINE0,DISOPT,KPWM,DBAND,X0,Y0,WFAC)
else
  go to 410
endif
goto 400
C
460 continue
C
*****************************
C  *** I/O FORMAT STATEMENTS ***
C ****************************
C
/* format///,24X,'PULSE WIDTH MODULATOR DESIGN AND ANALYSIS',/,

250
+ 26X,'Open Loop Response Characteristics',/, + 26X,'Automatic Gain Control Simulation',/, + 26X,'INTEL-8087 Math Co-processor version',///, + 10X,'NOTE: much of the screen control attained in this ', /, + 10X,' program is interactive with the DOS screen driver',/, + 10X,' ANSI.SYS. Ensure your CONFIG.SYS file contains',/, + 10X,' The input data file PWM.INP must be in the ', /, + 10X,' default disk drive. The output file PWM.OUT',/, + 10X,' will be written to the default drive.',//)

C
5 format(32X,*** MAIN MENU ***,///, + 20X,'[H]-----> HARDWARE Configuration Menu',/, + 20X,'[P]-----> PWM Design Menu',/, + 20X,'[I]-----> INPUT Waveform',/, + 20X,'[O]-----> OPTIONS for Simulation',/, + 20X,'[S]-----> SAVE All Changes',/, + 20X,'[R]-----> RUN Simulation Program',/, + 20X,'[Q]-----> QUIT the Program',/, + 8X,'ENTER SELECTION---->',\)
105 format(30X,*** HARDWARE MENU ***,///, + 20X,'[P]-----> PRINTER/PLOTTER configuration change',/, + 20X,'[Q]-----> QUIT THIS MENU',/, + 8X,'ENTER SELECTION---->',\)
1000 format(1X,I3,2X,A50) 1020 format(1X,3F12.7,1X,I3) 1022 format(1X,2F12.7) 1028 format(1X,4F12.7) 2000 format(1X,'END OF FILE')
C
Subroutine GRAPH(X,Y)

implicit REAL*4 (A-Z)
COMMON BEGTIM, FINTIM, NPTS, IOPORT, MODEL, XLEN, YLEN, PLEN, PPLANE,
+ XTITLE, YTITLE, PTITLE
real*4 X(1010), Y(1010)
integer*2 NPTS, IOPORT, MODEL, XLEN, YLEN, PPLANE, NCHAR
character*20 XTITLE, YTITLE
character*51 PTITLE

ASPRAT=.65
CHARHT=.22
PTX=1.5+(6.-PLEN*ASPRAT*CHARHT)/2.
PTY=5.5
NCHAR=ifix(PLEN)

call PLOTS(0, IOPORT, MODEL)
call FACTOR(1.00)
call ASPECT(ASPRAT)
call SCALE(X, 6., NPTS, 1)
call SCALE(Y, 4., NPTS, 1)
call STAXIS(.15,.22,.10,.080,3)
FIRSTX = X(NPTS+1)
if (PPLANE .eq. 1) then
  DELTAX=X(NPTS+2)
else
  X(NPTS+2)=(X(NPTS)-X(NPTS+1))/6.
  DELTAX = X(NPTS+2)
endif
FIRSTY = Y(NPTS+1)
DELTAY = Y(NPTS+2)
call AXIS(1.0,1.0,XTITLE, XLEN, 6.,0.,FIRSTX, DELTAX)
call STAXIS(.15,.22,.10,.080,2)
call AXIS(1.,1., YTITLE, YLEN, 4.,90.,FIRSTY, DELTAY)
call SYMBOL(PTX,PTY,CHARHT,PTITLE, 0., NCHAR)
if (PPLANE .eq. 1) then
  call SYMBOL(3.0,5.25,.18,'(',0.,1)
call NUMBER(999.,5.25,.18,BEGTIM,0.,4)
call SYMBOL(999.,5.25,.18,' - ',0.,3)
call NUMBER(999.,5.25,.18,FINTIM,0.,4)
call SYMBOL(999.,5.25,.18,' seconds'),0.,9)
endif
call PLOT(1.,1.,-13)
call LINE(X,Y,NPTS,1,0,0)
call PLOT(0.0,0.0,0.999)

MODEL=99
IOPORT=99

RETURN

END

*******************************************************************************
***** PLOTTING SUBROUTINES *****
*****(multi- function plot)*****
*******************************************************************************

Subroutine MGRAPH(X,Y5,Y6,Y7,Y8,LINE0,DISOPT,KPWM,DBAND,+
X0,Y0,WFAC)

implicit REAL*4 (A-Z)

COMMON BEGTIM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,PLEN,PPLANE,+
XTITLE,YTITLE,PTITLE

real*4 X(1010),Y5(1010),Y6(1010),Y7(1010),Y8(1010),LINE0(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,PPLANE

character*1 DISOPT

character*3 ANS

character*20 XTITLE,YTITLE

character*51 PTITLE

WFACT=.5

...Time axis ...

X(NPTS+1)=BEGTIM
FIRSTX = BEGTIM
X(NPTS+2)=(BEGTIM-FINTIM)/10.
DELTAX = (FINTIM-BEGTIM)/10.

call SCALE(Y5,4.,NPTS,1)
FIRSTY=Y5(NPTS+1)
DELTY=Y5(NPTS+2)
LINE0(NPTS+1)=Y5(NPTS+1)
LINE0(NPTS+2)=Y5(NPTS+2)
Y6(NPTS+1)=-1.0
Y6(NPTS+2)=-.2
Y7(NPTS+1)=-1.0
Y7(NPTS+2)=-.2
Y8(NPTS+1)=-12.0
Y8(NPTS+2)=1.

CALL FLOTS(0,IOPORT,MODEL)
CALL FACTOR(WFACT)
if (DISOPT .eq. '1') then
    call PLOT(2.5,1.,-13)
elseif (DISOPT .eq. '2') then
    call PLOT(X0/WFACT,Y0/WFACT,-13)

C ... Draw a border ...
call PLOT(16.0,0.0,2)

254
call PLOT(16.0,12.0,2)
call PLOT(0.0,12.0,2)
call PLOT(0.0,0.0,2)
C ... Redefine origin ...
call PLOT(1.3,11.0,-13)
endif
C
CALL STAXIS(.20,.27,.16,.080,2)
CALL AXIS(0.0,0.0 , 'TIME (sec)', -10,10,.270,.FIRSTX,DELTAX)
C
CALL STAXIS(.20,.27,.16,.080,1)
CALL AXIS(0.0 , 'ERROR SIGNAL',12,4,.0 ,FIRSY5,DELTY5)
CALL LINE(Y5,X,NPTS,1.0,0)
CALL CURVE(LINE0,X,NPTS,-.1)
C
CALL AXIS(5.0 , 'REFERENCE SIGNAL',16,5,.0,.0,.2)
CALL LINE(Y6,X,NPTS,1.0,0)
C
CALL AXIS(5.0,-10., 'THRESHOLD VOLTAGE',-17,5,.0,.0,.2)
CALL NEWPEN(2)
CALL LINE(Y7,X,NPTS,1.0,0)
C
CALL STAXIS(.20,.27,.13,.080,-1)
CALL AXIS(11.0 , 'DIRECTIONAL LOGIC',17,2,.0,-1,.1.)
CALL LINE(Y8,X,NPTS,1.0,0)
C
PTITLE='PWM OPEN LOOP PERFORMANCE'
call SYMBOL(14.2,-1.3,.35,PTITLE,270.,26)
call SYMBOL(13.6,-1.0,.3,'GAIN = ',270.,7).
call NUMBER(13.6,999,.3,KPWM,270.,2)
call SYMBOL(13.6,-5.0,.3,'DEAD ZONE = ',270.,12)
call NUMBER(13.6,999,.3,DBAND,270.,2)
C
CALL PLOT(0.0,0.0,999)
C
MODEL=99
IOPORT=99
C
RETURN
END
C
******************************************************************************
C ***** Cutoff-Saturation Limiting Subroutine *****
******************************************************************************
C
Subroutine LIMIT(RSAT,RCUT,INPUT,OUT)
implicit REAL*4 (A-Z)
if (INPUT .le. RSAT) then
   OUT=RSAT
elseif (INPUT .ge. RCUT) then
   OUT=RCUT
else
C
255
OUT=INPUT
end if
C
return
end
C

**************************************
C **** Function Switch Subroutine *****
C *******************************************************
Subroutine FCNSW(X1,X2,X3,X4,OUT)
implicit REAL*4 (A-Z)
if (X1 .lt. 0.0) then
  OUT=X2
elseif (X1 .eq. 0.0) then
  OUT=X3
else
  OUT=X4
end if
C
return
end
C

**************************************
C **** Step function Subroutine *****
C *******************************************************
Subroutine STEP(TIME,TSTEP,OUT)
implicit REAL*4 (A-Z)
if (TIME .ge. TSTEP) then
  OUT=1.0
else
  OUT=0.0
end if
C
return
end
C

**************************************
C **** Deadspace Subroutine *****
C *******************************************************
Subroutine DEADSP(P1,P2,VSGDEL,VSGERR)
implicit REAL*4 (A-Z)
if (VSGDEL .gt. P2) then
  VSGERR=VSGDEL-P2
elseif (VSGDEL .lt. P1) then
  VSGERR=VSGDEL-P1
else
  VSGERR=0.0
end if
C
return
end
C

**************************************
C **** Ramp Subroutine *****
C *******************************************************
Subroutine RAMP(TIME, TRAMP, OUT)
    implicit REAL*4 (A-Z)
    if (TIME .ge. TRAMP) then
        OUT = TIME - TRAMP
    else
        OUT = 0.0
    end if
    return
end

Subroutine TCONST(IC, X, TAU, NTIME, NTIM, DELTIM, Y)
    implicit REAL*4 (A-Z)
    integer*2 NTIME, NTIM
    if (NTIME .ne. NTIM) IC = Y
    DECAY = exp(-DELTIM / TAU)
    Y = X - (X - IC) * DECAY
    if (NTIME .eq. 1) Y = IC
    NTIM = NTIME
    return
end

Subroutine DERIV(DELTIM, NTIME, NTIM1, IC2, XM1, NOWVAL, XX, XDM1, +
                 XD, XDDM1, XDD, SLOPE)
    implicit REAL*4 (A-Z)
    integer*2 NTIME, NTIM1
    if (NTIME .eq. NTIM1) then
        XX = NOWVAL
    else
        XM1 = XX
        XX = NOWVAL
        XDM1 = XD
        XDDM1 = XDD
    end if
    XD = (XX - XM1) / DELTIM
    if (abs(XD) .lt. 1.E-8) XD = 0.
    if (NTIME .eq. 1) XD = IC2
    XDD = (XD - XDM1) / DELTIM
    if (abs(XDD) .lt. 1.E-8) XDD = 0.0
    NTIM1 = NTIME
    XPRED = XX + XD * DELTIM + XDD * (DELTIM**2) / 2.0
if (abs(XFRED) .lt. 1.E-8) XFRED=0.0
SLOPE=(XFRED-XM1)/(2.0*DELTIM)
if (abs(SLOPE) .lt. 1.E-8) SLOPE=0.0

return
end

-------------------------------------------------------------------------------------
***** Trapezoidal Integration Subroutine *****
-------------------------------------------------------------------------------------
Subroutine INTGRL(NTIME,NTIM2,DELTIM,IC3,FREVAL,NOWVAL,+
CURVAL,OUTOLD,OUTNEW)
implicit REAL*4 (A-Z)
integer*2 NTIME,NTIM2

if (NTIME .eq. NTIM2) then
  CURVAL=NOWVAL
else
  PREVAL = CURVAL
  CURVAL = NOWVAL
  OUTOLD = OUTNEW
end if
if (NTIME .eq. 1) OUTOLD=IC3
OUTNEW = OUTOLD+(CURVAL+PREVAL)*DELTIM/2.
NTIM2=NTIME

return
end

-------------------------------------------------------------------------------------
***** CLEAR SCREEN AND HOME CURSOR *****
-------------------------------------------------------------------------------------
subroutine CLRSCR
character*1 C1,C2,C3,C4
integer*2 IC(4)
equivalence (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
data IC/16#1B,16#5B,16#32,16#4A/

C *** Write Escape Code to Display ***
write(*,1) C1,C2,C3,C4
1 format(1X,4A1)

return
end

-------------------------------------------------------------------------------------
***** Position Cursor by Row,Column *****
-------------------------------------------------------------------------------------
subroutine GOTOXY(ROW,COLUMN)
integer*2 IC(4),ROW,COLUMN,L
character*1 C1,C2,C3,C8,LC(5)
character*5 CBUFF
equivalence (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),+
(CBUFF,LC(1))
```c

data IC/16#1B,16#5B,16#3B,16#66/

L=10000+100*ROW+COLUMN

*** Write Escape Codes to a Character Buffer ***
write(CBUFF,2) L
2 format(I5)

*** Write Escape Codes to Display ***
write(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
3 format(IX,8A1,\)
return
end
```
APPENDIX C

LUMPED PARAMETER MODEL SIMULATION PROGRAM

This program simulates the averaged parameter model configured for position control. Options are available to the user for Pulse Width Modulation, Relay, or Saturating Amplifier control. Step, Ramp, & Sinusoidal Responses are available. Both second and third order models may be analyzed.

IMPLICIT REAL*4 (A-Z)
COMMON BEGTIM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,FLEN,PLEN,NTERMS,
+ XTITLE,YTITLE,PTITLE,PTIT1
REAL*4 X(1010),Y1(1010),Y2(1010),Y3(1010),Y4(1010)
INTEGER*2 NTERMS,IOPORT,MODEL,XLEN,YLEN,NTIME,NUMIT,NTIM1,
+ NTIM2,NTIM3,NTIM4,DIR,PMODEL,NPTS,NCTR,PPLANE,
+ SIM2PL,ECTR,EDCTR,NUMBER,NDIM,CTR,ELEMENT,SYORD,TEMORD
LOGICAL*2 WAITNG,TOGGLE
CHARACTER*1 DISOPT,FLOPT
CHARACTER*6 ANS1,ANS11,ANS21,ANS27,PRTSEL,NOWLIN
CHARACTER*15 TYPE,TYPE1
CHARACTER*25 XTITLE,YTITLE,NLCHAR
CHARACTER*51 PRTCHR,PTITLE,PTIT1

... Introductory Page (1 time good deal!)
call CLRSCR
write(*,4)
PAUSE

*** OPEN/READ/CLOSE INPUT DATA FILE ***

open(7, FILE='LM.INP', STATUS='OLD', ACCESS='SEQUENTIAL')
read(7,1000) PMODEL,PRTCHR
read(7,1020) BEGTIM,FINTIM,MAXITS,SIM2PL
read(7,1022) KPWM,KV,E0,EDOT0
read(7,1024) R,L,J,F,KT,KP
read(7,1026) PERIOD,DBAND,XORG,YORG
read(7,1028) KA,SYORD
read(7,1030) TYPE,STPMAG,RSLOPE,SINAMP
read(7,1032) SINFRQ,SINPHA
close(7,STATUS='KEEP')

C

************************************************
C *** DISPLAY MAIN MENU SELECTIONS ***
C ************************************************

1 call CLRSCR
   call GOTOXY(8,1)
   write(*,5)
   read(*,'(A)') ANSI

C ... Hardware Options ...
   if ((ANSI .eq. 'h') .or. (ANSI .eq. 'H')) then

C 101 call CLRSCR
       call GOTOXY(17,24)
       write(*,'*** CURRENT PRINTER SELECTION ***')
       call GOTOXY(20,20)
       write(*,'*') FRTCHR
       call GOTOXY(10,11)
       write(*,105)
       read(*,'(A)') ANSI

C ... Printer Options ...
   if ((ANSI .eq. 'p') .or. (ANSI .eq. 'P')) then

C 131 call CLRSCR
       write(*,130)
       read(*,'(A)') FRTSEL

C
   if (FRTSEL .eq. '0') then
      FRTCHR='Epson FX-80 Printer, single density'
      PMODEL=0
   elseif (FRTSEL .eq. '1') then
      FRTCHR='Epson FX-80 Printer, double density'
      PMODEL=1
   elseif (FRTSEL .eq. '2') then
      FRTCHR='Epson FX-80 Printer, dbl spd,dual density'
      PMODEL=2
   elseif (FRTSEL .eq. '3') then
      FRTCHR='Epson FX-80 Printer, quad density'
      PMODEL=3
   elseif (FRTSEL .eq. '4') then
      FRTCHR='Epson FX-80 Printer, CRT Graphics I'
      PMODEL=4
   elseif (FRTSEL .eq. '5') then
      FRTCHR='Epson FX-80 Printer, plotter graphics'
      PMODEL=5
   elseif (FRTSEL .eq. '6') then
      FRTCHR='Epson FX-80 Printer, CRT Graphics II'
      PMODEL=6
   elseif (FRTSEL .eq. '10') then
PRTCHR='Epson FX-100 Printer, single density'
PMODEL=7
elseif (PRTSEL .eq. '11') then
  PRTCHR='Epson FX-100 Printer, double density'
  PMODEL=11
elseif (PRTSEL .eq. '12') then
  PRTCHR='Epson FX-100 Printer, dbl spd, dual density'
  PMODEL=12
elseif (PRTSEL .eq. '13') then
  PRTCHR='Epson FX-100 Printer, quad density'
  PMODEL=13
elseif (PRTSEL .eq. '14') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics I'
  PMODEL=14
elseif (PRTSEL .eq. '15') then
  PRTCHR='Epson FX-100 Printer, plotter graphics'
  PMODEL=15
elseif (PRTSEL .eq. '16') then
  PRTCHR='Epson FX-100 Printer, CRT Graphics II'
  PMODEL=16
elseif (PRTSEL .eq. '20') then
  PRTCHR='HP 7470A Graphics Plotter'
  PMODEL=20
elseif (PRTSEL .eq. '30') then
  PRTCHR='HP 7475A Graphics Plotter'
  PMODEL=30
elseif (PRTSEL .eq. '60') then
  PRTCHR='HP 2686A Laser Jet Printer'
  PMODEL=60

C ... Quit the Printer Menu ...
  elseif (((PRTSEL .eq. 'Q') .or. (PRTSEL .eq. 'q')) then
    go to 101
  else
    go to 131
  endif
  go to 101

C ... Quit the Hardware Menu ...
  elseif((ANS11 .eq. 'q') .or. (ANS11 .eq. 'Q')) then
    go to 1
  else
    go to 101
  endif

C ... Motor Parameters ...
  elseif ((ANS1 .eq. 'm') .or. (ANS1 .eq. 'M')) then

201     call CLRSCR
     write(*,230) KT,KB,R,L,J,F,KP,KV,SYSORD
     read(*,'(A)') ANS21
if (ANS21 .eq. 'KT') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for KT-->
read(*,*) KT
go to 201
elseif (ANS21 .eq. 'KB') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for KB-->
read(*,*) KB
go to 201
elseif (ANS21 .eq. 'R') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for R-->
read(*,*) R
go to 201
elseif (ANS21 .eq. 'L') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for L-->
read(*,*) L
go to 201
elseif (ANS21 .eq. 'J') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for J-->
read(*,*) J
go to 201
elseif (ANS21 .eq. 'F') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for F-->
read(*,*) F
go to 201
elseif (ANS21 .eq. 'KP') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for KP-->
read(*,*) KP
go to 201
elseif (ANS21 .eq. 'KV') then
call GOTOXY(24,1)
write(*,'(A)')'Enter a REAL value for KV-->
read(*,*) KV
go to 201
elseif (ANS21 .eq. 'SYSORD') then
call GOTOXY(24,1)
write(*,'(A)')'Enter an INTEGER value (2 or 3) for SYSORD-->
read(*,*) TEMORD
if ((TEMORD .ne. 2) .and. (TEMORD .ne. 3)) then
call CLRSCR
call GOTOXY(10,10)
write(*,*) 'The System Order may be 2 or 3 only!!'
call GOTOXY(22,1)
PAUSE
else
SYSORD=TEMORD
endif

C ... Quit Motor Parameters Menu ...
elseif (ANS21 .eq. 'Q') then
  go to 1
else
  go to 201
endif

C ... NON-LINEAR ELEMENT SELECTION MENU ...

C elseif ((ANS1 .eq. 'n') .or. (ANS1 .eq. 'N')) then
14  call CLRSCR
    call GOTOXY(21,1)
    write(*,272) NLCHAR
    call GOTOXY(1,1)
    write(*,270)
    read(*,'(A)') ANS27

C ... RELAY AS NON-LINEAR ELEMENT ...
if ((ANS27 .eq. 'r') .or. (ANS27 .eq. 'R')) then
  ELEMNT=1
  NLCHAR='RELAY'
291  call CLRSCR
    NONLIN='R'
    write(*,300) DBAND
    read(*,'(A)') ANS21
C
if (ANS21 .eq. 'DBAND') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for DBAND-->
  read(*,*) DBAND
  go to 291
elseif ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then
  go to 14
else
  go to 291
endif

C ... PULSE WIDTH MODULATOR AS NON-LINEAR ELEMENT ...
elseif ((ANS27 .eq. 'p') .or. (ANS27 .eq. 'P')) then
  ELEMNT=3
  NLCHAR='PULSE WIDTH MODULATOR'
301  call CLRSCR
    NONLIN='P'
    write(*,300) DBAND, PERIOD, KPWM
    read(*,'(A)') ANS21
C
if (ANS21 .eq. 'DBAND') then
  call GOTOXY(24,1)
  write(*,'(A)')'Enter a REAL value for DBAND-->

read(*,*) DBAND
    go to 301
else if (ANS21.eq. 'PERIOD') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for PERIOD-->
    read(*,*) PERIOD
    go to 301
elseif (ANS21.eq. 'KPWM') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for KPWM-->
    read(*,*) KPWM
    go to 301
elseif ((ANS21.eq. 'q').or. (ANS21.eq. 'Q')) then
    go to 14
else
    go to 301
endif

C
C  ... SATURATING AMPLIFIER AS NON-LINEAR ELEMENT ...
C
  elseif ((ANS27.eq. 'a').or. (ANS27.eq. 'A')) then
    ELEMNT=2
    NLCHAR='SATURATING AMPLIFIER'
    call CLRSCR
    NONLIN='A'
    write(*,280) DBAND,KA
    read(*,'(A)') ANS21

if (ANS21.eq. 'DBAND') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for DBAND-->
    read(*,*) DBAND
    go to 281
elseif (ANS21.eq. 'KA') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a REAL value for KA-->
    read(*,*) KA
    go to 281
elseif ((ANS21.eq. 'q').or. (ANS21.eq. 'Q')) then
    go to 14
else
    go to 281
endif
C
  elseif ((ANS27.eq. 'q').or. (ANS27.eq. 'Q')) then
    go to 1
else
    go to 14
endif
C
C  ... Simulation Options ...
C
  elseif ((ANS1.eq. 'o').or. (ANS1.eq. 'O')) then

265
DEL TIM = (FINTIM - BEG TIM) / (1000 * SIM2PL)
if (FINTIM / DEL TIM .gt. MAXITS) DEL TIM = FINTIM / MAXITS

NTERMS = IFIX (FINTIM / DEL TIM) + 1

write(*, 240) BEG TIM, FINTIM, MAXITS, SIM2PL, E0, EDOT0, XORG, YORG, DEL TIM, NTERMS
read(*, ' (A)') ANS21

if (ANS21 .eq. 'BEGIN T IM') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for BEGIN TIM — 
read(*, *) BEG TIM
go to 202
elseif (ANS21 .eq. 'FINTIM') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for FINT IM — 
read(*, *) FINT IM
go to 202
elseif (ANS21 .eq. 'MAXITS') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for MAXITS — 
read(*, *) MAXITS
go to 202
elseif (ANS21 .eq. 'SIM2PL') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a INTEGER value for SIM2PL — 
read(*, *) SIM2PL
go to 202
elseif (ANS21 .eq. 'E0 ') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for E0 (degrees) — 
read(*, *) E0
go to 202
elseif (ANS21 .eq. 'EDOT0') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for EDOT0 (deg/sec) — 
read(*, *) EDOT0
go to 202
elseif (ANS21 .eq. 'XORG') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for XORG — 
read(*, *) XORG
go to 202
elseif (ANS21 .eq. 'YORG') then
call GOTOXY (24, 1)
write(*, '(A)') 'Enter a REAL value for YORG — 
read(*, *) YORG
go to 202
C ... Quit Simulation Options Menu ...
  elseif (ANS21 .eq. 'Q') then
    go to 1
  else
    go to 202
  end if
C ... Command Input Selection ...
  elseif ((ANS1 .eq. 'c') .or. (ANS1 .eq. 'C')) then
C
203    call CLRSCR
C
  write(*,245) TYPE,STPMAG,RSLOPE,SINAMP,SINFRQ,SINPHA
  read(*, '(A)') ANS21
C
  if (ANS21 .eq. 'TYPE') then
    call GOTOXY(24,1)
    write(*,'(A)') 'Enter a STEP, RAMP, or SINE-->
    read(*, '(A)') TYPE1
    if ((TYPE1 .ne. 'STEP') .and. (TYPE1 .ne. 'RAMP') .and.
        (TYPE1 .ne. 'SINE')) then
      call CLRSCR
      call GOTOXY(10,10)
      write(*,*) 'Invalid Selection !!!'
      call GOTOXY(20,1)
      PAUSE
    else
      TYPE=TYPE1
    endif
  go to 203
elseif (ANS21 .eq. 'STPMAG') then
  call GOTOXY(24,1)
  write(*,'(A)') 'Enter a REAL value for STPMAG-->
  read(*,*) STPMAG
  go to 203
elseif (ANS21 .eq. 'RSLOPE') then
  call GOTOXY(24,1)
  write(*,'(A)') 'Enter a REAL value for RSLOPE-->
  read(*,*) RSLOPE
  go to 203
elseif (ANS21 .eq. 'SINAMP') then
  call GOTOXY(24,1)
  write(*,'(A)') 'Enter a REAL value for SINAMP-->
  read(*,*) SINAMP
  go to 203
elseif (ANS21 .eq. 'SINFRQ') then
  call GOTOXY(24,1)
  write(*,'(A)') 'Enter a REAL value for SINFRQ-->
  read(*,*) SINFRQ
go to 203
elseif (ANS21 .eq. 'SINPHA') then
  call GOTOXY(24,1)
  write(*,'(A)') 'Enter a REAL value for SINPHA-->
  read(*,*) SINPHA
  go to 203
C ... Quit Simulation Options Menu ...
elseif (ANS21 .eq. 'Q') then
  go to 1
else
  go to 203
end if
C ... Save Options to File ...
elseif ((ANS1 .eq. 's') .or. (ANS1 .eq. 'S')) then
  C
  ***********************************************************************
  C *** OPEN/READ/INPUT DATA FILE ***
  ***********************************************************************
  C
  open(7,FILE='LM.INP',STATUS='NEW')
  write(7,1000) PMODEL,PRTCHR
  write(7,1020) BEGTIM,FINTIM,MAXITS,SIM2PL
  write(7,1022) KPW,KV,E0,EDOT0
  write(7,1024) R,L,J,F,KT,KB,KP
  write(7,1026) PERIOD,DBAND,XORG,YORG
  write(7,1028) KA(SYSORD
  write(7,1030) TYPE,STPMAG,RSLOPE,SINAMP
  write(7,1032) SINFRQ,SINPHA
  write(7,2000)
  close(7,STATUS='KEEP')
  C
  go to 1
  C ... Run the Program ...
elseif ((ANS1 .eq. 'r') .or. (ANS1 .eq. 'R')) then
  go to 2
  C ... Quit the Program ...
elseif ((ANS1 .eq. 'q') .or. (ANS1 .eq. 'Q')) then
  stop
else
  go to 1
endif
C ... Open an Output Data File ...
  2 open(4,FILE='LM.OUT',STATUS='NEW')
  C
  PI=3.14159
  N=1
C ... Initializations ...
NCTR=0
NPTS=0
NTIM2=1
NTIM3=0
NTIM4=0
PARG1=0.0
PARG2=0.0
FVAL4=0.
CARG1=0.0
CARG2=0.0
CVAL4=0.
TOT1=0.0
TOT2=0.0
TVAL1=0.0
TVAL2=0.0
TVAL4=0.
XM1=0.0
XX=0.0
XDM1=0.0
XD=0.0
XDDM1=0.0
XDD=0.0
X1=0.0
F1=0.0
F2=0.0
X3=0.0
WM1=0.0
WM=0.0
IM=0.
IMO=0.
TSTART=0.0
THEDEG=0.0
TOGGLE=.true.
PPLANE=0
NUMBER=0
NDIM=20
VIN=0.

C

C ... Preliminary Relationships ...

C ... Initial conditions for THETA(fin, degrees) and Omega(fin, degrees).
E0RAD=10.*E0*PI/180.
EDOT0R=10.*EDOT0*PI/180.
if (KP .ne. 0.) then
WM0=-EDOT0R/KP
THETA0=-E0RAD/KP
elseif (KP .eq. 0.) then
WM0=-EDOT0
THETA0=-E0RAD
endif
IMO=(VIN-WM0*KB)/R
X2=WM0

269
PTHETA=THETA0
TAU1=L/R
TAU2=J/F

C call CLRSCR
C ... Output Simulation Options to Output Data File ...
write(4,1200) SYSORD,BEGTIM,FINTIM,MAXITS,SIM2PL,KT,KB,R,L,J,F,KP,
+ KV,KA,PERIOD,DBAND,EO,EDO20
write(4,1201) TYPE,STPMAG,RSLOPE,SINAMP,SINFRQ,SINPHA
C ...
C ... Display the simulation header ...
call CLRSCR
call GOTOXY(10,27)
write(*,*),'Simulation in Progress'
DELTIM=(FINTIM-BEGTIM)/(1000.*SIM2PL)
if (FINTIM/DELTIM.gt.MAXITS) DELTIM=FINTIM/MAXITS
NTERMS=IFIX(FINTIM/DELTIM)+1
call GOTOXY(14,1)
write(*,15) DELTIM,NTERMS
C
format(16X,'Simulation Step Size --> ',F9.8,' seconds',/,
+ 17X,'Total Number of Steps --> ',15)
call GOTOXY(21,1)
C
if ((NONLIN.eq.'r') .or. (NONLIN .eq. 'R')) then
write(*,*)'*** NON-LINEAR ELEMENT IS RELAY ***'
elseif ((NONLIN.eq.'a') .or. (NONLIN .eq. 'A')) then
write(*,*)'*** NON-LINEAR ELEMENT IS SATURATING AMPLIFIER ***'
elseif ((NONLIN.eq.'p') .or. (NONLIN .eq. 'P')) then
write(*,*)'*** NON-LINEAR ELEMENT IS PWM ***'
else
write(*,*)'*** NON-LINEAR ELEMENT NOT SELECTED ***'
write(*,*)'... Return to Main Menu ...
PAUSE
go to 1
endif
C
SPACE=DELTIM/2.0
C
*******************************************************************************
C *** START MAIN SIMULATION LOOP ***
*******************************************************************************
C
DO 100 NTIME=1,NTERMS
TIME=(NTIME-1)*DELTIM
NUMIT=0
THETA=PTHETA
C
*******************************************************************************
C *** START INNER SIMULATION LOOP ***
*******************************************************************************
200  NU=NU+1
  M=M*X2

... Command Input Signal Generator ...
  if (TYPE .eq. 'STEP') then
    ORDER=STPMAG
  elseif (TYPE .eq. 'RAMP') then
    ORDER=RSLOPE*TIME
  elseif (TYPE .eq. 'SINE') then
    ORDER=SINAMP*sin(SINFRQ*TIME+SINPHA)
  endif

THETA=.1*THETA*180./PI
OMEGA=.1*M*180./PI
POSSERR=ORDER-KP*THETA-KV*OMEGA

... Utilization of Non-Linear Element ...
... PULSE WIDTH MODULATOR ...
  if (((NONLIN .eq. 'P') .or. (NONLIN .eq. 'P')) then
    call PWMOD(TIME,NU,TSTART,PERIOD,TOGGLE,POSSERR,
    + DBAND,VIN,VREF,THRESH,KPWM)
  elseif (((NONLIN .eq. 'R') .or. (NONLIN .eq. 'R')) then
    call RELAY(POSSERR,DBAND,VIN)
  elseif (((NONLIN .eq. 'A') .or. (NONLIN .eq. 'A')) then
    call AMPLIF(POSSERR,DBAND,KA,VIN)
  else
    call GOTOXY(21,1)
    write(*,*) '*** NON-LINEAR ELEMENT NOT SELECTED ***'
    write(*,*) '... Return to Main Menu ...'
    PAUSE
    go to 1
  endif

EN=VIN-WM*KB

if (SYSORD .eq. 3) then
  call TCONST(IM0,EN/R,TAU1,NTIM1,DELTIM,IM)
endif
else (SYSORD .eq. 2) then
  IM=EN/R
else
  call CLRSCR
  call GOTOXY(10,10)
  write(*,*) 'SYSTEM ORDER IS NOT 2 OR 3!!'
  PAUSE
  go to 1
endif

TM=IM*KT
call TCONST(WMO, TM/F, TAU2, NTIME, NTIM2, DELTIM, WM1)
call INTEGRAL(NTIME, NTIM4, DELTIM, THETA0, FVAL4, WM1, CVAL4, TVAL4, THET1)

THETA=THET1

*******************************************************************************
*** CONVERGENCE CRITERIA (Newton's Method) ***
*******************************************************************************

if (NTIME .eq. 1) then
  X3=WM1
  F2=0.
  go to 150
end if

if (NUMIT .eq. 1) then
  F2=(WM1-WM)*1.2
  X3=1.001*WM1+1.0E-4
  RELERR=abs(F2/X3)
  go to 310
end if

F2=WM1-WM
if (F1 .eq. F2) F2=.999*F2-1.E-8
if (X2 .ne. 0.) RELERR=abs(F2/X2)
if (X2 .eq. 0.) RELERR=1.
if (RELERR .gt. 1.E-8) then
  X3=X2-F2*(X1-X2)/(F1-F2)
endif

310 X1=X2
    X2=X3
    F1=F2

WM=X2

if (NUMIT .ge. 10) go to 150
if (RELERR .gt. 1.E-8) go to 200

*******************************************************************************
*** END INNER SIMULATION LOOP ***
*******************************************************************************

150 call DERIV(DELTIM, NTIME, NTIM3, 0., XM1, WM1, XX, XDM1, XD, XDDM1, XDD, +
               ALPHA)

X2=WM1+ALPHA*DELTIM
PTHETA=THETA+WM1*DELTIM
THETAF=.1*THETA*180./PI
OMEGAF=.1*WM1*180./PI

C ... Generate Plotting Arrays ...
if (TIME .ge. BEGTIM) then
NCTR=NCTR+1
if (mod(NCTR,SIM2PL).eq. 0).or. (NCTR.eq. 1)) then
NPTS=NPTS+1
X(NPTS)=TIME
Y1(NPTS)=THETAF
Y2(NPTS)=OMEGAF
Y3(NPTS)=ORDER
Y4(NPTS)=IM
endif
endif

100 CONTINUE

********************************
*** END MAIN SIMULATION LOOP ***
********************************

close(4,STATUS='KEEP')

****************************
***** Plotting selection *****
****************************

C *** Clear Screen & Home Cursor ***
400 call CLRSCR
C
write(*,1305)
read(*,'(A)') DISOPT
C
if ((DISOPT.eq.'m').or.(DISOPT.eq.'M')) then
MODEL=99
IOPORT=99
endif
if ((DISOPT.eq.'p').or.(DISOPT.eq.'P')) then
MODEL=PMODEL
IOPORT=1
C ... Ioprt=9600 is COM1 ...
C ... Ioprt=9650 is COM2 ...
if ((MODEL.eq.20).or.(MODEL.eq.30)) IOPORT=9600
endif
GO TO 13
endif
C
**********************************************************************
*** OPEN/WRITE/CLOSE INPUT DATA FILE ***
**********************************************************************

open(7,FILE='LM.INP',STATUS='NEW')
write(7,1000) PMODEL,FRTCR
write(7,1020) BEGIN,TIM,FINTIM,MAXITS,SIM2PL
write(7,1022) KFWM,KW,EO,EDOTO
write(7,1024) R,L,J,KT,KB,KP
write(7,1026) PERIOD, DBAND, XORG, YORG
write(7,1028) KA, SYSORD
write(7,1030) TYPE, STPMAG, RSLOPE, SINAMP
write(7,1032) SINFRQ, SINPHA
write(7,2000)
close(7, STATUS='KEEP')

C
go to 400
C
elseif ((DISOPT .eq. 'Q') .or. (DISOPT .eq. 'q')) then
   GO TO 460
else
   go to 400
end if
C
410 call CLRSCR
   if ((DISOPT .eq. 'p') .or. (DISOPT .eq. 'P')) then
      call GOTOXY(20,20)
      write(*,*) FRTCHR
      call GOTOXY(1,1)
   endif
   write(*,1300)
read(*,'(A)') FLOPT
C
   if ((FLOPT .eq. 'Q') .or. (FLOPT .eq. 'q')) then
      go to 400
   elseif (FLOPT .eq. '1') then
      XTITLE='TIME (seconds)'
      XLEN=15
      YTITLE='FIN POSITION (deg)'
      YLEN=19
      if (ELEMNT .eq. 1) then
         PTITLE='FIN POSITION RESPONSE WITH RELAY CONTROLLER'
         PLEN=44.
      elseif (ELEMNT .eq. 2) then
         PTITLE='FIN POSITION RESPONSE WITH SAT AMP CONTROLLER'
         PLEN=46.
      elseif (ELEMNT .eq. 3) then
         PTITLE='FIN POSITION RESPONSE WITH PAM CONTROLLER'
         PLEN=42.
      endif
   C
      if (SYSORD .eq. 2) then
         PTIT1='REDUCED ORDER MODEL'
         PLEN1=20.
      else
         PTIT1='THIRD ORDER MODEL'
         PLEN1=18.
      endif
   C
      call MGRAPH(X, Y1, Y3, XORG, YORG, DISOPT)

274
elseif (PLOPT .eq. '2') then
  XTITLE='TIME (seconds)'
  XLEN=-15
  YTITLE='FIN VELOCITY (deg/sec)'
  YLEN=23
  if (ELEMNT .eq. 1) then
    PTITLE='FIN VELOCITY RESPONSE WITH RELAY CONTROLLER'
    PLEN=44.
  elseif (ELEMNT .eq. 2) then
    PTITLE='FIN VELOCITY RESPONSE WITH SAT AMP CONTROLLER'
    PLEN=46.
  elseif (ELEMNT .eq. 3) then
    PTITLE='FIN VELOCITY RESPONSE WITH PWM CONTROLLER'
    PLEN=42.
  endif
if (SYSORD .eq. 2) then
  PTITLE='REDUCED ORDER MODEL'
  PLEN1=20.
else
  PTITLE='THIRD ORDER MODEL'
  PLEN1=18.
endif

call GRAPH(X,Y2,XORG,YORG,DISOPT)

elseif (PLOPT .eq. '3') then
  XTITLE='TIME (seconds)'
  XLEN=-15
  YTITLE='MOTOR CURRENT (amps)'
  YLEN=20
  if (ELEMNT .eq. 1) then
    PTITLE='MOTOR CURRENT RESPONSE WITH RELAY CONTROLLER'
    PLEN=45.
  elseif (ELEMNT .eq. 2) then
    PTITLE='MOTOR CURRENT RESPONSE WITH SAT AMP CONTROLLER'
    PLEN=47.
  elseif (ELEMNT .eq. 3) then
    PTITLE='MOTOR CURRENT RESPONSE WITH PWM CONTROLLER'
    PLEN=43.
  endif
if (SYSORD .eq. 2) then
  PTITLE='REDUCED ORDER MODEL'
  PLEN1=20.
else
  PTITLE='THIRD ORDER MODEL'
  PLEN1=18.
endif

275
call GRAPH(X,Y4,XORG,YORG,DISOPT)

else
    go to 410
endif

go to 400
460 continue

********************************************************************************

*** I/O FORMAT STATEMENTS ***
********************************************************************************

4 format(///,25X,'LUMPED PARAMETER DC MOTOR SIMULATION',///,
       + 25X,' LT Vincent S. Rossitto, USN',///,
       + 25X,' Naval Postgraduate School',///,
       + 25X,' March 1987',//////)

5 format(32X,'*** MAIN MENU ***',///,
       + 20X,' [H]----> HARDWARE Configuration Menu',///,
       + 20X,' [M]----> MOTOR Parameter Menu',///,
       + 20X,' [N]----> NON-LINEAR Element Selection Menu',///,
       + 20X,' [O]----> OPTIONS for Simulation',///,
       + 20X,' [C]----> COMMAND Input Selection Menu',///,
       + 20X,' [S]----> SAVE All Changes',///,
       + 20X,' [R]----> RUN Simulation Program',///,
       + 20X,' [Q]----> QUIT the Program',///,
       + 8X,' ENTER SELECTION----> ',/)

105 format(30X,'*** HARDWARE MENU ***',///,
       + 20X,' [P]----> PRINTER/PLOTTER configuration change',///,
       + 20X,' [Q]----> QUIT THIS MENU',///,
       + 8X,' ENTER SELECTION----> ',/) 1000 format(1X,I3,2X,A50)
1020 format(1X,3F15.7,1X,I3)
1022 format(1X,4F15.7)
1024 format(1X,7F8.4)
1026 format(1X,4F8.4)
1028 format(1X,F12.7,1X,I1)
1030 format(1X,A15,3F15.7)
1032 format(1X,2F15.7)
2000 format(1X,' END OF FILE')

130 FORMAT(24X,'*** PRINTER OPTIONS MENU ***',///,
       + 15X,' [0] ----> Epson FX-80 Printer, single density',///,
       + 15X,' [1] ----> Epson FX-80 Printer, double density',///,
       + 15X,' [2] ----> Epson FX-80 Printer, dble spd,dual density',///,
       + 15X,' [3] ----> Epson FX-80 Printer, quad density',///,
       + 15X,' [4] ----> Epson FX-80 Printer, CRT Graphics I',///,
       + 15X,' [5] ----> Epson FX-80 Printer, plotter graphics',///,
       + 15X,' [6] ----> Epson FX-80 Printer, CRT Graphics II',///,
Epson FX-100 Printer, single density

Epson FX-100 Printer, double density

Epson FX-100 Printer, dble spd.dual density

Epson FX-100 Printer, quad density

Epson FX-100 Printer, CRT Graphics I

Epson FX-100 Printer, plotter graphics

Epson FX-100 Printer, CRT Graphics II

HP 7470A Graphics Plotter

HP 7475A Graphics Plotter

HP 2686A Laser Jet Printer (NPS installation)

QUIT THIS MENU

Enter Printer Selection Integer or Q to QUIT --->

Motor Torque Constant

Motor Back EMF Constant

Motor Equivalent Resistance (ohms)

Motor Inductance (henries)

Motor Inertia (oz-in/s²)

Motor Viscous Friction Coeff

Motor Position Feedback Constant

Motor Velocity Feedback Constant

System Order (2=Reduced; 3=Linearized)

QUIT THIS MENU

Enter variable name (UPPERCASE) or Q to QUIT --->

Start Time of Plotting Window

Stop Time of Plotting Window

Max Number of Simulation Iterations

Ratio: Points Simulated/Plotted

Initial Fin Position (deg)

Initial Fin Velocity (deg/s)

X Coordinate of Plotting Origin

Y Coordinate of Plotting Origin

QUIT THIS MENU

Computed simulation step size --->

Computed total number of steps--->

Enter variable name (UPPERCASE) or Q to QUIT --->

STEP, RAMP, or SINE Response

Commanded Position for STEP Response

Slope of RAMP Function

Amplitude of SINE Function

Frequency (deg/sec) of SINE Function

Phase Angle (deg) of SINE Function

QUIT THIS MENU

Enter variable name (UPPERCASE) or Q to QUIT --->

*** MOTOR PARAMETER SETTINGS MENU ***

*** SIMULATION OPTIONS MENU ***

*** COMMAND INPUT SELECTION MENU ***

*** NON-LINEAR ELEMENT SELECTION ***
+ 10X, '[R] Relay (Bang-Bang)'//,
+ 10X, '[P] Pulse Width Modulator'//,
+ 10X, '[A] Amplifier (Saturating)'//,
+ 10X, '[Q] QUIT THIS MENU/RETURN TO MAIN MENU'//,
+ 1X, 'Enter Selection ---> ',

C

272 format(10X, 'CURRENT SELECTION ---> ', A30)

C

280 format(///,8X,'*** SATURATING AMPLIFIER SPECIFICATIONS ***'//,
+ 1X,F15.7,1X, '[DBAND] Deadband Applied to System Feedback'//,
+ 1X,F15.7,1X, '[KA] Amplifier Gain'//,
+ 17X, '[Q] QUIT THIS MENU'//,
+ 1X, 'Enter the selection (UPPERCASE) ---> ',

C

290 format(///,15X,'*** RELAY SPECIFICATIONS ***'//,
+ 1X,F15.7,1X, '[DBAND] Deadband Applied to System Feedback'//,
+ 1X,F15.7,1X, '[Q] QUIT THIS MENU'//,
+ 1X, 'Enter the selection (UPPERCASE) ---> ',

C

300 format(///,5X,'*** PULSE WIDTH MODULATOR SPECIFICATIONS ***'//,
+ 1X,F15.7,1X, '[DBAND] Deadband Applied to System Feedback'//,
+ 1X,F15.7,1X, '[PERIOD] Period of PWM Reference Cycle(sec)'//,
+ 1X,F15.7,1X, '[KPWM] PWM Amplifier Gain'//,
+ 17X, '[Q] QUIT THIS MENU'//,
+ 1X, 'Enter the selection (UPPERCASE) ---> ',

C

1200 format(20X, 'LINEAR MODEL OF BRUSHLESS DC MOTOR'//,
+ 15X,I1,1X, '[SYSORD] System Order (2=Reduced;3=Linearized)'//,
+ 1X,F15.7,1X, '[BEGTIM] Start Time of Plotting Window'//,
+ 1X,F15.7,1X, '[FINTIM] Stop Time of Plotting Window'//,
+ 1X,F15.7,1X, '[MAXITS] Max Number of Simulation Iterations'//,
+ 13X,I3,1X, '[SIM2PL] Ratio: Points Simulated/Plotted'//,
+ 4X,F12.7,1X, '[KT] Motor Torque Constant'//,
+ 4X,F12.7,1X, '[KB] Motor Back EMF Constant'//,
+ 4X,F12.7,1X, '[R] Motor Equivalent Resistance (ohms)'//,
+ 4X,F12.7,1X, '[L] Motor Inductance (henries)'//,
+ 4X,F12.7,1X, '[F] Motor Viscous Friction Coeff'//,
+ 4X,F12.7,1X, '[KV] Motor Velocity Feedback Constant'//,
+ 4X,F12.7,1X, '[KA] Saturating Amplifier Gain (if used)'//,
+ 4X,F12.7,1X, '[PERIOD] Period of PWM Reference Cycle'//,
+ 4X,F12.7,1X, '[DBAND] Position Feedback Deadband'//,
+ 1X,F15.7,1X, '[E0] Initial Fin Position (deg)'//,
+ 1X,F15.7,1X, '[EDOTO] Initial Fin Velocity (deg/s)'

C

1201 format(1X,A15,1X, '[TYPE] STEP, RAMP, or SINE Response'//,
+ 1X,F15.7,1X, '[STPMAG] Commanded Position for STEP Response'//,
+ 1X,F15.7,1X, '[RSLOPE] Slope of RAMP Function'//,
+ 1X,F15.7,1X, '[SINAMP] Amplitude of SINE Function'//,
+ 1X,F15.7,1X, '[SINFRQ] Frequency (deg/sec) of SINE Function'//,
+ 1X,F15.7,1X,'[SINPHA] Phase Angle (deg) of SINE Function'

C
1205 FORMAT(1X,F8.6,1X,1P5E12.3)
1300 FORMAT(\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n
C
STOP
END
C

*******************************
C ***** PLOTTING SUBROUTINES *****
C *****(single function plot)*****
C *******************************

Subroutine GRAPH(X,Y,XORG,YORG,DISOPT)

C implicit REAL*4 (A-Z)
COMMON BEGTIM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,PLEN,PLEN1,
+ XTITLE,YTITLE,PTITLE,PTIT1
real*4 X(1010),Y(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,NCHAR,NCHAR1
character*1 DISOPT,ANS
character*25 XTITLE,YTITLE
character*51 PTITLE,PTIT1
C
C ... Make a new title...
5 call CLRSCR
   if ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
      call GOTOXY(8,30)
      write(*,*) 'Current Title is:
      call GOTOXY(10,20)
      write(*,*) PTITLE
      call GOTOXY(11,20)
      write(*,*) PTIT1
      call GOTOXY(15,25)
      write(*,'(A)') 'Do you want to change the title? '
      read(*,'(A)') ANS
      call CLRSCR
C
279
if ((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
  call GOTOXY(5,10)
write(*,*)'Enter titles in left justified format'
call GOTOXY(12,10)
write(*,*)'1234567890123456789012345678901234567890'
call GOTOXY(13,10)
write(*,*)
  1  2  3  4  5'
call GOTOXY(15,25)
write(*,*)'# of characters -->'
call GOTOXY(20,10)
write(*,*)'1234567890123456789012345678901234567890'
call GOTOXY(21,10)
write(*,*)
  1  2  3  4  5'
call GOTOXY(23,25)
write(*,*)'# of characters -->'
call GOTOXY(11,11)
read(*,'(A51)') PTITLE
call GOTOXY(15,46)
read(*,*) FLEN
call GOTOXY(19,11)
read(*,'(A51)') PTIT1
call GOTOXY(23,46)
read(*,*) FLEN1
elseif ((ANS .eq. 'N') .or. (ANS .eq. 'n')) then
go to 10
endif
  go to 5
endif
C
10 call GOTOXY(10,25)
write(*,*),'Calculating Plotting Data'
C
ASFRAT=.65
CHARHT=.22
CHARH1=.20
PTX=.5+(6.-FLEN*ASFRAT*CHARHT)/2.
PTY=4.5
PTX1=.5+(6.-FLEN1*ASFRAT*CHARH1)/2.
PTY1=4.1
NCHAR=fix(FLEN)
NCHAR1=fix(FLEN1)
C
call PLOTS(0,IOPORT,MODEL)
call FACTOR(1.00)
call ASPECT(ASFRAT)
C ... Draw a Border
if ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
call PLOT(XORG,YORG,-13)
call PLOT(8.0,0.0,2)
call PLOT(8.0,6.0,2)
call PLOT(0.0,6.0,2)

280
call PLOT(0.0,0.0,0.2)
call PLOT(1.25,1.0,-13)

elseif ((DISOPT .eq. 'M') .or. (DISOPT .eq. 'm')) then
call PLOT(1.1,1.0,-13)
endif

... This scaling applies when the X axis represents Time...
X(NPTS+1)=BEGTIM
FIRSTX = X(NPTS+1)
X(NPTS+2)=((X(NPTS)-X(NPTS+1))/6.
DELTAX = X(NPTS+2)

call SCALE(Y,4.,NPTS,1)
FIRSTY = Y(NPTS+1)
DELTAY = Y(NPTS+2)
call STAXIS(.15,.22,.12,.080,3)
call AXIS(0.0,0.0,XTITLE,XLEN,6.0,FIRSTX,DELTAX)
call STAXIS(.15,.22,.12,.080,2)
call AXIS(0.0,0.0,YTITLE,YLEN,90,FIRSTY,DELTAY)
call SYMBOL(PTX,PTY,CHARHT,PTITLE,0.,NCHAR)
call SYMBOL(PTX1,PTY1,CHARH1,PTIT1,0.,NCHAR1)
call LINE(X,Y,NPTS,1,0,0)
call PLOT(0.,0.,999)

MODEL=99
IOPORT=99

return
end

********************************************
***** PLOTTING SUBROUTINES *****
***** (multi-function plot)*****
********************************************

Subroutine MGRAPH(X,Y,Z,XORG,YORG,DISOPT)

implicit REAL*4 (A-Z)
COMMON BEGTIM,FINTIM,NPTS,IOPORT,MODEL,XLEN,YLEN,PLEN,NCHAR
+ XTITLE,YTITLE,PTITLE,PTIT1
real*4 X(1010),Y(1010),Z(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,NCHAR,NCHAR1
character*25 XTITLE,YTITLE
character*51 PTITLE,PTIT1

... Make a new title...
5 call CLRSCR
if ( (DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
call GOTOXY(8,30)
write(*,*)'Current Title is:'
call GOTOXY(10,20)
write(*,*), PTITLE
call GOTOXY(11,20)
write(*,*), PTIT1
call GOTOXY(15,25)
write(*,'(A)'), 'Do you want to change the title?'
read(*,'(A)'), ANS

if ((ANS .eq. 'Y') .or. (ANS .eq. 'y')) then
    call GOTOXY(5,10)
    write(*,*), 'Enter titles in left justified format'
    call GOTOXY(12,10)
write(*,*), '12345678901234567890123456789012345678901234567890'
call GOTOXY(13,10)
write(*,*), '
    call GOTOXY(15,25)
write(*,*), '# of characters -->'
call GOTOXY(20,10)
write(*,*), '12345678901234567890123456789012345678901234567890'
call GOTOXY(21,10)
write(*,*), '
    call GOTOXY(23,25)
write(*,*), '# of characters -->'
call GOTOXY(11,11)

read(*,'(A51)'), PTITLE
call GOTOXY(15,46)
read(*,*), PLEN
call GOTOXY(19,11)
read(*,'(A51)'), PTIT1
call GOTOXY(23,46)
read(*,*), PLEN1

elseif ((ANS .eq. 'N') .or. (ANS .eq. 'n')) then
    go to 10
endif

go to 5
endif

call GOTOXY(10,25)
write(*,*), 'Calculating Plotting Data'

ASPRAT=.65
CHARHT=.22
CHARH1=.20
PTX=0.5+(6.-PLEN*ASPRAT*CHARHT)/2.
PTY=4.5
PTX1=0.5+(6.-PLEN1*ASPRAT*CHARH1)/2.
PTY1=4.1
NCHAR=fix(PLEN)
NCHAR1=fix(PLEN1)
call PLOTS(0,IOPORT,MODEL)
call FACTOR(1.00)
call ASPECT(ASPRAT)
C ... Draw a Border
if ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
    call PLOT(XORG,YORG,-13)
    call PLOT(8.0,0.0,2)
    call PLOT(8.0,6.0,2)
call PLOT(0.0,6.0,2)
call PLOT(0.0,0.0,2)
call PLOT(1.25,1.,-13)
elseif ((DISOPT .eq. 'M') .or. (DISOPT .eq. 'm')) then
     call PLOT(1.,1.,-13)
endif
C
C
C ... This scaling applies when the X axis represents Time...
X(NPTS+1)=BEGTIM
FIRSTX = X(NPTS+1)
X(NPTS+2)=(X(NPTS)-X(NPTS+1))/6.
DELTAX = X(NPTS+2)
C
call SCALE(Y,4.,NPTS,1)
call SCALE(Z,4.,NPTS,1)
if (Z(NPTS+2) .gt. Y(NPTS+2)) then
    Y(NPTS+2)=Z(NPTS+2)
else
    Z(NPTS+2)=Y(NPTS+2)
endif
FIRSTY = Y(NPTS+1)
DELTAY = Y(NPTS+2)
C
call STAXIS(.15,.22,.12,.080,3)
call AXIS(0.0,0.0,XTITLE,XLEN,6.,0.,FIRSTX,DELTAX)
call STAXIS(.15,.22,.12,.080,2)
call AXIS(0.0.,YTITLE,YLEN,4.,90.,FIRSTY,DELTAY)
call SYMBOL(PTX,PTY,CHARHT,PTITLE,0.,NCHAR)
call SYMBOL(PTX1,PTY1,CHARH1,PTIT1,0.,NCHAR1)
call LINE(X,Y,NPTS,1,0,0)
call CURVE(X,Z,NPTS,-.1)
call PLOT(0.,0.,999)
C
MODEL=99
IOPORT=99
C
return
end
C
******************************************************************************
C ***** PULSE WIDTH MODULATOR MODULE *****
******************************************************************************
C
Subroutine PWMOD(TIME,NUMIT,TSTART,PERIOD,TOGGLE,POSERR,
... Reset the saw-tooth reference signal ...

if (TIME .ge. TSTART+PERIOD) then
   TSTART=TSTART+PERIOD
   TOGGLE=.true.
endif

if (POSERR .gt. (O.+DBAND)) then
   DIR=1
   ERROR=abs(KPWM*(POSERR-DBAND))
elseif (POSERR .lt. (O.-DBAND)) then
   DIR=-1
   ERROR=abs(KPWM*(POSERR+DBAND))
else
   ERROR=0.
endif

call LIMIT(0.,1.,ERROR,ERROR)
call RAMP(TIME,TSTART,NREF)
VREF=NREF/PERIOD
THRESH=1.-ERROR

... "WAITING" is a logical variable indicating whether or not a new pulse may be generated ...
if (NUMIT .eq. 1) WAITNG=TOGGLE

if (WAITING) then
   if (VREF .gt. THRESH) then
      if (DIR .eq. 1) then
         VIN=150.
         TOGGLE=.false.
      elseif (DIR .eq. -1) then
         VIN=-150.
         TOGGLE=.false.
      endif
   elseif (VREF .lt. THRESH) then
      VIN=0.
      TOGGLE=.true.
   endif
endif

return
end

************************
***** RELAY MODULE *****
************************
Subroutine RELAY(POSERR, DBAND, VIN)

implicit REAL*4 (A-Z)

if (POSERR .gt. DBAND) then
  VIN=150.
else if (POSERR .lt. -DBAND) then
  VIN=-150.
else if (abs(POSERR) .le. DBAND) then
  VIN=0.
endif

return
end

***************************************
***** SATURATING AMPLIFIER MODULE *****
***************************************

Subroutine AMPLIF(POSERR, DBAND, KA, VIN)

implicit REAL*4 (A-Z)

if (abs(POSERR) .le. DBAND) then
  VIN=0.
else if (((POSERR-DBAND)*KA .gt. 0.) .and. 
  ((POSERR-DBAND)*KA .lt. 1.)) then
  VIN=150.*((POSERR-DBAND)*KA
else if (((POSERR+DBAND)*KA .lt. 0.) .and. 
  ((POSERR+DBAND)*KA .gt. -1.)) then
  VIN=150.*((POSERR+DBAND)*KA
else if (((POSERR-DBAND)*KA .ge. 1.) then
  VIN=150.
else if (((POSERR+DBAND)*KA .le. -1.) then
  VIN=-150.
endif

return
end

***************************************
***** Cutoff-Saturation Limiting Subroutine *****
***************************************

Subroutine LIMIT(RSAT, RCUT, INPUT, OUT)

implicit REAL*4 (A-Z)

if (INPUT .le. RSAT) then
  OUT=RSAT
else if (INPUT .ge. RCUT) then
  OUT=RCUT
else

285
OUT=INPUT
end if
C
return
end
C
**************************************
C
***** Function Switch Subroutine *****
C
******************************************************************************
Subroutine FCNSW(X1,X2,X3,X4,OUT)
implicit REAL*4 (A-Z)
if (X1 .lt. 0.0) then
OUT=X2
elseif (X1 .eq. 0.0) then
OUT=X3
else
OUT=X4
end if
C
return
end
C
******************************************************************************
C
***** Step function Subroutine *****
C
******************************************************************************
Subroutine STEP(TIME,TSTEP,OUT)
implicit REAL*4 (A-Z)
if (TIME .ge. TSTEP) then
OUT=1.0
else
OUT=0.0
end if
C
return
end
C
******************************************************************************
C
***** Deadspace Subroutine *****
C
******************************************************************************
Subroutine DEADSP(P1,P2,VSGDEL,VSGERR)
implicit REAL*4 (A-Z)
if (VSGDEL .gt. P2) then
VSGERR=VSGDEL-P2
elseif (VSGDEL .lt. P1) then
VSGERR=VSGDEL-P1
else
VSGERR=0.0
end if
C
return
end
C
******************************************************************************
C
***** Ramp Subroutine *****
C
******************************************************************************
Subroutine RAMP(TIME, TRAMP, OUT)
  implicit REAL*4 (A-Z)
  if (TIME .ge. TRAMP) then
    OUT=TIME-TRAMP
  else
    OUT=0.0
  end if
  return
end

********************
C
*** TIME CONSTANT ***
C

Subroutine TCONST(Y0, X, TAU, NTIME, NTIM, DELTIM, Y)
  implicit real*4 (A-Z)
  integer*2 NTIME, NTIM
  if (NTIME .ne. NTIM) Y0=Y
  DECAY=exp(-DELTIM/TAU)
  Y=Y0+(X-Y0)*(1.-DECAY)
  if (NTIME .eq. 1) Y=Y0
  NTIM=NTIME
  return
end

*************************
C
***** First Order Derivative Subroutine *****
C

Subroutine DERIV(DELTIM, NTIME, NTIM1, IC2, XM1, NOWVAL, XX, XDM1, XD, XDDM1, XDD, SLOPE)
  implicit REAL*4 (A-Z)
  integer*2 NTIME, NTIM1
  if (NTIME .eq. NTIM1) then
    XX=NOWVAL
  else
    XM1=XX
    XX=NOWVAL
    XDM1=XD
    XDDM1=XDD
  end if
  XD=(XX-XM1)/DELTIM
  if (abs(XD) .lt. 1.E-8) XD=0.
  if (NTIME .eq. 1) XD=IC2
  XDD=(XD-XDM1)/DELTIM
  if (abs(XDD) .lt. 1.E-8) XDD=0.0
  NTIM1=NTIME
  XPRED=XX*XD*DELTIM+XDD*(DELTIM**2)/2.0
if (abs(XPRED) .lt. 1.0E-8) XPRED=0.0
SLOPE=(XPRED-XM1)/(2.0*DELTIM)
if (abs(SLOPE) .lt. 1.0E-8) SLOPE=0.0

C

return
end

***********************************************************************

***** Trapezoidal Integration Subroutine *****
***********************************************************************

Subroutine INTGRAL(NTIME, NTIM2, DELTIM, IC3, PREVAL, NOWVAL, +
CURVAL, OUTOLD, OUTNEW)
implicit REAL*4 (A-Z)
integer*2 NTIME, NTIM2

C

if ((NTIME .eq. NTIM2) .or. (NTIME .eq. 1)) then
CURVAL=NOWVAL
else
PREVAL = CURVAL
CURVAL = NOWVAL
OUTOLD = OUTNEW
end if
if (NTIME .eq. 1) OUTOLD=IC3
OUTNEW = OUTOLD+(CURVAL+PREVAL)*DELTIM/2.
NTIM2=NTIME

C

return
end

***********************************************************************

***** CLEAR SCREEN AND HOME CURSOR *****
***********************************************************************

subroutine CLRSCR
character*1 C1, C2, C3, C4
integer*2 IC(4)
equivalence (C1, IC(1)), (C2, IC(2)), (C3, IC(3)), (C4, IC(4))
data IC/16#1B, 16#5B, 16#32, 16#AA/

C

*** Write Escape Code to Display ***
write(*,1) C1, C2, C3, C4
1 format(1X, 4A1)

C

return
end

***********************************************************************

***** Position Cursor by Row, Column *****
***********************************************************************

subroutine GOTOXY(ROW, COLUMN)
integer*2 IC(4), ROW, COLUMN, L
character*1 C1, C2, C3, C4, C5, C8, LC(5)
character*5 CBUFF
equivalence (C1, IC(1)), (C2, IC(2)), (C3, IC(3)), (C4, IC(4)), +
(CBUFF, LC(1))
data IC/16#1B,16#5B,16#3B,16#66/

C
L=10000+100*ROW+COLUMN

C *** Write Escape Codes to a Character Buffer ***
write(CBUFF,2) L
  2 format(I5)

C *** Write Escape Codes to Display ***
write(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
  3 format(1X,8A1,\))
return
end
APPENDIX D

PHASE PLANE ANALYSIS PROGRAM

This program computes and plots the characteristic slope markers for the reduced order Brushless DC Motor model. Motor parameters are provided by the user. Additionally, the Non-Linear Element performance is plotted. Step and Ramp responses may be analyzed. Plotting density is selectable (1000pts/plt max).

IMPLICIT REAL*4 (A-Z)
COMMON XTIME, Y1, Y2, Y3, Y4, BEGTIM, FINTIM, NPTS, IOPORT, MODEL,
+ XLEN, YLEN, PLEN, PPLANE, XTITLE, YTITLE, PTITLE
REAL*4 X(1010), Y(1010), Y1(1010), Y2(1010), Y3(1010), Y4(1010),
+ XTIME(1010)
INTEGER*2 NTERMS, IOPORT, MODEL, XLEN, YLEN, NTIME, NUMIT, NTIM1,
+ NTIM2, NTIM3, NTIM4, DIR, PMODEL, NPTS, NCTR, PPLANE, PTSPL1,
+ SIM2PL, CTR, ECTR, NUMBER, NDIM, CTR, ELEMNT, PTSPLT, VCTR
LOGICAL*2 WAITNG, TOGGLE
CHARACTER*1 DISOPT, PLOPT
CHARACTER*6 ANSI, ANS11, ANS21, ANS27, PRTSEL, NONLIN
CHARACTER*25 XTITLE, YTITLE
CHARACTER*25 NLCHAR
CHARACTER*51 FRCHCR, PTITLE
C
C ... Introductory Page (1 time good deal!)
call CLRSCR
write(*,4)
PAUSE
C
*** OPEN/READ/CLOSE INPUT DATA FILE ***
C
C
13 open(7,FILE='PHPLANE.INP',STATUS='OLD',ACCESS='SEQUENTIAL')
read(7,1000) PMODEL, PRTCHR
read(7,1020) BEGTIM, FINTIM, MAXITS, SIM2PL
read(7,1022) KP, KV, E0, EDOT0
read(7,1024) R, J, F, KT, KB, KP
read(7,1025) RSLOPE, PERIOD, DBAND
read(7,1028) KA, PTSPLT, XORG, YORG, WFACT
read(7,1030) EMIN, EMAX, EDMIN, EDMAX
close(7,STATUS='KEEP')
C  *****************************************************
C  *** DISPLAY MAIN MENU SELECTIONS ***
C  *****************************************************
C
1  call CLRSCR
   call GOTOXY(8,1)
   write(*,5)
   read(*,'(A)') ANS1
C ... Hardware Options ...
   if ((ANS1 .eq. 'h') .or. (ANS1 .eq. 'H')) then
C
101  call CLRSCR
    call GOTOXY(17,24)
    write(*,*)'*** CURRENT PRINTER SELECTION ***'
    call GOTOXY(20,20)
    write(*,* ) PRTCHR
    call GOTOXY(10,1)
    write(*,105)
    read(*,'(A)') ANS11
C
C ... Printer Options ...
   if ((ANS11 .eq. 'p') .or. (ANS11 .eq. 'P')) then
131  call CLRSCR
    write(*,130)
    read(*,'(A)') PRTSEL
C
if (PRTSEL .eq. '0') then
   PRTCHR='Epson FX-80 Printer, single density'
   PMODEL=0
elseif (PRTSEL .eq. '1') then
   PRTCHR='Epson FX-80 Printer, double density'
   PMODEL=1
elseif (PRTSEL .eq. '2') then
   PRTCHR='Epson FX-80 Printer, dble spd,dual density'
   PMODEL=2
elseif (PRTSEL .eq. '3') then
   PRTCHR='Epson FX-80 Printer, quad density'
   PMODEL=3
elseif (PRTSEL .eq. '4') then
   PRTCHR='Epson FX-80 Printer, CRT Graphics I'
   PMODEL=4
elseif (PRTSEL .eq. '5') then
   PRTCHR='Epson FX-80 Printer, plotter graphics'
   PMODEL=5
elseif (PRTSEL .eq. '6') then
   PRTCHR='Epson FX-80 Printer, CRT Graphics II'
   PMODEL=6
elseif (PRTSEL .eq. '10') then
   PRTCHR='Epson FX-100 Printer, single density'
   PMODEL=7
elseif (PRTSEL .eq. '11') then

291
PRCHR='Epson FX-100 Printer, double density'
PMODEL=11
elseif (FRTSEL .eq. '12') then
PRCHR='Epson FX-100 Printer, dble spd, dual density'
PMODEL=12
elseif (FRTSEL .eq. '13') then
PRCHR='Epson FX-100 Printer, quad density'
PMODEL=13
elseif (FRTSEL .eq. '14') then
PRCHR='Epson FX-100 Printer, CRT Graphics I'
PMODEL=14
elseif (FRTSEL .eq. '15') then
PRCHR='Epson FX-100 Printer, plotter graphics'
PMODEL=15
elseif (FRTSEL .eq. '16') then
PRCHR='Epson FX-100 Printer, CRT Graphics II'
PMODEL=16
elseif (FRTSEL .eq. '20') then
PRCHR='HP 7470A Graphics Plotter'
PMODEL=20
elseif (FRTSEL .eq. '30') then
PRCHR='HP 7475A Graphics Plotter'
PMODEL=30
elseif (FRTSEL .eq. '60') then
PRCHR='HP 2686A Laser Jet Printer'
PMODEL=60

C
C ... Quit the Printer Menu ...
elsei f ((FRTSEL .eq. 'Q') .or. (FRTSEL .eq. 'q')) then
  go to 101
else
  go to 131
endif
  go to 101

C
C ... Quit the Hardware Menu ...
elsei f ((ANS11 .eq. 'q') .or. (ANS11 .eq. 'Q')) then
  go to 1
else
  go to 101
endif

C
C ... Motor Parameters ...
elsei f ((ANS1 .eq. 'm') .or. (ANS1 .eq. 'M')) then

C
201 call CLRSCR
write(*,230) KT,KB,R,L,J,F,KP,KV
read(*,'(A)') ANS21

C
if (ANS21 .eq. 'KT') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for KT-->
read(*,*) KT
go to 201
elseif (ANS21 .eq. 'KB') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for KB-->
read(*,*) KB
go to 201
elseif (ANS21 .eq. 'R') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for R-->
read(*,*) R
go to 201
elseif (ANS21 .eq. 'L') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for L-->
read(*,*) L
go to 201
elseif (ANS21 .eq. 'J') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for J-->
read(*,*) J
go to 201
elseif (ANS21 .eq. 'F') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for F-->
read(*,*) F
go to 201
elseif (ANS21 .eq. 'KP') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for KP-->
read(*,*) KP
go to 201
elseif (ANS21 .eq. 'KV') then
call GOTOXY(24,1)
write(*,'(A)') 'Enter a REAL value for KV-->
read(*,*) KV
go to 201
C ... Quit Motor Parameters Menu ...
elseif (ANS21 .eq. 'Q') then
go to 1
else
go to 201
end if
C
C ... NON-LINEAR ELEMENT SELECTION MENU ...
C
call CLRSCR

call GOTOXY(21,1)
write(*,272) NLCHAR
call GOTOXY(1,1)
write(*,270)
read(*,'(A)') ANS27

C ... RELAY AS NON-LINEAR ELEMENT ...
if ((ANS27 .eq. 'r') .or. (ANS27 .eq. 'R')) then
   ELEMENT=1
   NLCHAR='RELAY'
291  call CLRSCR
   NONLIN='R'
   write(*,290) DBAND
   read(*,'(A)') ANS21

C if (ANS21 .eq. 'DBAND') then
   call GOTOXY(24,1)
   write(*,'(A)') 'Enter a REAL value for DBAND-->
   read(*,*) DBAND
   go to 291
elseif ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then
   go to 14
else
   go to 291
endif
C ... PULSE WIDTH MODULATOR AS NON-LINEAR ELEMENT ...
elself ((ANS27 .eq. 'p') .or. (ANS27 .eq. 'P')) then
   ELEMENT=3
   NLCHAR='PULSE WIDTH MODULATOR'
301  call CLRSCR
   NONLIN='P'
   write(*,300) DBAND,PERIOD,KPWM
   read(*,'(A)') ANS21

C if (ANS21 .eq. 'DBAND') then
   call GOTOXY(24,1)
   write(*,'(A)') 'Enter a REAL value for DBAND-->
   read(*,*) DBAND
   go to 301
elseif (ANS21 .eq. 'PERIOD') then
   call GOTOXY(24,1)
   write(*,'(A)') 'Enter a REAL value for PERIOD-->
   read(*,*) PERIOD
   go to 301
elseif (ANS21 .eq. 'KPWM') then
   call GOTOXY(24,1)
   write(*,'(A)') 'Enter a REAL value for KPWM-->
   read(*,*) KPWM
   go to 301
elseif ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then
   go to 14
else
   go to 301
endif

294
C ... SATURATING AMPLIFIER AS NON-LINEAR ELEMENT ...
    elseif ((ANS27 .eq. 'a') .or. (ANS27 .eq. 'A')) then
        ELEMNT=2
        NLCHAR='SATURATING AMPLIFIER'
        call CLRSCR
        NONLIN='A'
        write(*,280) DBAND, KA
        read(*,'(A)') ANS21
    C
C
    if (ANS21 .eq. 'DBAND') then
        call GOTOXY(24,1)
        write(*,'(A)')'Enter a REAL value for DBAND-->
        read(*,*) DBAND
        go to 281
    elseif (ANS21 .eq. 'KA') then
        call GOTOXY(24,1)
        write(*,'(A)')'Enter a REAL value for KA-->
        read(*,*) KA
        go to 281
    elseif ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then
        go to 14
    else
        go to 281
    endif
C
C elseif ((ANS27 .eq. 'n') .or. (ANS27 .eq. 'N')) then
    go to 400
elseif ((ANS27 .eq. 'q') .or. (ANS27 .eq. 'Q')) then
    go to 1
else
    go to 14
endif
C
C ... Phase Plane Design Menu ...
    elseif ((ANS1 .eq. 'p') .or. (ANS1 .eq. 'P')) then
C
    call CLRSCR
    write(*,260) EMIN, EMAX, EDMIN, EMAX
    read(*,'(A)') ANS21
C
    if (ANS21 .eq. 'EMIN') then
        call GOTOXY(24,1)
        write(*,'(A)')'Enter the minimum value of E-->'
        read(*,*) EMIN
        go to 204
    elseif (ANS21 .eq. 'EMAX') then
        call GOTOXY(24,1)
        write(*,'(A)')'Enter the maximum value of E-->'
        read(*,*) EMAX
        go to 204

elseif (ANS21 .eq. 'EDMIN') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter the minimum value of E DOT-> EDMIN  
    read(*, *) EDMIN  
    go to 204
elseif (ANS21 .eq. 'EDMAX') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter the maximum value of E DOT-> EDMAX  
    read(*, *) EDMAX  
    go to 204
C ... Quit PHASE PLANE Dimensioning Menu ...
else ((ANS21 .eq. 'q') .or. (ANS21 .eq. 'Q')) then  
    go to 1
else  
    go to 204
end if
C ... Simulation Options ...
else ((ANS1 .eq. 'o') .or. (ANS1 .eq. 'O')) then
C
202 call CLRSCR
C
DELTIM = (FINTIM-BEATIM)/(float(PTSPLT)*SIM2PL)
if (FINTIM/DELTIM .gt. MAXITS) DELTIM = FINTIM/MAXITS  
NTERMS = IFIX(FINTIM/DELTIM)+1
C
write(*,240) BEATIM,FINTIM,MATITS,SIM2PL,PTSPLT,E0,EDOT0,  
+ RSLOPE,XORG,YORG,WFACT,DELTIM,NTERMS
read(*, '(A') ANS21
C
if (ANS21 .eq. 'BEATIM') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter a REAL value for BEATIM--> BEATIM  
    read(*, *) BEATIM  
    go to 202
elseif (ANS21 .eq. 'FINTIM') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter a REAL value for FINTIM--> FINTIM  
    read(*, *) FINTIM  
    go to 202
elseif (ANS21 .eq. 'MAXITS') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter a REAL value for MAXITS--> MAXITS  
    read(*, *) MAXITS  
    go to 202
elseif (ANS21 .eq. 'SIM2PL') then  
    call GOTOXY(24,1)  
    write(*, '(A') 'Enter a INTEGER value for SIM2PL--> SIM2PL  
    read(*, *) SIM2PL  
    go to 202
elseif (ANS21 .eq. 'PTSPLT') then
call GOTOXY(24,1)
write(*,'(A!)')'Enter a INTEGER value for PTSPLT--> ' 
read(*,*) PTSPL1
if (PTSPL1.gt.1000) then
  PTSPLT=1000
else
  PTSPLT=PTSPL1
endif
go to 202
elseif (ANS21.eq. 'E0') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for E0 (degrees)-->
  read(*,*) E0
  go to 202
elseif (ANS21.eq. 'EDOT0') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for EDOT0 (deg/sec)-->
  read(*,*) EDOT0
  go to 202
elseif (ANS21.eq. 'RSLOPE') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for RSLOPE (deg/sec)-->
  read(*,*) RSLOPE
  go to 202
elseif (ANS21.eq. 'XORG') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for XORG-->
  read(*,*) XORG
  go to 202
elseif (ANS21.eq. 'YORG') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for YORG-->
  read(*,*) YORG
  go to 202
elseif (ANS21.eq. 'WFACT') then
  call GOTOXY(24,1)
  write(*,'(A!)')'Enter a REAL value for WFACT-->
  read(*,*) WFACT
  go to 202
C ... Quit Simulation Options Menu ...
  elseif (ANS21.eq. 'Q') then
    go to 1
  else
    go to 202
  end if
C ... Save Options to File ...
  elseif ((ANS1.eq. 's') .or. (ANS1.eq. 'S')) then
C C *************************************************************
C C **** OPEN/WRITE/CLOSE INPUT DATA FILE ****
C C *************************************************************
C
    open(7,FILE='PBPLANE.INP',STATUS='NEW')
    write(7,1000) PMODEL,PRTCHR
    write(7,1020) BEGTIM,FINTIM,MAXITS,SIM2PL
    write(7,1022) KP+W,KV,E0,EDOT0
    write(7,1024) RSLOPE,PERIOD,DBAND
    write(7,1026) RSPL,PERIOD, DBAND
    write(7,1028) KA,PTSPLT,XORG,YORG,WFAC
    write(7,1030) EMIN,EMAX,EDMIN,EDMAX
    write(7,2000)
    close(7,STATUS='KEEP')
C
    go to 1
C
    C ... Run the Program ...
    elseif ((ANS1 .eq. 'r') .or. (ANS1 .eq. 'R')) then
goto 2
C
    C ... Quit the Program ...
    elseif ((ANS1 .eq. 'q') .or. (ANS1 .eq. 'Q')) then
stop
else
goto 1
endif
C
    C ... Open an Output Data File ...
    2 open(4,FILE='PHPLANE.OUT',STATUS='NEW')
C
    PI=3.14159
    N=.1
C
    C ... Initializations ...
    NCTR=0
    NPTS=0
    NTIM2=1
    NTIM3=0
    NTIM4=0
    PARG1=0.0
    PARG2=0.0
    FVAL4=0.
    CARG1=0.0
    CARG2=0.0
    CVAL4=0.
    TOT1=0.0
    TOT2=0.0
    TVAL1=0.0
    TVAL2=0.0
    TVAL4=0.
    XM1=0.0
    XX=0.0
    XDM1=0.0

298
XD=0.0
XDDM1=0.0
XDD=0.0
X¹=0.0
F¹=0.0
F2=0.5
X3=0.0
WM1=0.0
WM=0.0
IM=0.
IMO=0.
TSTART=0.0
THEDEG=0.0
TOGGLE=.true.
PPLANE=0
NUMBER=0
NDIM=20

C
C    Preliminary Relationships ...
C    Initial perturbation of THETA for stability check. (0 otherwise)
E0RAD=E0*PI/(180.*N)
EDOT0R=-EDOT0*PI/(180.*N)
if (KP .ne. 0.) WM0=EDOT0R/KP
X2=WM0
if (KP .ne. 0.) THETA0=E0RAD/KP
PTHETA=THETA0
TAU1=L/R
TAU2=J/F

C    call CLRSCR
C    Display the simulation header ...
    call CLRSCR
    call GOTOXY(10,29)
    write(*,*) 'Simulation in Progress'
    DELTIM=(FINTIM-BEGTIM)/(float(PTSPLT)*SIM2PL)
    if (FINTIM/DELTIM .gt. MAXITS) DELTIM =FINTIM/MAXITS
    NTERMS=IFIX(FINTIM/DELTIM)+1

C    Output Simulation Options to Output Data File ...
    write(4,1200) BEGTIM,FINTIM,MAXITS,SIM2PL,DELTIM,KT,KB,R,L,J,F,
+     KP,KV,KPWM,KA,RSLOPE
    write(4,1201) PERIOD,DBAND,E0,EDOT0

C    call GOTOXY(14,1)
    write(*,15) DELTIM, NTERMS
15  format(16X,'Simulation Step Size --> ',F9.8,' seconds',/,
+    17X,'Total Number of Steps--> ',I6)
    call GOTOXY(21,1)

C    if ((NONLIN .eq. 'r') .or. (NONLIN .eq. 'R')) then
        write(*,*),*** NON-LINEAR ELEMENT IS RELAY ***
    elseif ((NONLIN .eq. 'a') .or. (NONLIN .eq. 'A')) then

299
write(*,*), (** NON-LINEAR ELEMENT IS SATURATING AMPLIFIER **) +FIER ***

elseif ((NONLIN .eq. 'p') .or. (NONLIN .eq. 'P')) then
write(*,*), (** NON-LINEAR ELEMENT IS PWM ***)
else
write(*,*), (** NON-LINEAR ELEMENT NOT SELECTED ***)
write(*,*), (** ... Return to Main Menu ... ***)
PAUSE

go to 1
endif

C
SPACE=DELTIM/2.0
C
****************************************************************************
C
*** START MAIN SIMULATION LOOP ***
C
*****************************************************************************

C
DO 100 NTIME=1,NTERMS
TIME=(NTIME-1)*DELTIM
NUMIT=0
THETA=PTHETA
C
****************************************************************************
C
*** START INNER SIMULATION LOOP ***
C
*****************************************************************************

C
200 NUMIT=NUMIT+1
WM=X2
C
THETAF=.1*THETA*180./PI
OMEGAF=.1*WM*180./PI
ORDER=RSLOPE*TIME
POSERR=ORDER-KP*THETAF-KV*OMEGAF
C
... Utilization of Non-Linear Element ...
C
... PULSE WIDTH MODULATOR ...
if ((NONLIN .eq. 'p') .or. (NONLIN .eq. 'P')) then
call PMMOD(TIME,NUMIT,TSTART,PERIOD,TOGGLE,POSERR,
+ DBAND,AGCSAT,AGCCUT,ERRSAT,ERRCUT,VIN,VREF,THRESH,
+ KP=WM)
C
... IDEAL RELAY ...
elseif ((NONLIN .eq. 'r') .or. (NONLIN .eq. 'R')) then
call RELAY(POSERR,DBAND,VIN)
C
... SATURATING AMPLIFIER ...
elseif ((NONLIN .eq. 'a') .or. (NONLIN .eq. 'A')) then
call AMPLIF(POSERR,DBAND,KA,VIN)
else
call GOTOXY(21,1)
write(*,*), (** NON-LINEAR ELEMENT NOT SELECTED ***)
write(*,*), (** ... Return to Main Menu ... ***)
PAUSE

300
C

...Electrical Time Constant due to Inductance Neglected ...
C-----> call TCONST(IM0,EN/R,TAU1,NTIME,NTIM1,DELTIM,IM)
   IM=EN/R
   TM=IM*KT
   call TCONST(WM0,TM/F,TAU2,NTIME,NTIM2,DELTIM,WM1)
call INTEGRAL(NTIME,NTIM4,DELTIM,THETA0,FVAL4,WM1,CVAL4,TVAL4,THET1)
   THETA=THET1
C

******************************
C *** CONVERGENCE CRITERIA (Newton's Method) ***
C

if (NTIME .eq. 1) then
   X3=WM1
   F2=0.
   go to 150
end if
C

if (NUMIT .eq. 1) then
   F2=(WM1-WM)*1.2
   X3=1.001*WM1+1.0E-4
   RELERR=abs(F2/X3)
   go to 310
end if
C

F2=WM1-WM
if (F1 .eq. F2) F2=.999*F2-1.E-8
if (X2 .ne. 0.) RELERR=abs(F2/X2)
if (X2 .eq. 0.) RELERR=1.
if (RELERR .gt. 1.E-8) then
   X3=X2-F2*(X1-X2)/(F1-F2)
endif
310 X1=X2
   X2=X3
   F1=F2
C

WM=X2
C

if (NUMIT .ge. 10) go to 150
if (RELERR .gt. 1.E-8) go to 200
C

******************************
C *** END INNER SIMULATION LOOP ***
C

C 150 call DERIV(DELTIM,NTIME,NTIM3,0.,XM1,WM1,XX,XDM1,XD,XDDM1,XDD,
C

X2=W1+ALPHA*DELTIM
PTheta=Theta+W1*DELTIM
ThetaF=.1*Theta*180./PI
OMEGA=1.*W1*180./PI
DIRLOG=V1/150.
if (ELEMNT.eq. 1) THRESH=POSERR
if (ELEMNT.eq. 2) THRESH=POSERR
if (ELEMNT.ne. 3) VREF=0.

C ... Generate Plotting Arrays ...
if (TIME.ge. BEGTIM) then
    NCTR=NCTR+1
    if ((mod(NCTR,SIM2PL).eq. 0) or. (NCTR.eq. 1)) then
        NPTST=NPTST+1
        X(NPTST)=ORDER-KP*ThetaF
        Y(NPTST)=RSLOPE-KP*OMEGA
        XTIME(NPTST)=TIME
        Y1(NPTST)=POSERR
        Y2(NPTST)=VREF
        Y3(NPTST)=THRESH
        Y4(NPTST)=DIRLOG
    endif
endif

100 CONTINUE

C

********************************
C
*** END MAIN SIMULATION LOOP ***
********************************

C close(4,status='KEEP')

C

********************************
C
***** Plotting selection *****
********************************

C

*** Clear Screen & Home Cursor ***
400 call CLRSCR

C

write(*,1305)
read(*,'(A)') DISOPT
C
if ((DISOPT.eq.'m').or. (DISOPT.eq.'M')) then
    MODEL=99
    IOPORT=99
elseif ((DISOPT.eq.'p').or. (DISOPT.eq.'P')) then
    MODEL=MODEL
    IOPORT=1
C ... IOport=9650 is COM2 ...
if ((MODEL .eq. 20) .or. (MODEL .eq. 30)) IPORT=9650
elseif ((DISOPT .eq. 'r') .or. (DISOPT .eq. 'R')) then
    GO TO 13
elseif ((DISOPT .eq. 's') .or. (DISOPT .eq. 'S')) then

****************************************
C *** OPEN/READ/CLOSE INPUT DATA FILE ***
C ****************************************
C
open(7,FILE='PHPLANE.INP',STATUS='NEW')
write(7,1000) PMODEL,PRCHR
write(7,1020) BEGTIM,FINITIM,MAXITS,SM2PL
write(7,1022) KPWM,KV,E0,EDOT0
write(7,1024) R,L,J,F,KT,KB,KP
write(7,1026) RSLOPE,PERIOD,DBAND
write(7,1028) KA,PITSPLT,XORG,YORG,WFACT
write(7,1030) EMIN,EMAX,EDMIN,EDMAX
write(7,2000)
close(7,STATUS='KEEP')
go to 400

elseif ((DISOPT .eq. 'w') .or. (DISOPT .eq. 'W')) then

C ... Printer ready?? ...
call CLRSCR
call GOTOXY(12,25)
write(*,*) 'Please ensure PRINTER is ready'
call GOTOXY(20,1)
PAUSE
C ...
C Output Simulation Options to Printer ...
open(8,FILE='PRN',STATUS='NEW')
write(8,1200) BEGTIM,FINITIM,MAXITS,SM2PL,DELTIM,
+    KT,KB,R,L,J,F,KP,KV,KPWM,KA,RSLOPE
write(8,1201) PERIOD,DBAND,E0,EDOT0
write(8,1202)
1202 format('1')
close(8,STATUS='KEEP')
go to 400
C
elseif ((DISOPT .eq. 'Q') .or. (DISOPT .eq. 'q')) then
    GO TO 460
else
go to 400
end if
C
410 call CLRSCR
if ((DISOPT .eq. 'p') .or. (DISOPT .eq. 'P')) then
call GOTOXY(20,20)
write(*,*) PRCHR
call GOTOXY(1,1)
endif
write(*,1300)
read(*,'

  C
  if ((FLOPT .eq. 'Q') .or. (FLOPT .eq. 'q')) then
    go to 400
  elseif (FLOPT .eq. 'l') then
    if (ELEMENT .eq. 2) KAPWM=KA
    if (ELEMENT .eq. 3) KAPWM=KPWM
    call PGRAPH(X,Y,NPTS,EMIN,EMAX,EDMIN,EDMAX,DBAND,KT,KP,
      + KV,KB,F,R,J,ANS27,Model,MODEL,ELEMNT,RSlope,
      + XOR,G,ORY,WFACT,PERIOD,DISOPT,BEGTIM,FINTIM)
  elseif (FLOPT .eq. '2') then
    if (ELEMENT .eq. 2) KAPWM=KA
    if (ELEMENT .eq. 3) KAPWM=KPWM
    XTITLE='TIME (sec)'
    XLEN=10
    if (ELEMENT .eq. 3) then
      YTITLE='PWM RESPONSE'
      YLEN=12
    elseif (ELEMENT .eq. 1) then
      YTITLE='RELAY RESPONSE'
      YLEN=15
    elseif (ELEMENT .eq. 2) then
      YTITLE='AMPLIFIER RESPONSE'
      YLEN=19
    endif
  endif
  if ((DISOPT .eq. 'M') .or. (DISOPT .eq. 'm')) then
    call M1GRAPH(ELEMNT)
  elseif ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
    call MGRAPH(DISOPT,KAPWM,DBAND,XORG,ORY,WFACT,ELEMNT)
  endif
else
  go to 410
endif
400 continue
C

**I/O FORMAT STATEMENTS**

4 format(///,24X,'REduced ORDER MODEL PHASE PLANE DESIGN',/.
  + 24X,' LT Vincent S. Rossitto, USN',/.
  + 24X,' Naval Postgraduate School',/.
  + 24X,' March 1987',///)

5 format(32X,'*** MAIN MENU ***',///. 

304
+ 20X,'[H]----> HARDWARE Configuration Menu'./,
+ 20X,'[M]----> MOTOR Parameter Menu'./,
+ 20X,'[N]----> NON-LINEAR Element Selection Menu'./,
+ 20X,'[P]----> PHASE PLANE Dimensioning Menu'./,
+ 20X,'[O]----> OPTIONS for Simulation'./,
+ 20X,'[S]----> SAVE All Changes'./,
+ 20X,'[R]----> RUN Simulation Program'./,
+ 20X,'[Q]----> QUIT the Program'./,
+ 8X,'ENTER SELECTION----->',/)

105  format(30X,'*** HARDWARE MENU ***',/,
+ 20X,'[P]----> PRINTER/PLOTTER configuration change'./,
+ 20X,'[Q]----> QUIT THIS MENU'./,
+ 8X,'ENTER SELECTION----->',/)
1000 format(1X,13,2X,A50)
1020 format(Ix,4F15.7)
1024 format(Ix,7F8.4)
1026 format(Ix,3F15.7)
1028 format(Ix,F12.7,I5,1X,3F12.7)
1030 format(Ix,4F12.4)
2000 format(Ix,'END OF FILE')

C

130  FORMAT(24X,'*** PRINTER OPTIONS MENU ***',/,
+ 13X,'[0] --> Epson FX-80 Printer, single density'./,
+ 13X,'[1] --> Epson FX-80 Printer, double density'./,
+ 13X,'[2] --> Epson FX-80 Printer, dble spd,dual density'./,
+ 13X,'[3] --> Epson FX-80 Printer, quad density'./,
+ 13X,'[4] --> Epson FX-80 Printer, CRT Graphics I'./,
+ 13X,'[5] --> Epson FX-80 Printer, plotter graphics'./,
+ 13X,'[6] --> Epson FX-80 Printer, CRT Graphics II'./,
+ 13X,'[10] --> Epson FX-100 Printer, single density'./,
+ 13X,'[11] --> Epson FX-100 Printer, double density'./,
+ 13X,'[12] --> Epson FX-100 Printer, dble spd,dual density'./,
+ 13X,'[13] --> Epson FX-100 Printer, quad density'./,
+ 13X,'[14] --> Epson FX-100 Printer, CRT Graphics I'./,
+ 13X,'[15] --> Epson FX-100 Printer, plotter graphics'./,
+ 13X,'[16] --> Epson FX-100 Printer, CRT Graphics II'./,
+ 13X,'[20] --> HP 7470A Graphics Plotter'./,
+ 13X,'[30] --> HP 7475A Graphics Plotter'./,
+ 13X,'[60] --> HP 2686A Laser Jet Printer (NPS installation)'./,
+ 13X,'[Q] --> QUIT THIS MENU'./,
+ 3X,'Enter Printer Selection Integer or Q to QUIT ---> ',/)

C

230  format(12X,'*** MOTOR PARAMETER SETTINGS MENU ***',/,
+ 4X,F12.7,1X,'[KT] Motor Torque Constant'./,
+ 4X,F12.7,1X,'[KB] Motor Back EMF Constant'./,
+ 4X,F12.7,1X,'[R] Motor Equivalent Resistance (ohms)'./,
+ 4X,F12.7,1X,'[L] Motor Inductance (henries)'./,
+ 4X,F12.7,1X,'[J] Motor Inertia (oz-in/s^2)'./,
+ 4X,F12.7,1X,'[F] Motor Viscous Friction Coeff.'./,
C 240 format(12X,'*** SIMULATION OPTIONS MENU ***',/, + 1X,F15.7,1X,'[BETTIM] Start Time of Plotting Window',/ , + 1X,F15.7,1X,'[FINITIM] Stop Time of Plotting Window',/ , + 1X,F15.4,1X,'[MAXITS] Max Number of Simulation Iterations',/ , + 13X,13,1X,'[SIM2PL] Ratio: Points Simulated/Points Plotted',/ , + 12X,14,1X,'[PTSPLT] # of Points per Curve per Plot(1000max)',/ , + 1X,F15.7,1X,'[E0] Initial E Perturbation Offset (deg)',/ , + 1X,F15.7,1X,'[EDOTO] Initial E DOT Perturbation Offset(deg/s)',/ , + 1X,F15.7,1X,'[RSLOPE] Ramp Slope (0 for Step; (+) for Ramp)',/ , + 1X,F15.7,1X,'[XORG] X Coordinate of Reference Origin',/ , + 1X,F15.7,1X,'[YORG] Y Coordinate of Reference Origin',/ , + 1X,F15.7,1X,'[WFACT] Plotting Scaling Factor (0.75 nominal)',/ , + 17X,'[Q] QUIT THIS MENU',/// , + 15X,'Computed simulation step size ---> ',F9.8, 'seconds',/ , + 15X,'Computed total number of steps ---> ',I6,/// , + 1X,'Enter variable name (UPPERCASE) or Q to QUIT ---> ',\)
C 260 format(///,16X,'*** PHASE PLANE DIMENSIONING MENU ***',/// , + 1X,F15.7,1X,'[EMIN] Minimum E to be Plotted on Grid',/ , + 1X,F15.7,1X,'[EMAX] Maximum E to be Plotted on Grid',/ , + 1X,F15.7,1X,'[EDMIN] Minimum E DOT to be Plotted on Grid',/ , + 1X,F15.7,1X,'[EDMAX] Maximum E DOT to be Plotted on Grid',/ , + 17X,'[Q] QUIT THIS MENU',/// , + 1X,'Enter variable name (UPPERCASE) or Q to QUIT ---> ',\)
C 270 format(///,8X,'*** NON-LINEAR ELEMENT SELECTION ***',/// , + 10X,'[R] Relay (Bang-Bang)',/ , + 10X,'[P] Pulse Width Modulator',/ , + 10X,'[A] Amplifier (Saturating)',/ , + 10X,'[N] No Trajectory Calculation/ Only Phase PLANE Map',/ , + 10X,'[Q] QUIT THIS MENU/RETURN TO MAIN MENU',/// , + 1X,'Enter Selection ---> ',\)
C 272 format(10X,'CURRENT SELECTION --> ',A30)
C 280 format(///,8X,'*** SATURATING AMPLIFIER SPECIFICATIONS ***',/// , + 1X,F15.7,1X,'[DBAND] Deadband Applied to System Feedback',/ , + 1X,F15.7,1X,'[KA] Amplifier Gain',/ , + 17X,'[Q] QUIT THIS MENU',/// , + 1X,'Enter the selection (UPPERCASE) ---> ',\)
C 290 format(///,15X,'*** RELAY SPECIFICATIONS ***',/// , + 1X,F15.7,1X,'[DBAND] Deadband Applied to System Feedback',/ , + 17X,'[Q] QUIT THIS MENU',/// , + 1X,'Enter the selection (UPPERCASE) ---> ',\)
C 300 format(///,5X,'*** PULSE WIDTH MODULATOR SPECIFICATIONS ***',/// ,
Deadband Applied to System Feedback

Period of PWM Reference Cycle (sec)

PWM Amplifier Gain

QUIT THIS MENU

Enter the selection (UPPERCASE) ---> 

Start Time of Plotting Window

Stop Time of Plotting Window

Max Number of Simulation Iterations

Ratio: Points Simulated/Plotted

Simulation Step Size

Motor Torque Constant

Motor Back EMF Constant

Motor Equivalent Resistance (ohms)

Motor Inductance (henries)

Motor Inertia (oz-in/s^2)

Motor Viscous Friction Coef

Motor Position Feedback Constant

Motor Velocity Feedback Constant

PWM Amplifier Gain

Saturating Amplifier Gain (if used)

Ramp Slope (0 for Step; (+) for Ramp)

Position Feedback Deadband

Initial E Perturbation Offset (deg)

Initial EDOT Perturbation Offset (deg/s)

The following are plotting options

1] PHASE PLANE Trajectory

2] NON-LINEAR Element Performance (PRINT only)

QUIT THIS MENU

Enter selection [1,2,Q] ---> 

Display options:

MONITOR

PRINTER

RETURN TO START-UP MENU (RE-INITIALIZE)

SAVE SIMULATION SPECIFICATIONS TO DISK

WRITE SIMULATION SPECIFICATIONS TO PRINTER

QUIT THE PROGRAM

Enter selection {M,P,R,S,W,Q} ---> 

STOP

END
**Phase Plane Plot**

Subroutine `PGRAPH(X,Y,NPTS,EMIN,EMAX,EDMIN,EDMAX,DBAND,` 
+ `KT,KT,KV,KB,F,R,J,ANS27,IOPORT,MODEL,KA,ELEMNT,RSLOPE,` 
+ `XORG,YORG,WFACT,PERIOD,DISOPT,BEGTIM,FINTIM)`

C

```
C implicit REAL*4 (A-Z)
real*4 X(1010),Y(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,PPLANE,NUMBR,CTR,NCHAR,+ 
    NDIM,ECTR,EDCTR,CNTR,DFLAG,ELEMNT
character*1 DISOPT
character*6 ANS27
character*25 XTITLE,YTITLE
character*51 PTITLE
C
C ...Patience Please !!! ...

  call CLESCR
  call GOTOXY(12,27)
  write(*,*),'Calculating Data for Plot'
  call GOTOXY(20,1)

C

  DELTAX=(EMAX-EMIN)/6.
  DELTAY=(EDMAX-EDMIN)/6.
  N=1
  PI=3.14159
  RAZDEG = N*180./PI
  NDIM=20
  NUMBR=0
C

  if (ELEMNT.eq.1) then
    ESS=RSLOPE*KV/KP
  elseif ((ELEMNT.eq.2) .or. (ELEMNT.eq.3)) then
    ESS=RSLOPE*(F*R+KT*KB+KA*150.*KT*KV*RAZDEG)/+
      (KT*KP*KA*150.*RAZDEG)
  endif
C

  if (ELEMNT.eq.1) then
    if (DBAND .eq. 0.) then
      PTITLE='PHASE PLANE CHARACTERISTICS (IDEAL RELAY)'
      PLEN=42.
    elseif (DBAND .ne. 0.) then
      PTITLE='PHASE PLANE CHARACTERISTICS (RELAY WITH DEADBAND)'
      PLEN=50.
    endif
  elseif (ELEMNT.eq.2) then
    if (DBAND .eq. 0.) then
      PTITLE='PHASE PLANE CHARACTERISTICS (Saturating Amplifier)'
      PLEN=51.
    elseif (DBAND .ne. 0.) then
      PTITLE='PHASE PLANE CHARACTERISTICS (Sat Amp w/ Deadband)'
      PLEN=50.
  endif
```

308
endif

elseif (ELEMNT .eq. 3) then
    if (DBAND .eq. 0.) then
        PTITLE='PHASE PLANE CHARACTERISTICS (PULSE WIDTH MODULATOR)'
PLEN=51.
    elseif (DBAND .ne. 0.) then
        PTITLE='PHASE PLANE CHARACTERISTICS (PWM w/ DEADBAND)'
PLEN=46.
    endif
endif

ASPRAT=.65
CHARHT=.20
PTX=1.5+(6.-PLEN*ASPRAT*CHARHT)/2.
PTY=7.40
NCHAR=ifix(PLEN)
CALL PLOTS(0,IOPORT,MODEL)
call ASPECT(asprat)
cALL FACTOR(WFACT)

if ((DISOPT .eq. 'M') .or. (DISOPT .eq. 'm')) then
call PLOT(1.0,0.,-13)
call NEWPEN(2)
call PLOT(0.,0.,3)
call PLOT(8.,0.,2)
call PLOT(8.,8.,2)
call PLOT(0.,8.,2)
call PLOT(0.,0.,2)
call NEWPEN(1)
elseif ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
    call PLOT(XORG/WFACT,YORG/WFACT,-13)
call PLOT(8.0/WFACT,0.0,2)
call PLOT(8.0/WFACT,6.0/WFACT,2)
call PLOT(0.0,6.0/WFACT,2)
call PLOT(0.0,0.0,2)
call PLOT(5.55/WFACT,6.0/WFACT,3)
call PLOT(5.55/WFACT,0.0/WFACT,0.2)

C ... Specification Summary
    call SYMBOL(6.55/WFACT,4.0/WFACT,28,'SUMMARY',0.,19)
call PLOT(6.55/WFACT,3.95/WFACT,3)
call PLOT(7.35/WFACT,3.95/WFACT,2)
C
if (RSLOPE .eq. 0.) then
    call SYMBOL(5.75/WFACT,3.7/WFACT,22,'Type of Input STEP',0.,19)
elseif (RSLOPE .ne. 0.) then
    call SYMBOL(5.75/WFACT,3.7/WFACT,22,'Type of Input RAMP',0.,19)
endif
C
call SYMBOL(5.75/WFACT,3.4/WFACT,22,'Start Time ',0.,15)
call NUMBER(999.3.4/WFACT.,22,BEGTIM,0.,5)
call SYMBOL(5.75/WFACT,3.1/WFACT,22,'Stop Time',0.,15)
call NUMBER(999.3.1/WFACT,22,FINTIM,0.,5)
call SYMBOL(5.75/WFACT,2.8/WFACT,22,'KF',0.,15)
call NUMBER(999.2.8/WFACT,22,KF,0.,5)
call SYMBOL(5.75/WFACT,2.5/WFACT,22,'KV',0.,15)
call NUMBER(999.2.5/WFACT,22,KV,0.,5)
call SYMBOL(5.75/WFACT,2.2/WFACT,22,'Ess',0.,15)
call NUMBER(999.2.2/WFACT,22,ESS+DBAND,0.,5)
if ((ELEMNT.eq.2) .or. (ELEMNT.eq.3)) then
call SYMBOL(5.75/WFACT,1.9/WFACT,22,'GAIN',0.,15)
call NUMBER(999.1.9/WFACT,22,KA,0.,5)
endif
if (ELEMNT.eq.3) then
call SYMBOL(5.75/WFACT,1.6/WFACT,22,'PWM Freq (Hz)',0.,15)
call NUMBER(999.1.6/WFACT,22,1./PERIOD,0.,1)
endif
C
endif
C ... Draw a Title
   call SYMBOL(PTX,PTY,CHARHT,PTITLE,0.,NCHAR)
C
C ... Redefine origin ...
call FLOT(1.,1.0,-13)
C
CALL STAXIS(.18,CHARHT,10.,080.2)
CALL AXIS(0.0,0.0,0.,1.6.,0.,EMIN,DELTAX)
C
CALL STAXIS(.18,CHARHT,10.,080.1)
CALL AXIS(0.,0.,'E DOT",5.6.,90.,EDMIN,DELTAY)
C
C ... Generation of Limit Lines (Discontinuity)
if (KV .eq. 0.) then
C ... Positive Deadband & Relay Switching ...
   L1X1=DBAND+ESS
   L1X2=DBAND+ESS
   L1Y1=EDMIN+(EDMAX-EDMIN)/float(NDIM)
   L1Y2=EDMAX
C
C ... Negative Deadband ...
   L2X1=DBAND+ESS
   L2X2=DBAND+ESS
   L2Y1=EDMIN+(EDMAX-EDMIN)/float(NDIM)
   L2Y2=EDMAX
C
if (KA .ne. 0.) then
C ... Positive Saturation ...
   L3X1=1./KA+DBAND+ESS
   L3X2=1./KA+DBAND+ESS
   L3Y1=EDMIN+(EDMAX-EDMIN)/float(NDIM)
   L3Y2=EDMAX
C
310
Negative Saturation ...

L4X1=-(1./KA+DBAND)+ESS
L4X2=-(1./KA+DBAND)+ESS
L4Y1=EDMIN+(EDMAX-EDMIN)/float(NDIM)
L4Y2=EDMAX

endif

elseif (KV .ne. 0.) then

Point (X1,Y1) of Line 1 ...

if ((DBAND-EMIN+ESS)*KP/KV .le. EDMAX) then
  L1X1=EMIN+(EDMAX-EMIN)/float(NDIM)
  L1Y1=(DBAND-EMIN+ESS)*KP/KV
elseif ((DBAND-EMIN+ESS)*KP/KV .gt. EDMAX) then
  L1X1=DBAND-EDMAX*KP/KV+ESS
  L1Y1=EDMAX
endif

Point (X2,Y2) of Line 1 ...

if ((DBAND-EMAX+ESS)*KP/KV .ge. EDMIN) then
  L1X2=EMAX
  L1Y2=(DBAND-EMAX+ESS)*KP/KV
elseif ((DBAND-EMIN+ESS)*KP/KV .lt. EDMAX) then
  L1X2=DBAND-((EDMAX-EMIN)/float(NDIM))*KP/KV+ESS
  L1Y2=EDMAX+(EDMAX-EDMIN)/float(NDIM)
endif

Point (X1,Y1) of Line 2 ...

if ((-DBAND-EMIN+ESS)*KP/KV .le. EDMAX) then
  L2X1=EMIN+(EMAX-EMIN)/float(NDIM)
  L2Y1=-(DBAND-EMIN+ESS)*KP/KV
elseif ((-DBAND-EMIN+ESS)*KP/KV .gt. EDMAX) then
  L2X1=-DBAND-EDMAX*KP/KV+ESS
  L2Y1=EDMAX
endif

Point (X2,Y2) of Line 2 ...

if ((-DBAND-EMAX+ESS)*KP/KV .ge. EDMIN) then
  L2X2=EMAX
  L2Y2=(-DBAND-EMAX+ESS)*KP/KV
elseif ((-DBAND-EMAX+ESS)*KP/KV .lt. EDMAX) then
  L2X2=-DBAND-((EDMIN+EDMAX-EDMIN)/float(NDIM))*KP/KV+ESS
  L2Y2=EDMAX+(EDMAX-EDMIN)/float(NDIM)
endif

if (KA .ne. 0.) then

Point (X1,Y1) of Line 3 ...

if ((1./KA+DBAND-EMIN+ESS)*KP/KV .le. EDMAX) then
  L3X1=EMIN+(EMAX-EMIN)/float(NDIM)
  L3Y1=1./KA+DBAND-EMAX*KP/KV+ESS
elseif ((1./KA+DBAND-EMIN+ESS)*KP/KV .gt. EDMAX) then
  L3X1=1./KA+DBAND-EDMAX*KP/KV+ESS
  L3Y1=EDMAX
endif

Point (X2,Y2) of Line 3 ...

311
if ((1./KA+DBAND-EMAX+ESS)*KP/KV .ge. EDMIN) then
  L3X2=EMAX
  L3Y2=(1./KA+DBAND-EMAX+ESS)*KP/KV
elseif ((1./KA+DBAND-EMAX+ESS)*KP/KV .lt. EDMIN) then
  L3X2=1./KA+DBAND-(EDMIN+(EDMAX-EDMIN)/float(NDIM))*KP/ESS
  L3Y2=EDMIN+(EDMAX-EMIN)/float(NDIM)
endif

C ... Point (X1,Y1) of Line 4 ...
if (((-1./KA-DBAND-EMIN+ESS)*KP/KV .le. EDMIN) then
  L4X1=EMIN+(EMAX-EMIN)/float(NDIM)
  L4Y1=(-1./KA-DBAND-(EMIN+(EMAX-EMIN)/float(NDIM)))+ESS)*KP/KV
elseif (((-1./KA-DBAND-EMIN+ESS)*KP/KV .gt. EDMAX) then
  L4X1=1./KA-DBAND-EDMAX*KP/ESS
  L4Y1=EDMAX
endif

C ... Point (X2,Y2) of Line 4 ...
if (((-1./KA-DBAND-EMAX+ESS)*KP/KV .ge. EDMIN) then
  L4X2=EMAX
  L4Y2=(1./KA-DBAND-EMAX+ESS))*KP/KV
elseif (((-1./KA-DBAND-EMAX+ESS)*KP/KV .lt. EDMAX) then
  L4X2=1./KA-DBAND-EDMAX*KP/ESS
  L4Y2=EDMIN+(EDMAX-EDMIN)/float(NDIM)
endif
endif
endif
C
C ... Points are Scaled to Real World Values ...
C ... Line #1
L1X1S=(L1X1-EMIN)/DELTAX
L1Y1S=(L1Y1-EDMIN)/DELTAY
L1X2S=(L1X2-EMIN)/DELTAX
L1Y2S=(L1Y2-EDMIN)/DELTAY
C ... Line #2
L2X1S=(L2X1-EMIN)/DELTAX
L2Y1S=(L2Y1-EDMIN)/DELTAY
L2X2S=(L2X2-EMIN)/DELTAX
L2Y2S=(L2Y2-EDMIN)/DELTAY
C ... Line #3
L3X1S=(L3X1-EMIN)/DELTAX
L3Y1S=(L3Y1-EDMIN)/DELTAY
L3X2S=(L3X2-EMIN)/DELTAX
L3Y2S=(L3Y2-EDMIN)/DELTAY
C ... Line #4
L4X1S=(L4X1-EMIN)/DELTAX
L4Y1S=(L4Y1-EDMIN)/DELTAY
L4X2S=(L4X2-EMIN)/DELTAX
L4Y2S=(L4Y2-EDMIN)/DELTAY
C
if (ELEMNT .eq. 3) then
  ELEMNT=2
  FLAG=1.
endif
C
if (ELEMNT .eq. 3) then
  ELEMNT=2
  FLAG=1.
endif
end

C
C ... Plotting of Dashed Limit Lines ...
call STDASH(.05,.15)
call PLOTD(L1X1S,L1Y1S,3)
call PLOTD(L1X2S,L1Y2S,2)
if (DBAND .ne. 0.) then
call PLOTD(L2X1S,L2Y1S,3)
call PLOTD(L2X2S,L2Y2S,2)
endif
if ((ELEMNT .eq. 2) .and. (KA .ne. 0.)) then
call PLOTD(L3X1S,L3Y1S,3)
call PLOTD(L3X2S,L3Y2S,2)
call PLOTD(L4X1S,L4Y1S,3)
call PLOTD(L4X2S,L4Y2S,2)
endif
C
C ... Phase Plane Slope Marker Generation and Plotting ...
do 415 ECTR=1,NDIM
   E=EMIN+((EMAX-EMIN)/FLOAT(NDIM))*FLOAT(ECTR)
do 425 EDCTR=1,NDIM
   DBFLAG=0
   NUMBR=NUMBR+1
   EDOT=EDMIN+((EDMAX-EDMIN)/FLOAT(NDIM))*FLOAT(EDCTR)
   if (abs(E+EDOT*KV/KP-ESS) .le. DBAND) then
     endif
   if (EDOT .ne. 0.) then
      SLOPEX=J*R*EDOT/DELTAX
      SLOPEY=-(KT*KB+F*R)*(EDOT-RSLOPE)/DELTAY
     elseif (EDOT .eq. 0.) then
      DBFLAG=1
      SLOPEX=0.
      SLOPEY=1.
     endif
   C ... Positive Saturation Region ...
   elseif (((ELEMNT .eq. 1) .and. (E+EDOT*KV/KP-ESS .gt. + DBAND)) .or. ((ELEMNT.eq.2) .and. (((E+EDOT*KV/KP-ESS) + -DBAND)*KA .ge. 1.1)) then
      SLOPEX=J*R*(EDOT)/DELTAX
      SLOPEY=-(150.*KT*KP*RA2DEG+(KT*KB+F*R)*(EDOT-RSLOPE))/
      + DELTAY
   C ... Negative Saturation Region ...
   elseif (((ELEMNT .eq. 1) .and. (E+EDOT*KV/KP-ESS .lt. + -DBAND)) .or. ((ELEMNT.eq.2) .and. (((E+EDOT*KV/KP-ESS) + +DBAND)*KA .le. -1.)) then
      SLOPEX=J*R*(EDOT)/DELTAX
      SLOPEY=-(150.*KT*KP*RA2DEG+(KT*KB+F*R)*(EDOT-RSLOPE))/
      + DELTAY
   C ... Positive LINEAR Region ...
   elseif (((E+(EDOT)*KV/KP)-ESS-DBAND)*KA .gt. 0.) .and.
      (((E+(EDOT)*KV/KP)-ESS-DBAND)*KA .lt. 1.)) then

313
SLOPEX = J*R*(EDOT)/DELTAX
V = 150.*((E+(EDOT-RSLOPE)*KV/KP)*DBAND)*KA
SLOPEY = -(V*KT*KP*RA2DEG+(KT*KB+F*R)*EDOT-RSLOPE)/DELTAY

C ... Negative LINEAR Region

elseif (((E+(EDOT)*KV/KP-ESS)+DBAND)*KA < 0.) .and.
+ (((E+(EDOT)*KV/KP-ESS)+DBAND)*KA > -1.)) then
SLOPEX = J*R*(EDOT)/DELTAX
V = 150.*((E+(EDOT-RSLOPE)*KV/KP)*DBAND)*KA
SLOPEY = -(V*KT*KP*RA2DEG+(KT*KB+F*R)*EDOT-RSLOPE)/DELTAY

endif

C ... Points are Scaled to Real World Coordinates

.VECLN=sqrt(SLOPEX**2+SLOPEY**2)
EIX=(E-EMIN)/DELTAX
E2X=EIX+.15*SLOPEX/VECLN
E1Y=(EDOT-EDMIN)/DELTAY
E2Y=E1Y+.15*SLOPEY/VECLN

C ... Filter out end point for ideal relay

if ((E .ne. ESS) .or. (EDOT .ne. 0.)) then

C ... Filter out end points for relay with Dead Band

if (DBFLAG .eq. 0) then

call PLOT(E1X,E1Y,3)
call PLOT(E2X,E2Y,2)
endif
endif

call SYMBOL(E1X,E1Y,.04,1,0.,-1)

425 continue
415 continue

C ... Overlay a Phase Plane Trajectory

X(NPTS+1)=EMIN
X(NPTS+2)=DELTAX
Y(NPTS+1)=EDMIN
Y(NPTS+2)=DELTAY
call LINE(X,Y,NPTS,1,0,0)

C ... Mark the Start of the Trajectory (Real World Coordinates)

XMARK1= (X(1)-EMIN)/DELTAX
YMARK1= (Y(1)-EDMIN)/DELTAY
call SYMBOL(XMARK1,YMARK1,.12,11,0.,-1)

C ... Mark the Ordered Position (Real World Coordinates)

XMARK2= (ESS+DBAND-EMIN)/DELTAX
YMARK2= (0.-EDMIN)/DELTAY
call SYMBOL(XMARK2,YMARK2,.14,11,0.,-1)

endif

314
CALL PLOT(0,0,0,0,999)
if (FLAG .eq. 1.) then
   ELEMNT=3
   FLAG=0.
endif
RETURN
END

********************************
C ****** PLOTTING SUBROUTINES ******
C ****** (multi-function plot) ******
C ****** (Printer Function) ******
C ****************************

Subroutine MGRAPH(DISOPT,KAPWM,DBAND,X0,Y0,WFACT1,ELEMNT)
implicit REAL*4 (A-Z)
COMMON XTIME,Y1,Y2,Y3,Y4,BEGTIM,FINTIM,NPTS,IOPORT,MODEL,
   + XLEN,YLEN,PLEN,PPLANE,XTITLE,YTITLE,PTITLE
real*4 XTIME(1010),Y1(1010),Y2(1010),Y3(1010),Y4(1010)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,PPLANE,ELEMNT
character*1 DISOPT
character*3 ANS
character*20 XTITLE,YTITLE
character*51 PTITLE
WFACT=(2./3.)*WFACT1
Time axis ...
XTIME(NPTS+1)=BEGTIM
FIRSTX = BEGTIM
XTIME(NPTS+2)=(BEGTIM-FINTIM)/10.
DELTAX = (FINTIM-BEGTIM)/10.
call SCALE(Y1,4.,NPTS,1)
MINY1=Y1(NPTS+1)
DELY1=Y1(NPTS+2)
if (ELEMNT .ne. 3) then
call SCALE(Y3,5.,NPTS,1)
MINY3=Y3(NPTS+1)
DELY3=Y3(NPTS+2)
else
   Y3(NPTS+1)=-1.0
   Y3(NPTS+2)=.2
   MINY3=0.
   DELY3=.2
end
Y2(NPTS+1)=1.0
Y2(NPTS+2)=-2
MINY2=0.
DELY2=.2
endif
Y4(NPTS+D)=-12.
Y4(NPTS+2)=1.
call PLOTS(0,IOPORT,MODEL)
call FACTOR(WFACT)
if (((DISOPT .eq. 'M') .or. (DISOPT .eq. 'm')) then
call PLOT(2.5,1.,-13)
elseif ((DISOPT .eq. 'P') .or. (DISOPT .eq. 'p')) then
call PLOT((X0/WFACT),(Y0/WFACT),-13)
call PLOT(16.0,0.0,2)
call PLOT(0.0,12.0,2)
call PLOT(0.0,0.0,2)
call PLOT(1.3,11.0,-13)
call STAXIS(.20,.27,.16,.080,2)
call AXIS(0.0,0.0,,'TIME (sec)',-10,10,.270.,FIRSTX,DELTAX)
call STAXIS(.20,.27,.13,.080,-1)
call AXIS(11.0,'DIRECTIONAL LOGIC',17.2,0.0,,-1.1.)
call LINE(Y4,XTIME,NPTS,1,0,0)
PTITLE='CLOSED LOOP PERFORMANCE'
call SYMBOL(14.2,-.5,.38,PTITLE,270.,23)
if (ELEMNT .eq. 1) call SYMBOL(14.2,999.,.38,,'(AMP)',270.,6)
if (ELEMNT .eq. 2) call SYMBOL(14.2,999.,.38,,'(RELAY)',270.,8)
if (ELEMNT .eq. 3) call SYMBOL(14.2,999.,.38,,'(PWM)',270.,6)
call SYMBOL(13.6,-1.0,.3,'GAIN = ',.270.,7)
if (ELEMNT .ne. '2') then
call NUMBER(13.6,999.,.3,KAPWM,270.,2)
elseif (ELEMNT .eq. '2') then
call SYMBOL(13.6,999.,.3,236,270.,+1)
endif

call SYMBOL(13.6,-5.0,.3,'DEAD ZONE = ',270.,12)
call NUMBER(13.6,999.,3,DBAND,270.,2)

C
CALL PLOT(0.0,0.0,999)
C
MODEL=99
IPORT=99
C
RETURN
END
C
********************************
C***** PLOTTING SUBROUTINES *****
C******(multi-function plot)*****
C***** (Monitor Function) *****
C********************************

Subroutine M1GRAPH(ELEMNT)

implicit REAL*4 (A-Z)
COMMON XTIME,Y1,Y2,Y3,Y4,BEGTIM,FINTIM,NPTS,IOPORT,MODEL,
+ XLEN,YLEN,PPLANE,XTITLE,YTITLE,PTITLE
real*4 XTIME(IOIO),Y1(IO10),Y2(IO10),Y3(IO10),Y4(IO10)
integer*2 NPTS,IOPORT,MODEL,XLEN,YLEN,PPLANE,ELEMNT
character*25 XTITLE,YTITLE
character*51 PTITLE
C
C...Patience Please !!! ...
call CLRSCR
call GOTOXY(12,27)
write(*,*), 'Calculating Data for Plot'
call GOTOXY(20,1)
C
WFACT=.55
C...Time axis ...
CALL SCALE(XTIME,10.,NPTS,1)
FIRSTX = XTIME(NPTS+1)
XTIME(NPTS+2)=XTIME(NPTS)-XTIME(NPTS+1))/10.
DELTAX = XTIME(NPTS+2)
C
Y4(NPTS+1)=-8.
Y4(NPTS+2)=1.
CALL PLOTS(0,IOPORT,MODEL)
CALL FACTOR(WFACT)
CALL PLOT(2.5,1.,-13)
C
CALL STAXIS(.18,.25,.10,.080,3)
CALL AXIS(0.0,0.0,XTITLE,XLEN,10.,0.,FIRSTX,DELTAX)
C
if (ELEMNT .ne. 3) then
call SCALE(Y3,5.,NPTS,1)
MINY3=Y3(NPTS+1)
DELY3=Y3(NPTS+2)
else
  Y3(NPTS+1)=0.
  Y3(NPTS+2)=.2
  MINY3=0.
  DELY3=.2
endif
CALL AXIS(10.,0.,'ERROR VOLTAGE',-13.5.,90.,MINY3,DELY3)
CALL LINE(XTIME,Y3,NPTS,1,0,0)

C
Y2(NPTS+1)=MINY3
Y2(NPTS+2)=DELY3
CALL STAXIS(.18,.25,10.,.080,2)
CALL AXIS(0.,0.,'REFERENCE SIGNAL',16.5.,90.,MINY3,DELY3)
CALL LINE(XTIME,Y2,NPTS,1,0,0)
C
CALL STAXIS(.18,.25,10.,.080,-1)
CALL AXIS(0.,7.,YTITLE,YLEN,2.,90.,-1.,1.)
CALL LINE(XTIME,Y4,NPTS,1,0,0)
C
CALL PLOT(0.0,0.0,999)
MODEL=99
IOPORT=99
RETURN
END

C ****************************************
C ***** PULSE WIDTH MODULATOR MODULE *****
C *************************************************
C Subroutine PWMOD(TIME,NUMIT,TSTART,PERIOD,TOGGLE,POSERR,+
  DBAND,AGCSAT,AGCCUT,ERRSAT,ERRCUT,VIN,VREF,THRESH,KPWM)
C
IMPLICIT REAL*(A-Z)
INTEGER*2 NUMIT,DIR
LOGICAL*2 WAITING,TOGGLE

C ... Reset the saw-tooth reference signal ...
  if (TIME .ge. TSTART+PERIOD) then
    TSTART=TSTART+PERIOD
    TOGGLE= .true.
  endif
  if (POSERR .gt. (0.+DBAND)) then
    DIR=1
    ERROR=abs(KPWM*(POSERR-DBAND))
  elseif (POSERR .lt. (0.-DBAND)) then
    DIR=-1
    ERROR=abs(KPWM*(POSERR+DBAND))
  else
    if (POSERR .gt. (0.+DBAND)) then
      DIR=1
      ERROR=abs(KPWM*(POSERR-DBAND))
    elseif (POSERR .lt. (0.-DBAND)) then
      DIR=-1
      ERROR=abs(KPWM*(POSERR+DBAND))
    else
      ERROR=abs(KPWM*(POSERR))
    endif
  endif
else
    ERROR=0.
endif
call LIMIT(0.,1.,ERROR,ERROR)
call RAMP(TIME,TSTART,NREF)
VREF=WREF/PERIOD
THRESH=1.-ERROR

"WAITING" is a logical variable indicating whether or not a new
pulse may be generated ...
if (NUMIT .eq. 1) WAITING=TOGGLE

if (WAITING) then
    if (VREF .gt. THRESH) then
        if (DIR .eq. 1) then
            VIN=150.
        TOGGLE=.false.
    elseif (DIR .eq. -1) then
        VIN=-150.
    TOGGLE=.false.
    endif
    elseif (VREF .lt. THRESH) then
        VIN=0.
    TOGGLE=.true.
    endif
endif

return
end

*********************************************************************
***** RELAY MODULE *****
*********************************************************************

Subroutine RELAY(POSSERR, DBAND, VIN)

implicit REAL*4 (A-Z)

if (POSSERR .gt. DBAND) then
    VIN=150.
elseif (POSSERR .lt. -DBAND) then
    VIN=-150.
elseif (abs(POSSERR) .le. DBAND) then
    VIN=0.
endif

return
end

*********************************************************************
***** SATURATING AMPLIFIER MODULE *****
*********************************************************************
Subroutine AMPLIF(POSSERR, DBAND, KA, VIN)

implicit REAL*4 (A-Z)

if (abs(POSSERR) .le. DBAND) then
  VIN = 0.
elseif (((POSSERR - DBAND) * KA .gt. 0.) .and. 
  + ((POSSERR - DBAND) * KA .lt. 1.)) then
  VIN = 150. * (POSSERR - DBAND) * KA
elseif (((POSSERR + DBAND) * KA .lt. 0.) .and. 
  + ((POSSERR + DBAND) * KA .gt. -1.)) then
  VIN = 150. * (POSSERR + DBAND) * KA
elseif ((POSSERR - DBAND) * KA .ge. 1.) then
  VIN = 150.
elseif ((POSSERR + DBAND) * KA .le. -1.) then
  VIN = -150.
endif

return
end

Subroutine LIMIT(RSAT, RCUT, INPUT, OUT)

implicit REAL*4 (A-Z)

if (INPUT .le. RSAT) then
  OUT = RSAT
elseif (INPUT .ge. RCUT) then
  OUT = RCUT
else
  OUT = INPUT
end if

return
end

Subroutine FCNSW(X1, X2, X3, X4, OUT)

implicit REAL*4 (A-Z)

if (X1 .lt. 0.0) then
  OUT = X2
elseif (X1 .eq. 0.0) then
  OUT = X3
else
  OUT = X4
end if
Subroutine STEP(TIME,TSTEP,OUT)
  implicit REAL*4 (A-Z)
  if (TIME .ge. TSTEP) then
    OUT=1.0
  else
    OUT=0.0
  end if

return
end

Subroutine DEADS(P1,P2,VSGDEL,VSGERR)
  implicit REAL*4 (A-Z)
  if (VSGDEL .gt. P2) then
    VSGERR=VSGDEL-P2
  elseif (VSGDEL .lt. P1) then
    VSGERR=VSGDEL-P1
  else
    VSGERR=0.0
  end if

return
end

Subroutine RAMP(TIME,TRAMP,OUT)
  implicit REAL*4 (A-Z)
  if (TIME .ge. TRAMP) then
    OUT=TIME-TRAMP
  else
    OUT=0.0
  end if

return
end

Subroutine TCONST(Y0,X,TAU,NTIME,NTIM,DELTIM,Y)
  implicit real*4 (A-Z)
  integer*2 NTIME,NTIM
  if (NTIME .ne. NTIM) Y0=Y
DECAY=exp(-DELTIM/TAU)
Y=Y0+(X-Y0)*(1.-DECAY)
if (NTIME .eq. 1) Y=Y0
NTIM=NTIME
return
end

*********************************************
C ***** First Order Derivative Subroutine *****
*********************************************
Subroutine DERIV(DELTIM,NTIME,NTIM1,IC2,XM1,NOWVAL,XX,XDM1,+
XD,XDDM1,XDD,SLOPE)
implicit REAL*4 (A-Z)
integer*2 NTIME,NTIM1

if (NTIME .eq. NTIM1) then
XX=NOWVAL
else
XM1=XX
XX=NOWVAL
XDM1=XD
XDDM1=XDD
end if

XD=(XX-XM1)/DELTIM
if (abs(XD) .lt. 1.E-8) XD=0.
if (NTIME .eq. 1) XD=IC2
XDD=(XD-XDM1)/DELTIM
if (abs(XDD) .lt. 1.E-8) XDD=0.0

NTIM1=NTIME

time XRED=XX*XD*(DELTM+XDD*(DELTM**2)/2.0
if (abs(XRED) .lt. 1.E-8) XRED=0.0
SLOPE=(XRED-XM1)/(2.0*DELTIM)
if (abs(SLOPE) .lt. 1.E-8) SLOPE=0.0

return
end

*********************************************
C ***** Trapezoidal Integration Subroutine *****
*********************************************
Subroutine INTGRL(NTIME,NTIM2,DELTIM,IC3,PREVAL,NOWVAL,+
CURVAL,OUTOLD,OUTNEW)
implicit REAL*4 (A-Z)
integer*2 NTIME,NTIM2

if (NTIME .eq. NTIM2) then
CURVAL=NOWVAL
else
PREVAL = CURVAL
CURVAL = NOWVAL
OUTOLD = OUTNEW
end if
if (NTIME .eq. 1) OUTOLD=IC3
OUTNEW = OUTOLD+(CURVAL+PREVAL)*DELTIM/2.
NTIM2=NTIME
return
end
C
*****************************************************************
C ***** CLEAR SCREEN AND HOME CURSOR *****
C*****************************************************************
subroutine CLRSCR
character*1 C1,C2,C3,C4
integer*2 IC(4)
equivalence (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
data IC/16#1B,16#5B,16#32,16#4A/
C
C *** Write Escape Code to Display ***
write(*,1) C1,C2,C3,C4
1 format(1X,4A1)
C
return
end
C
*****************************************************************
C ***** Position Cursor by Row,Column *****
C*****************************************************************
subroutine GOTOXYC(ROW,COLUMN)
integer*2 IC(4),ROW,COLUMN,L
character*1 C1,C2,C5,C8,LC(5)
character*5 CBUFF
equivalence (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),
+ (CBUFF,LC(1))
data IC/16#1B,16#5B,16#32,16#66/
C
L=10000+100*ROW+COLUMN
C
C *** Write Escape Codes to a Character Buffer ***
write(CBUFF,2) L
2 format(I5)
C
C *** Write Escape Codes to Display ***
write(*,3) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
3 format(1X,8A1,\)
return
end
LIST OF REFERENCES


5. Gerba, Alex, Jr., "Simulation and Performance of a Brushless DC Motor for Cruise Missile Fin Position Control" (Progress Report to NWC, China Lake, Ca.), April 1986.


BIBLIOGRAPHY


<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Initial Distribution List</th>
</tr>
</thead>
</table>
| 1.  | 2      | Defense Technical Information Center  
Cameron Station  
Alexandria, Virginia  22304-6145 |
| 2.  | 2      | Library, Code 0142  
Naval Postgraduate School  
Monterey, California  93943-5002 |
| 3.  | 1      | Department Chairman, Code 62  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, California  93943 |
| 4.  | 5      | Professor Alex Gerba, Jr., Code 62Gz  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, California  93943 |
| 5.  | 3      | Professor George J. Thaler, Code 62Tr  
Department of Electrical and Computer Engineering  
Naval Postgraduate School  
Monterey, California  93943 |
| 6.  | 1      | Professor Robert H. Nunn, Code 69Nn  
Department of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California  93943 |
| 7.  | 3      | Naval Weapons Center, China Lake  
Weapons Power System Branch  
Code 3275  
ATTN: R.F. Dettling  
China Lake, California  93555 |
| 8.  | 1      | LT Vincent S. Rossitto, USN  
272 Ball Pond Rd.  
New Fairfield, Connecticut  06812 |
Thesis R7775 c.1

Rossitto
Pulse width modulator controller design for a brushless dc motor position servo.