156
26

# REAL-TIME DIGITAL SIMULATION OF THE GENERATOR MODEL

by

Yujie Irene Lu

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

A.G. Phadke, Chairman

J. De La Ree Lopez

L.M. Mili

September 1990

Blacksburg, Virginia

LD
5655
V855
1990
L82
C.2

# REAL-TIME DIGITAL SIMULATION OF THE GENERATOR MODEL

by

Yujie Irene Lu

A.G. Phadke, Chairman

Electrical Engineering

(ABSTRACT)

This thesis is in an attempt to realistically model a real-time digital generator which interfaces to an analog system simulator and which consists of the synchronous machine and its peripheral controllers such as the exciter and the governor-turbine subsystems. In this work, the exciter, the synchronous machine, the machine dynamics and the governor are modeled in detail while a simplified model of the turbine is used.

The synchronous machine, the main component of this simulation, solves the discretized Park's machine equations which include flux derivative terms and terms pertaining to the two amortisseur windings. Treatment of saturation effects in the mutual inductances is also discussed. The Park's model is arranged to obtain a field voltage and machine armature current input - machine terminal voltage output structure, where the armature current and terminal voltage are rotor based quantities (i.e. in d-q domain). In order to interface the Park's machine model to the analog system model, the Park's and inverse Park's transformation are implemented by software modules.

The implementation of a prototype model generator using a Motorola 68020 microprocessor and fast computer peripherals is discussed. The results of the digital computer simulation in real-time for the generator model under various operating conditions are presented.

# Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. A.G. Phadke, for his professional guidance and constant encouragement. His intellectual and financial support was indispensable for the successful completion of this thesis. I would also like to thank Dr. J. De La Ree Lopez and Dr. L.M. Mili for serving on my graduate committee. They were constant sources of inspiration to me. Appreciation is expressed to Mrs. P.A. Manning, for her assistance and friendship. I am greatly thankful to the graduate students of the Power System Laboratory for their help and friendship, with special thanks to N. Vichare for allowing me to use his interface software and assisting me with hardware equipment. To the following persons go my sincere thanks; Star Wu and Vineeta Gangla, for doing the thesis proof reading; Dr. Yilu Liu, Weiping Li and Janie Liu, for helping me through my years of graduate school. Thanks are also due to the secretaries of the EE main office who cheerfully let me use the conference room for presenting this thesis.

Thanks to my Mother and Father, Grandmother and Grandfather, for standing behind my decisions.

# Table of Contents

**Introduction** . . . . . 1

1.1 Real-time Power System Simulation . . . . . 1

1.1.1 Power System Scale Models . . . . . 1

1.1.2 Digital Computer Simulation . . . . . 2

1.2 Comparison between Physical Scale Model (TNA) and Digital Computer Simulation . . . . 3

1.2.1 The Transient Network Analyzer (TNA) . . . . . 3

1.2.2 Real-time Digital Computer Simulation . . . . . 4

1.3 General View of the Thesis . . . . . 4

1.4 Outline of Thesis . . . . . 9

**Simulation Model** . . . . . **10**

2.1 Exciter Model . . . . . 10

2.2 Synchronous Machine Model . . . . . 15

2.2.1 Physical Description . . . . . 15

2.2.2 Park's Model . . . . . 17

2.2.3 Flux Linkage Equations . . . . . 21

2.2.3.1 Stator Inductances . . . . . 21

2.2.3.2 Rotor Inductances . . . . . 22
2.2.3.3 Stator-to-rotor Mutual Inductances . . . . . 23
2.2.3.4 Transformation of Stator Fluxes . . . . . 24
2.2.4 Voltage Equations . . . . . 25
2.2.5 Equivalent Circuits . . . . . 30
2.2.6 Mathematical Model . . . . . 35
2.3 Governor-turbine Model . . . . . 46
2.4 Mechanical Model . . . . . 49

**Simulation Algorithm . . . . . 56**
3.1 Digital Non-linear System Simulation . . . . . 56
3.2 Simulation Scheme . . . . . 57
3.3 Discretization of Exciter Model . . . . . 59
3.4 Discretization of Synchronous Machine Model . . . . . 63
3.5 Discretization of Governor-turbine Model . . . . . 65
3.6 Discretization of Mechanical Model . . . . . 70

**Transformation of Input and Output . . . . . 73**
4.1 Park's and Inverse Park's Transformation . . . . . 75
4.2 Microprocessor Implementation . . . . . 77

**Implementation and Simulation Results . . . . . 89**
5.1 Hardware Organization . . . . . 89
5.2 Software Organization . . . . . 90
5.3 Analog Interface . . . . . 96
5.4 Computational Workload . . . . . 97
5.5 Simulation Results . . . . . 101
5.5.1 Open-loop Test . . . . . 105

5.5.2 Closed-loop Test ..... 105
5.5.3 Constant Voltage Test ..... 105
5.5.4 Exciter Subsystem Tests ..... 106

**Conclusions and Suggestions for Future Research ..... 113**
6.1 Conclusions ..... 113
6.2 Suggestions for Further Research ..... 114

**References ..... 116**

**Inductances and Time Constants of Machine ..... 118**

**Program listings ..... 120**

**Nomenclature ..... 172**

**Vita ..... 177**

# List of Illustrations


Figure 1. Structure of model generator . . . . . 6

Figure 2. Simulation scheme . . . . . 7

Figure 3. The block diagram of IEEE Type-I excitation model . . . . . 12

Figure 4. Saturation function and exciter saturation . . . . . 14

Figure 5. Pictorial representation of the synchronous machine . . . . . 20

Figure 6. Schematic diagram of a synchronous machine . . . . . 28

Figure 7. Equivalent circuits of synchronous machine . . . . . 31

Figure 8. Structure of equivalent circuits of Solid Iron Rotor Model . . . . . 32

Figure 9. Solid Iron Rotor Model equivalent circuits . . . . . 34

Figure 10. Solid Iron Rotor Model equivalent circuits in Thevenin equivalent form . . . . . 36

Figure 11. Block diagram for simulation of Solid Iron Rotor Model . . . . . 38

Figure 12. Step by step derivation of d-axis transfer function . . . . . 39

Figure 13. Step by step derivation of q-axis transfer function . . . . . 40

Figure 14. Construction used for computing saturation . . . . . 43

Figure 15. The approximate saturation function . . . . . 45

Figure 16. The block diagram of electro-hydraulic speed-governing and steam turbine system 47

Figure 17. The block diagram of mechanical model . . . . . 54

Figure 18. Non-linear, realization block diagram of the IEEE Type-I exciter system . . . . . 60

Figure 19. Non-linear, realization block diagram of Solid Iron Rotor Model for d-axis . . . . . 66

Figure 20. Non-linear, realization block diagram of Solid Iron Rotor Model for q-axis . . . . 67

Figure 21. Non-linear, realization block diagram of the Westinghouse EH speed-governor . . 68

Figure 22. Non-linear, realization block diagram of the mechanical system ..... 71
Figure 23. Typical generator simulation scheme ..... 74
Figure 24. Block diagram of Park Transformation module (PT) ..... 83
Figure 25. Block diagram of Inverse Park Transformation module (IPT) ..... 84
Figure 26. Table look up $\theta$ or $\phi$ in 1st & 2nd quadrant ..... 85
Figure 27. Table look up $\theta$ or $\phi$ in 3rd & 4th quadrant ..... 86
Figure 28. The extrapolation for a piecewise linear function ..... 87
Figure 29. The extrapolation for a sine function ..... 88
Figure 30. Hardware organization ..... 91
Figure 31. Software organization ..... 93
Figure 32. Pre and post filtered waveforms ..... 98
Figure 33. Two-stage RC filter with a cut-off frequency of 382 Hz ..... 99
Figure 34. Bode plots of transfer function H(j$\omega$) ..... 100
Figure 35. Microprocessor generated three phase voltage waveforms ..... 107
Figure 36. Trajectory of $V_T$ for unit step input (open-loop) ..... 108
Figure 37. Trajectory of $V_T$ for unit step input (closed-loop) ..... 109
Figure 38. Plots for constant voltage test ..... 110
Figure 39. Trajectory of $E_{FD}$ for unit step input (open-loop) ..... 111
Figure 40. Trajectory of $E_{FD}$ for unit step input (closed-loop) ..... 112

# List of Tables

Table 1. Typical constants of rotating rectifier exciter with static voltage regulator ..... 16
Table 2. Typical parameters of equivalent circuits of Solid Iron Rotor Motor ..... 41
Table 3. Typical parameters of synchronous machine ..... 42
Table 4. Speed-governing system and turbine parameters for Figure 16 (B) ..... 48
Table 5. Mechanical parameters ..... 52
Table 6. Arithmetic Work Per Timestep ..... 102
Table 7. Arithmetic Operation Times ..... 103
Table 8. Arithmetic execution times ..... 104

# Chapter I

# Introduction

## *1.1 Real-time Power System Simulation*

The capability of real-time simulation is essential to study certain power system problems, such as power system stability, operation, control, protection etc.. Real-time power system simulation can be accomplished in two ways [20]:

1. Simulation by using scale models, and
2. Simulation by using a digital computer.

### 1.1.1 Power System Scale Models

Traditionally, real-time power system simulation is done on a scaled down physical model of a power system, known as a Transient Network Analyzer (TNA) [20]. It consists of synchronous

machines, passive component $\pi$ section models of transmission lines, transformers, circuit breakers etc.. These elements can be connected together to form an electrical analog of some subset of a power system subjected to various operating conditions. Then, the response of the model to certain operating conditions match that of the power system under study. Transient analysis is best studied by using the TNA. Before the widespread use of digital computers, the TNA was extensively used. Now it is still being applied in some situations where accurate digital simulation is either impossible or computationally expensive.

### 1.1.2 Digital Computer Simulation

The other type of real-time digital simulation is the process of solving the system difference equations using the system inputs and producing results of system calculations. It is implemented by computer programs which model the system. The time interval between the solution points is the same as the time step used in the input signals. This simulation process requires the following conditions:

1. A knowledge of the mathematical model for the system.
2. A fast, effective calculation algorithm with a suitable time step that is adequate for reasonable accuracy in solution.
3. Accurate and fast measurement of system state variables and conversion of calculated values into system inputs.

Currently, digital computer software packages work on transient and dynamic stability analysis, machine modeling, network analysis, development and testing of protective devices and so on. These programs are powerful tools for the study of power system planning, operation, protection and control.

## *1.2 Comparison between Physical Scale Model (TNA) and Digital Computer Simulation*

### 1.2.1 The Transient Network Analyzer (TNA)

The advantages offered by a TNA are as follows [3]:

- The phenomena take place in real-time providing an actual time frame for the system under consideration.
- Certain results are considered to be closer to the real system since they are obtained through physical components rather than through a mathematical model.

However, TNA suffers from the following drawbacks [3]:

- Inflexibility. Great care must be taken within setting up each simulator for an experiment, which is a time consuming process. In addition, only a limited number of system configurations can be simulated.
- If simulating a moderate size power system, the models take up considerable space and are quite expensive.
- Certain component models may not exactly reproduce the behavior of the actual system components they represent.

In spite of these disadvantages, TNA has continued to play an important role in transient analysis and can be expected to maintain that position for several more years.

### 1.2.2 Real-time Digital Computer Simulation

With the advent of fast 16 and 32 bit microprocessors, high precision analog-to-digital (A/D) and digital-to-analog (D/A) converters, real-time digital computer simulation becomes possible. Each component of the system can be solved in real-time in the digital domain by a microprocessor, and then interfaced with the simulation system through A/D and D/A converters. This approach has several advantages over physical scale models:

1. The mathematical model and the element parameter values are easily modified or changed since they are encoded in the software.
2. Due to the low cost of digital hardware, digital computer implementation of a system model is far less expensive than the TNA implementation described previously.
3. Having the state variables and other information available in digital form simplifies data logging and interfacing to larger computers.

Recognizing this exciting prospect, a power system simulator based on the real-time digital simulation concept was built at Virginia Tech. several years ago [3]. This simulation attempts to refine the synchronous generator system, one of the main element of the simulator, since the existing generator is a simplified model.

## *1.3 General View of the Thesis*

The general block diagram of the synchronous generator system model is shown in Figure 1. The machine terminal voltage $V_T$ is controlled by the exciter and machine armature currents $I_d$ and $I_q$, the rotor speed $\omega$ by the speed governor-turbine subsystem, and the rotor dynamics $\Delta\omega$ and $\Delta\delta$

by the mechanical subsystem (i.e. swing equation). The structure and the transfer functions of these subsystems have been identified and their characteristics are well known [1,4,9].

This thesis focuses on the six subsystems, which are: exciter, synchronous machine, governor-turbine, mechanical module, Park's transformation and its inverse (see Figure 1). The simulation scheme is organized as shown in Figure 2. The simulation of the synchronous generator system must have the following capabilities:

- The real-time simulation of the whole system.
- A measurement technique to sense the samples of phase currents $i_a$ , $i_b$ and $i_c$ , and voltages $V_a$, $V_b$ and $V_c$ from a real power system or a simulated power system at each 1/720 second. These samples form the input of the synchronous machine after Park's transformation.
- Ability of setting the required initial or reference values, like $V_{ref}$ , $\omega_{ref}$ , $\delta_{ref}$ , $P_0$ etc.
- Capability of varying the generated voltage amplitude. The exciter subsystem makes use of this facility.
- Provision of changing the generated frequency. The governor-turbine model makes use of this.
- Function for changing the machine torque angle $\delta$, which would be necessitated by the mechanical simulation.
- Capacity of generating the smooth sinusoidal three phase voltage, i.e. the machine terminal voltage, with a minimum amount of distortion and harmonics by inverse Park's transformation, and simultaneously sending the voltage, through D/A converters, to the three phase power system simulator.

These stringent requirements are very difficult to be realized by using an analog circuit, but can be easily achieved through a microprocessor operating in real-time.

The exciter is modeled in complete detail with the continuously acting regulator and exciter model [1]. The magnitude of the terminal voltage of the machine, $V_T$ (discussed later) is compared with the voltage of the reference set point $V_{ref}$, and this error signal is input to the exciter system module

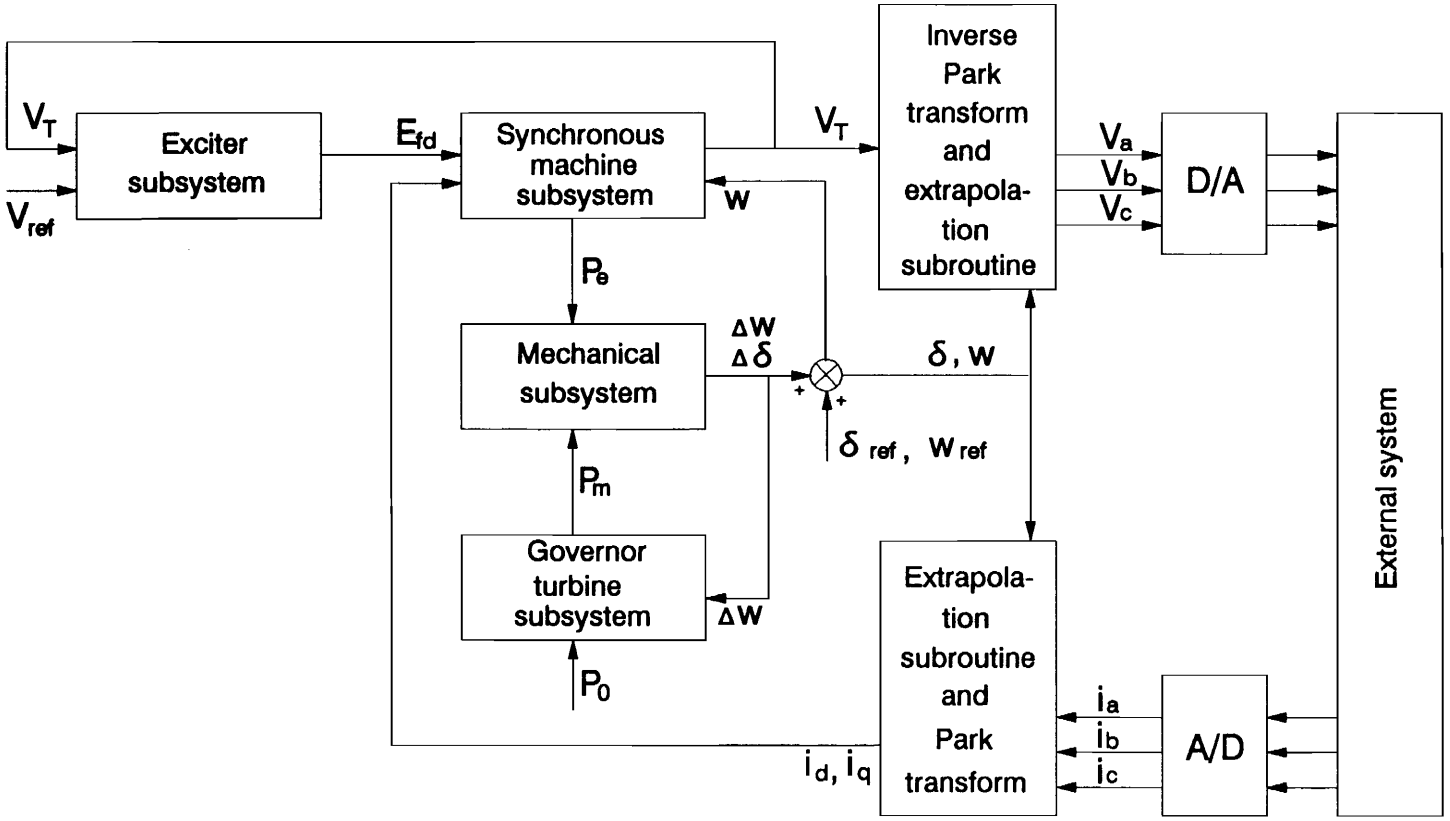
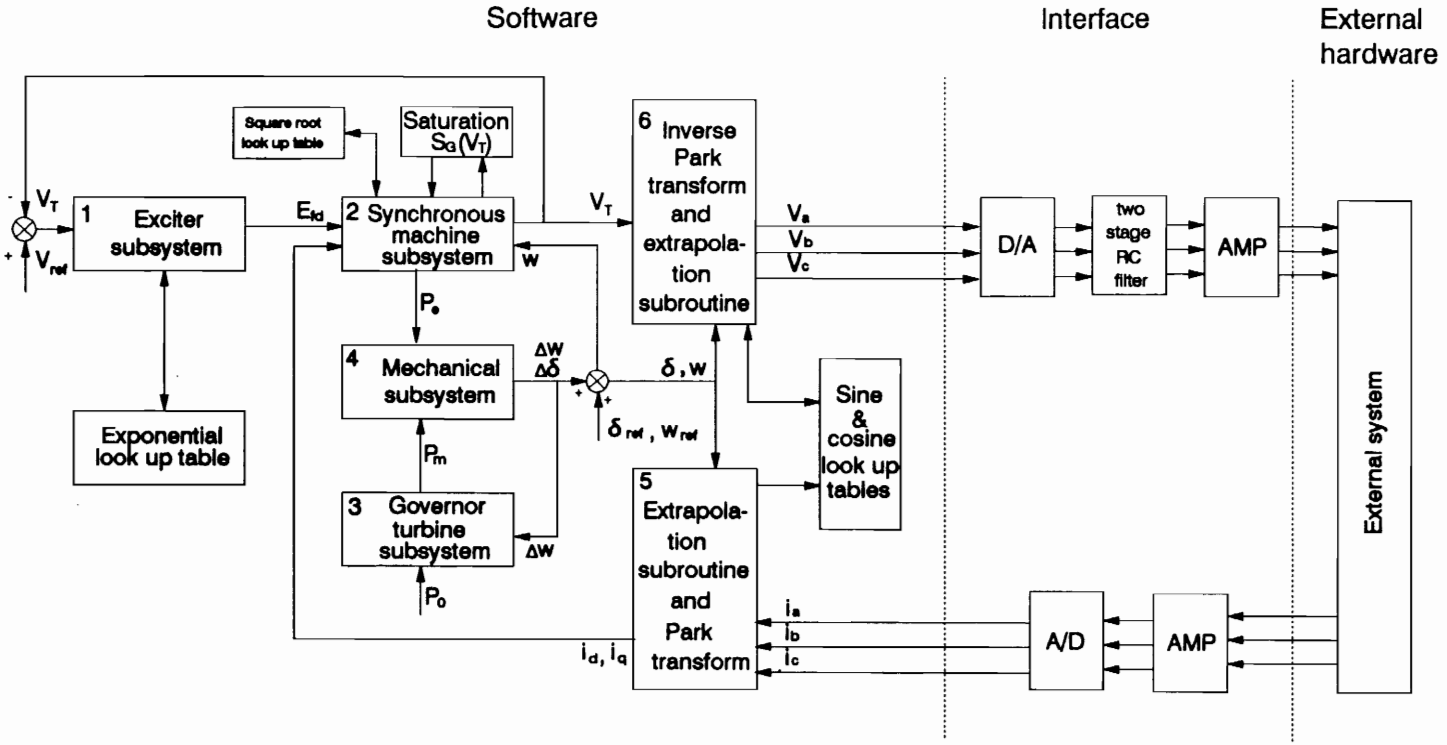



Figure 1. Structure of model generator




| Module | Function | Sampling rate |
|---|---|---|
| 1 | Solves exciter subsystem equations | 720 Hz |
| 2 | Solves synchronous machine subsystem equations | 720 Hz |
| 3 | Solves governor-turbine subsystem equations | 720 Hz |
| 4 | Solves swing equation | 720 Hz |
| 5 | Generates d and q axis currents to be fed into synchronous machine module | 720 Hz |
| 6 | Generates output waveform to be fed into the external system | 2880 Hz |

Processor: MC 68020

Coding: Assembly

Figure 2. Simulation scheme

(ES). The output of this module is the field voltage $E_{fd}$, which is one of the inputs of the synchronous machine system.

In general, the synchronous machine is represented in the direct and quadrature axis (d & q) quantities by flux or voltage equations. The transformation from the phase (a, b & c) quantities to d, q-axis quantities is performed by the Park's transformation. In order to simplify analysis, the synchronous machine equations are usually expressed in state space form. For real-time simulation, the machine fluxes form the state vector, the axis currents are the inputs and the axis voltages are the outputs, because it is more convenient to generate a voltage signal by digital-to- analog (D/A) converters than to produce a current signal. In this work, a solid iron rotor model for the synchronous machine [4] is used. The field voltage $E_{fd}$, from the model ES, is inputed to the synchronous machine subsystem (SMS). SMS also needs the machine armature currents $I_d$ and $I_q$ as inputs. This module will produce the internal voltage of the machine, which is treated as the terminal voltage of the machine $V_T$. In addition, the saturation of the machine is calculated as a function of armature flux linkage $\psi$, represented as $S_G = f(\psi)$ in this module.

The governor-turbine subsystem can be considered to be a cascade of low pass filters, each filter block represents a reheat pass of the steam. Nonlinearities like rate and position limits are imposed on the governor valve or gate opening [9]. Here, the governor is modeled as the Westinghouse electro-hydraulic governor with steam feedback speed-governing system, and the steam turbine is simulated as a simplified model, i.e. a nonreheat first-order system. The inputs of the governor-turbine subsystem (GTS) are the angular speed deviation $\Delta\omega$ and the initial mechanical power $P_0$. The response of the module GTS, i.e. mechanical power $P_m$, will be forwarded to the mechanical subsystem.

The mechanical dynamics of the synchronous machine is very complex, but usually it can be described by the rotor dynamic equation, known as the swing equation. This is a second order nonlinear differential equation, which governs the motion of the rotor based on the rotor dynamics. This subsystem is implemented by module MS. The mechanical power $P_m$ is compared with the

output electric power $P_e$ of the machine, and this error signal, $(P_m - P_e)$, is input to the MS module. The outputs of the module are the angular speed deviation $\Delta\omega$ and the torque angle deviation $\Delta\delta$.

This thesis also discusses the Park's transformation and its inverse, of currents and voltages at the terminal of the machine, or at the terminal bus of the system simulator. Transforming the three phase currents and voltages of the system to the demodulated quantities in the d-q domain is implemented by the Park's transformation module PT, and the inverse Park's transformation module IPT transfers the machine terminal voltage $V_T$ from the d-q domain to the three phase voltages $V_a$, $V_b$ and $V_c$.

## *1.4 Outline of Thesis*

This chapter has provided a brief description of the simulation module. Chapter II focuses on developing four simulation models, which are the exciter, the synchronous machine, the governor-turbine and mechanical subsystems. The simulation algorithm and the discretization for the four models are described in Chapter III. The modeling method and the implementation of the Park's transformation and its inverse for the inputs and outputs of this simulation system are discussed in Chapter IV. Chapter V explains the software and hardware implementation of the above simulation algorithm and presents the simulation results. And in the last chapter, the conclusions and suggestions are summarized.

# Chapter II

# Simulation Model

A typical machine system simulation model consists of six interconnected subsystems, namely exciter, synchronous machine, governor-turbine, mechanical model, Park's transformation and inverse Park's transformation. These simulation models have to be represented correctly. This chapter focuses on the first four subsystems, while Park's transformation and its inverse are modeled in Chapter IV.

## *2.1 Exciter Model*

The IEEE has standardized some exciter models which can be used to represent the different excitation systems [1]. The Type-I exciter model, i.e. the continuously acting regulator and exciter model, has been chosen for this research. This system is "representative of the majority of the modern systems now in service and presently being supplied." [1]

Figure 3 shows the block diagram of the Type-I exciter model. [1, 3]. The transfer functions and nomenclature of Type-I will be described in detail. $V_T$ is the generator terminal voltage magnitude available as the input to the regulator. The first transfer function is a simple time constant $T_r$ representing regulator input filtering. Usually, $T_r$ is very small and can be assumed to be zero. The first summing point compares the regulator reference with the output of the input filter to determine the voltage error input to the regulator amplifier. The second summing point compares the excitation major loop damping signal with the voltage error at the first summing point, where the loop damping is provided by the feedback transfer function $s K_f / (1 + s T_f)$ from exciter output $E_{fd}$ to the first summing point. Next comes the regulator with a gain of $K_a$ and a time constant of $T_a$. The output of the regulator is clamped between two levels. Then this signal is compared with the saturation function of the exciter and the error is fed to the exciter transfer function $1 / (K_e + s T_e)$. There are two non-linear functions in the exciter model. One is the clamp at the output of the regulator, the other is the saturation function of the exciter. The former is included to keep the regulator output within practical limits (for this research, the upper limit $V_{Rmax} = 7.3$ and the lower limit $V_{Rmin} = -7.3$ ). Since the gain of the regulator is quite large ($K_a \simeq 400$) and its time constant relatively small ($T_a \simeq 0.02 sec.$), that portion of the transfer function is very sensitive to large error inputs.

Saturation causes increased excitation requirements and it is dependent on the exciter output. As shown in Figure 4, the saturation function is defined as follows:

$$S_E = (x - y) / y \tag{2.1}$$

where x and y are points on the constant resistance load saturation line (RL) and the airgap line (AG), respectively, for the same field voltage. This function can be described by the following exponential function

$$S_E = f(E_{fd}) = P \times \exp(Q \times E_{fd}) \tag{2.2}$$

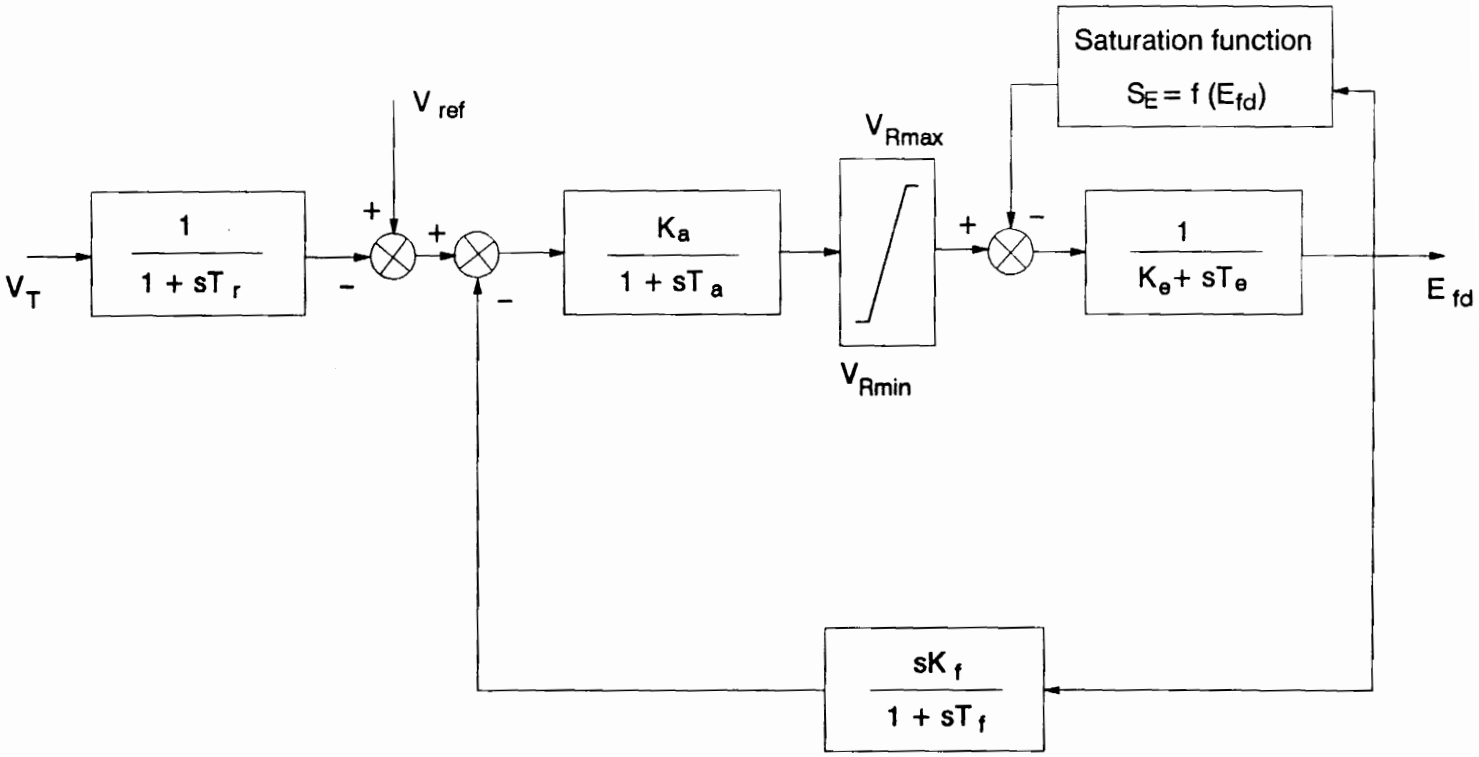



Figure 3. The block diagram of the IEEE Type-I excitation model

with reasonable accuracy [2]. The exponential constants P and Q can be determined from some recommended specifications [1], i.e. $S_E$ can be specified at $E_{fdmax}$ and $0.75E_{fdmax}$ ( = $E_{fdq}$), which may be denoted as $S_{Em}$ and $S_{Eq}$ respectively. From Figure 4,

$$S_{Em} = (A - B) / B$$

$$S_{Eq} = (E - F) / E$$

$$S_{Ef\backslash} = (C - D) / D$$

Taking $E_{fd}$ as the base

$$E_{fdmax} = B / D \quad p.u., \quad i.e. \; B = D \times E_{fdmax}$$

and, $B / F = 4 / 3 = (D \times E_{fdmax}) / F$, which implies that $F = 0.75 \times D \times E_{fdmax}$.

According to the recommendation in [1],

$$S_{Em} = P \times \exp(Q \times E_{fdmax})$$

$$S_{Eq} = P \times \exp(Q \times E_{fdq})$$

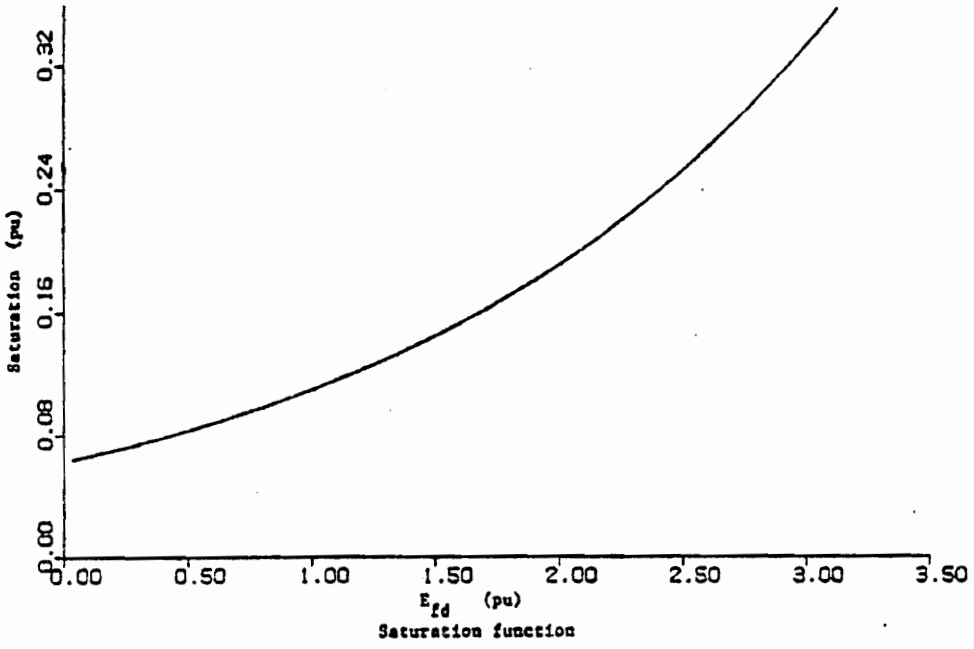
Solving for P and Q,

$$P = S_{Eq}^4 / S_{Em}^3 \tag{2.3}$$

$$Q = (4 / E_{fdmax}) \times \ln(S_{Em} / S_{Eq}) \tag{2.4}$$

while, due to the following restriction [1],

$$V_{Rmax} = (K_e + S_{Em}) \times E_{fdmax}$$



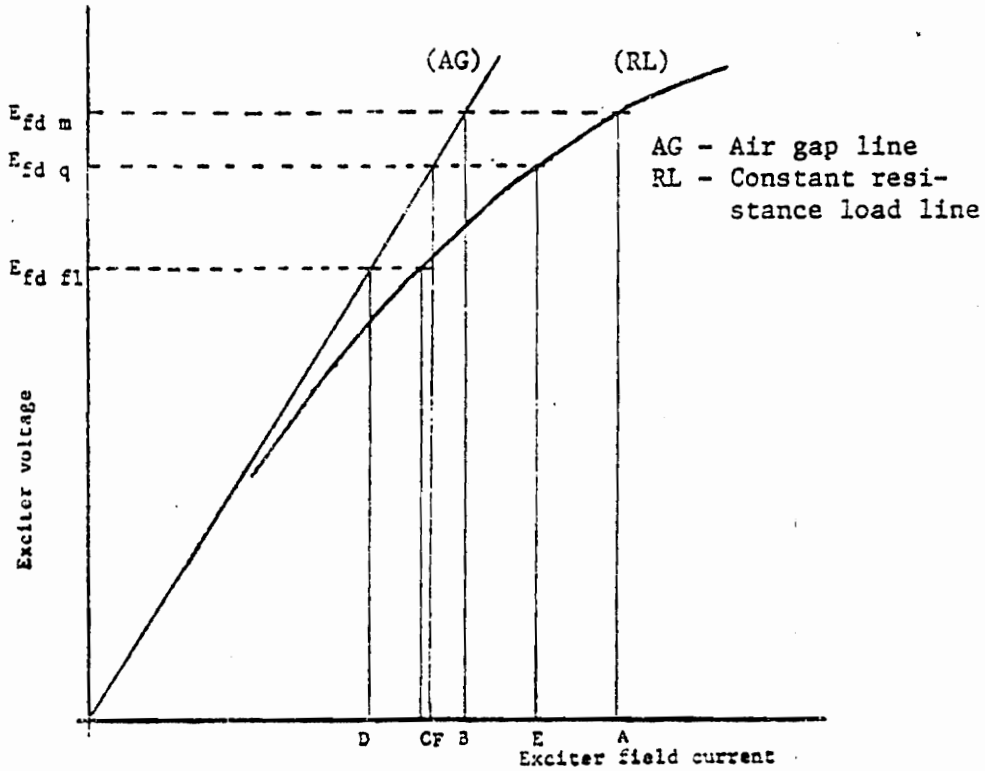




Figure 4. Saturation function and exciter saturation

we get

$$Q = (4(K_e + S_{Em}) / V_{Rmax}) \times \ln(S_{Em} / S_{Eq})$$

In this work, a rotating rectifier with static voltage regulator is used. Table 1 lists the special excitation system constants. From Table 1,

$$P = 0.0983$$

$$Q = 0.553$$

## *2.2 Synchronous Machine Model*

In this section, the Park's model for a synchronous machine is developed. The notation used here is consistent with those in [2] and [4], as is the per unit system.

### 2.2.1 Physical Description

A synchronous machine consists of three stator or armature windings, one field winding on the rotor and several additional rotor windings, known as amortisseur or damper windings. A constant current injected into the field winding will generate a magnetic field aligned with the magnetic axis of the field coil. All of the windings are magnetically coupled. As the rotor is turned, the motion of this magnetic field relative to the fixed stator windings induces voltages in these windings; the instantaneous terminal voltage v of any winding is of the form,

Table 1. Typical constants of rotating rectifier exciter with static voltage regulator

| Symbol | Values |
|---|---|
| $T_r$ | 0.00 sec. |
| $T_a$ | 0.02 sec. |
| $T_f$ | 1.00 sec. |
| $T_e$ | 0.80 sec. |
| $K_a$ | 400.00 |
| $K_f$ | 0.03 |
| $K_e$ | 1.00 |
| $V_{Rmax}$ | 7.30 |
| $V_{Rmin}$ | -7.30 |
| $S_{Emax}$ | 0.86 |
| $S_{E0.75max}$ | 0.50 |

$$v = \pm \sum r \times i \pm \sum \dot{\psi} \tag{2.5}$$

where r is the winding resistance, $\psi$ is the flux linkage and i is the current, with the stator current flowing out of the generator terminals taken as positive. The notation $\pm \sum$ indicates the summation of all appropriate terms with due regard to signs. These expressions for the winding voltages are quite complicated because of the variation of $\psi$ with the rotor position [2]. If the stator windings are arranged such that their magnetic axes are separated by 120° in space, the voltages induced in them will be separated by 120° in phase.

The damper windings are shorted coils, some aligned with and some orthogonal to the magnetic axis of the field winding. The primary effects of the amortisseur windings are to damp rotor speed oscillations and to reduce voltage harmonics [6].

### 2.2.2 Park's Model

The synchronous machine is assumed to be "ideal" in the sense of Park's transformation, in which some assumptions are made [5]:

- All inductances are independent of current (saturation is neglected).
- All self and mutual inductances may be represented as constants plus a simple variation of the rotor angle $\theta$ and $2\theta$.
- All distributed windings may be adequately represented as lumped windings. The effects of currents flowing in the iron parts of the rotor or in the damper windings may be represented by the amortisseur winding coils.

A great simplification in the mathematical description of the synchronous machine is obtained if Park's transformation of variables is performed. It defines a new set of stator variables such as currents, voltages, and flux linkages in terms of the actual winding variables. The new quantities are obtained from the projection of the actual variables on three axes; one along the direct axis of the rotor field winding, called direct axis (d-axis), the second, lagging the d-axis by 90°, along the neutral axis of the main field winding, called quadrature axis (q-axis), and the third one on a stationary axis.

For convenience, the axis of phase A is chosen to be the reference position. Now we define the d-axis of the rotor at some instant of time to be at an angle $\theta$ (radians) with respect to the fixed reference position, as shown in Figure 5 [6].

The effect of Park's transformation is simply to transfer stator electrical variables from phase A, B, C to a coordinate system fixed on the rotor. A multiplier is used to simplify the numerical calculations. By definition

$$X_{0dq} = P(\theta)\, X_{abc} \tag{2.6}$$

where **X** represents voltages, currents or fluxes, and

$$\boldsymbol{X_{0dq}} = \begin{bmatrix} X_0 \\ X_d \\ X_q \end{bmatrix}, \qquad \boldsymbol{X_{abc}} = \begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix}$$

where:

$X_0$ = stationary term representing unbalanced components

$X_d$ = "projection" of $\overline{X}_{abc}$ along the d-axis

$X_q$ = "projection" of $\overline{X}_{abc}$ along the q-axis

and where the transformation matrix $\boldsymbol{P}(\theta)$ is defined as:

$$\boldsymbol{P}(\theta) = \frac{2}{3}\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \cos(\theta) & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ \sin(\theta) & \sin(\theta - 2\pi/3) & \sin(\theta + 2\pi/3) \end{bmatrix}$$

The main field winding flux is along the direction of the d-axis of the rotor. It produces an EMF that lags this flux by 90°. Therefore, the machine EMF E (the stator air gap rms voltage) is primarily along the q-axis. For generator action the phasor $\overline{E}$ should be leading the phasor $\overline{V}$ (machine terminal voltage). The angle between $\overline{E}$ and $\overline{V}$ is the machine torque angle $\delta$ if the phasor $\overline{V}$ is in the direction of the reference phase.

At time $t = 0$, the phasor $\overline{V}$ is located at the axis of phase A. The q-axis is located at an angle $\delta$, and the d-axis is located at

$$\theta_0 = \delta + \frac{\pi}{2}$$

At any given time $t > 0$, the rotor makes an angle $\omega t$ with $\theta_0$, therefore the d-axis is located at

$$\theta(t) = \omega t + \delta + \frac{\pi}{2} \qquad (rad.) \tag{2.7}$$

where $\omega( = \omega_s + \Delta\omega)$ is the angular frequency in rad./sec., $\omega_s$ is the rated (synchronous) angular frequency in rad./sec., $\Delta\omega$ is the rotor speed oscillation about synchronous speed and $\delta$ is the synchronous torque angle in electrical radians [2].

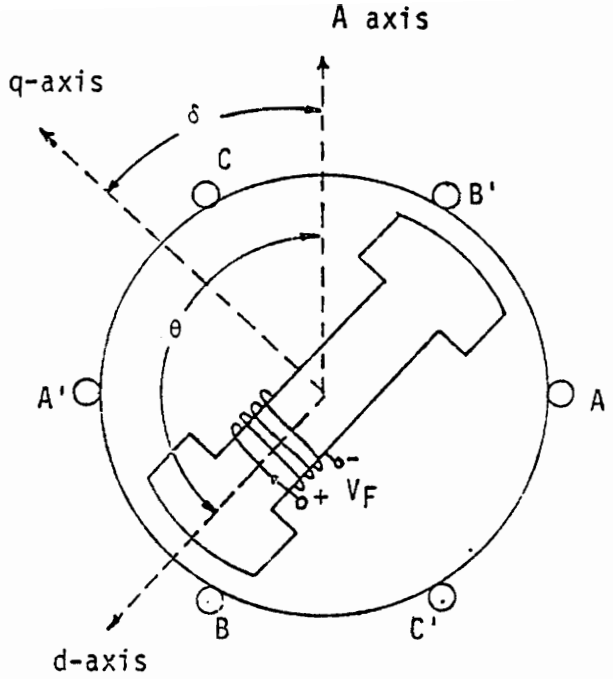



**Figure 5. Pictorial representation of the synchronous machine**

## 2.2.3 Flux Linkage Equations

From Figure 5, we know that a synchronous machine consists of six mutually coupled coils. These are the three phase windings, the field winding, and the two equivalent damper windings. We can write the flux linkage equation for these six circuits as:

$$\begin{bmatrix} \psi_a \\ \psi_b \\ \psi_c \\ \psi_F \\ \psi_D \\ \psi_Q \end{bmatrix} = \begin{bmatrix} L_{aa} & L_{ab} & L_{ac} & L_{aF} & L_{aD} & L_{aQ} \\ L_{ba} & L_{bb} & L_{bc} & L_{bF} & L_{bD} & L_{bQ} \\ L_{ca} & L_{cb} & L_{cc} & L_{cF} & L_{cD} & L_{cQ} \\ L_{Fa} & L_{Fb} & L_{Fc} & L_{FF} & L_{FD} & L_{FQ} \\ L_{Da} & L_{Db} & L_{Dc} & L_{DF} & L_{DD} & L_{DQ} \\ L_{Qa} & L_{Qb} & L_{Qc} & L_{QF} & L_{QD} & L_{QQ} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \\ i_F \\ i_D \\ i_Q \end{bmatrix} \tag{2.8}$$

where

$L_{ij}$ = *self inductance when* $i = j$ and

$L_{ij}$ = *mutual inductance when* $i \neq j$

and where $L_{ij} = L_{ji}$ in all cases. Note that the subscript convention in eq.(2.8), where the lower case subscript is used for stator quantities and the upper case subscript for rotor quantities. These inductances may be described as follows.

### *2.2.3.1 Stator Inductances*

The phase-winding self-inductances can be expressed as a Fourier series of the rotor angle. For example, the inductance $L_{aa}$ is

$$L_{aa} = L_s + L_m ( \cos 2\theta + a_4 \cos 4\theta + \cdots )$$

Since all terms beyond the second harmonic in this series may be neglected, and all windings are assumed to be symmetric about either d or q axis, we get:

$$\begin{aligned} L_{aa} &= L_s + L_m \cos 2\theta \\ L_{bb} &= L_s + L_m \cos 2(\theta - 2\pi/3) \\ L_{cc} &= L_s + L_m \cos 2(\theta + 2\pi/3) \end{aligned} \tag{2.9}$$

where $L_s > L_m$ and both $L_s$ and $L_m$ are constants.

The stator mutual inductances are also functions of $\theta$ and are symmetric, given approximately by:

$$\begin{aligned} L_{ab} &= L_{ba} = -M_s - L_m \cos 2(\theta + \pi/6) \\ L_{bc} &= L_{cb} = -M_s - L_m \cos 2(\theta - \pi/2) \\ L_{ca} &= L_{ac} = -M_s - L_m \cos 2(\theta + 5\pi/6) \end{aligned} \tag{2.10}$$

where $|M_s| > L_m$. Note that the sign of the mutual inductance term depends on assumed current directions and coil orientations. In addition, the second harmonic terms in the stator self and mutual inductances are assumed to be equal.

### *2.2.3.2 Rotor Inductances*

Since saturation and slotting effects are neglected, all rotor self inductances are constants, given by:

$$L_{FF} = L_F \qquad L_{DD} = L_D \qquad L_{QQ} = L_Q \tag{2.11}$$

The rotor mutual inductances are also constants but do not vary with $\theta$. Since the d and q axis windings are with 90° displacement, the coefficient of the magnetic coupling between d and q axis is zero, i.e.

$$L_{FD} = L_{DF} = M_R \qquad L_{FQ} = L_{QF} = 0 \qquad L_{DQ} = L_{QD} = 0 \tag{2.12}$$

### *2.2.3.3 Stator-to-rotor Mutual Inductances*

All of the stator-to-rotor mutual inductances are functions of the rotor angle $\theta$. They can be written as a Fourier series of odd harmonic terms, i.e.

$$M_{aF} = M_F(\cos\theta + b_3\cos 3\theta + \cdots)$$

Remember the assumption that all terms above the second harmonic are ignored. Therefore, the mutual inductances between the phase windings and the field winding may be written as:

$$\begin{aligned} L_{aF} &= L_{Fa} = M_F\cos(\theta) \\ L_{bF} &= L_{Fb} = M_F\cos(\theta - 2\pi/3) \\ L_{cF} &= L_{Fc} = M_F\cos(\theta + 2\pi/3) \end{aligned} \tag{2.13}$$

Similarly, the mutual inductances between the phase and damper windings are:

$$\begin{aligned} L_{aD} &= L_{Da} = M_D\cos(\theta) \\ L_{bD} &= L_{Db} = M_D\cos(\theta - 2\pi/3) \\ L_{cD} &= L_{Dc} = M_D\cos(\theta + 2\pi/3) \end{aligned} \tag{2.14}$$

and

$$\begin{aligned} L_{aQ} &= L_{Qa} = M_Q\sin(\theta) \\ L_{bQ} &= L_{Qb} = M_Q\sin(\theta - 2\pi/3) \\ L_{cQ} &= L_{Qc} = M_Q\sin(\theta + 2\pi/3) \end{aligned} \tag{2.15}$$

Again, the sign of the mutual inductance term depends on assumed current directions and coil orientations.

### *2.2.3.4 Transformation of Stator Fluxes*

We now apply Park's transformation to simplify the stator terms of the flux equation (2.8), which requires that both sides of the equation are multiplied by

$$\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}$$

where $P(\theta)$ is Park's transform matrix and **I** is a 3 by 3 unit matrix.

Thus

$$\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \psi_{abc} \\ \psi_{FDQ} \end{bmatrix} = \begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} L_{aa} & L_{aR} \\ L_{Ra} & L_{RR} \end{bmatrix}\begin{bmatrix} P^{-1}(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} i_{abc} \\ i_{FDQ} \end{bmatrix}$$

where

$L_{aa}$ = stator-to-stator inductances

$L_{aR}$, $L_{Ra}$ = stator-to-rotor inductances

$L_{RR}$ = rotor-to-rotor inductances

Performing the matrix operation, we have:

$$\begin{bmatrix} \psi_0 \\ \psi_d \\ \psi_q \\ \psi_F \\ \psi_D \\ \psi_Q \end{bmatrix} = \begin{bmatrix} L_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & L_d & 0 & KM_F & KM_D & 0 \\ 0 & 0 & L_q & 0 & 0 & KM_Q \\ 0 & KM_F & 0 & L_F & M_R & 0 \\ 0 & KM_D & 0 & M_R & L_D & 0 \\ 0 & 0 & KM_Q & 0 & 0 & L_Q \end{bmatrix}\begin{bmatrix} i_0 \\ i_d \\ i_q \\ i_F \\ i_D \\ i_Q \end{bmatrix} \tag{2.16}$$

where

$$L_d = L_s + M_s + 3/2L_m$$
$$L_q = L_s + M_s - 3/2L_m$$
$$L_0 = L_s - 2M_s$$
$$K = \sqrt{3/2}$$

and where $\psi_d$ is the flux linkage in a circuit moving with the rotor and centered on the d-axis. Similarly, $\psi_q$ is centered on the q-axis.

## 2.2.4 Voltage Equations

The machine voltage equations are in the form of eq.(2.5). Schematically, the circuits are shown in Figure 6. For the conditions indicated in Figure 6, rewrite eq.(2.5) in matrix form as follows:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \\ -V_F \\ 0 \\ 0 \end{bmatrix} = - \begin{bmatrix} r_a & 0 & 0 & 0 & 0 & 0 \\ 0 & r_b & 0 & 0 & 0 & 0 \\ 0 & 0 & r_c & 0 & 0 & 0 \\ 0 & 0 & 0 & r_F & 0 & 0 \\ 0 & 0 & 0 & 0 & r_D & 0 \\ 0 & 0 & 0 & 0 & 0 & r_Q \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \\ i_F \\ i_D \\ i_Q \end{bmatrix} - \begin{bmatrix} \dot{\psi}_a \\ \dot{\psi}_b \\ \dot{\psi}_c \\ \dot{\psi}_F \\ \dot{\psi}_D \\ \dot{\psi}_Q \end{bmatrix} + \begin{bmatrix} V_n \\ 0 \end{bmatrix} \qquad (2.17)$$

or in partitioned form:

$$\begin{bmatrix} V_{abc} \\ V_{FDQ} \end{bmatrix} = - \begin{bmatrix} R_{abc} & 0 \\ 0 & R_{FDQ} \end{bmatrix} \begin{bmatrix} i_{abc} \\ i_{FDQ} \end{bmatrix} - \begin{bmatrix} \dot{\psi}_{abc} \\ \dot{\psi}_{FDQ} \end{bmatrix} + \begin{bmatrix} V_n \\ 0 \end{bmatrix} \qquad (2.18)$$

where $\overline{V}_n$ is the vector of the neutral voltages.

Similar to transforming the stator fluxes, applying Park's transformation to the machine voltage equations (2.18), gives:

$$\begin{aligned}\begin{bmatrix} V_{0dq} \\ V_{FDQ} \end{bmatrix} &= \begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} V_{abc} \\ V_{FDQ} \end{bmatrix} \\ &= -\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} r_{abc} & 0 \\ 0 & r_{FDQ} \end{bmatrix}\begin{bmatrix} P^{-1}(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} i_{abc} \\ i_{FDQ} \end{bmatrix} \\ &\quad -\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \dot{\psi}_{abc} \\ \dot{\psi}_{FDQ} \end{bmatrix} + \begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} V_n \\ 0 \end{bmatrix}\end{aligned} \tag{2.19}$$

If the machine is balanced, $r_a = r_b = r_c = r$, as is usually the case, we may compute the resistance voltage drop terms as:

$$\begin{aligned}&\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} r_{abc} & 0 \\ 0 & r_{FDQ} \end{bmatrix}\begin{bmatrix} P^{-1}(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} P(\theta) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} i_{abc} \\ i_{FDQ} \end{bmatrix} \\ &= \begin{bmatrix} P(\theta)r_{abc}P^{-1}(\theta) & 0 \\ 0 & r_{FDQ} \end{bmatrix}\begin{bmatrix} i_{0dq} \\ i_{FDQ} \end{bmatrix} \\ &= \begin{bmatrix} r_{abc} & 0 \\ 0 & r_{FDQ} \end{bmatrix}\begin{bmatrix} i_{abc} \\ i_{FDQ} \end{bmatrix}\end{aligned} \tag{2.20}$$

where

$$\begin{bmatrix} r_{abc} & 0 \\ 0 & r_{FDQ} \end{bmatrix} = \begin{bmatrix} r & 0 & 0 & 0 & 0 & 0 \\ 0 & r & 0 & 0 & 0 & 0 \\ 0 & 0 & r & 0 & 0 & 0 \\ 0 & 0 & 0 & r_F & 0 & 0 \\ 0 & 0 & 0 & 0 & r_F & 0 \\ 0 & 0 & 0 & 0 & 0 & r_Q \end{bmatrix}$$

The second term on the right side of eq. (2.19) can be transformed by recalling that $\psi_{0dq} = P(\theta)\psi_{abc}$, from which we compute

$$\dot{\psi}_{0dq} = P(\theta)\dot{\psi}_{abc} + \dot{P}(\theta)\psi_{abc}$$

which is rearranged to obtain:

$$P(\theta)\dot{\psi}_{abc} = \dot{\psi}_{0dq} - \dot{P}(\theta)\psi_{abc} = \dot{\psi}_{0dq} - \dot{P}(\theta)P^{-1}(\theta)\psi_{0dq} \tag{2.21}$$

Matrix multiplication shows that

$$\dot{P}(\theta)P^{-1}(\theta) = \omega_s \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Next, the third term on the right side of eq. (2.19) is written as:

$$\begin{bmatrix} \boldsymbol{P(\theta)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{V_n} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n_{0dq}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} -3r_n i_0 - 3L_n i_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.22}$$

Note that the voltage drop occurs only in the zero sequence. While, under balanced conditions, the zero sequence voltage vanishes. Finally, the resulting set of equations may be written without the zero sequence equation as:

$$\begin{aligned} V_d &= -ri_d - \dot{\psi}_d - \omega\psi_q \\ V_q &= -ri_q - \dot{\psi}_q - \omega\psi_d \\ V_F &= -r_F i_F - \dot{\psi}_F \\ V_D &= -r_D i_D - \dot{\psi}_D = 0 \\ V_Q &= -r_Q i_Q - \dot{\psi}_Q = 0 \end{aligned} \tag{2.23}$$

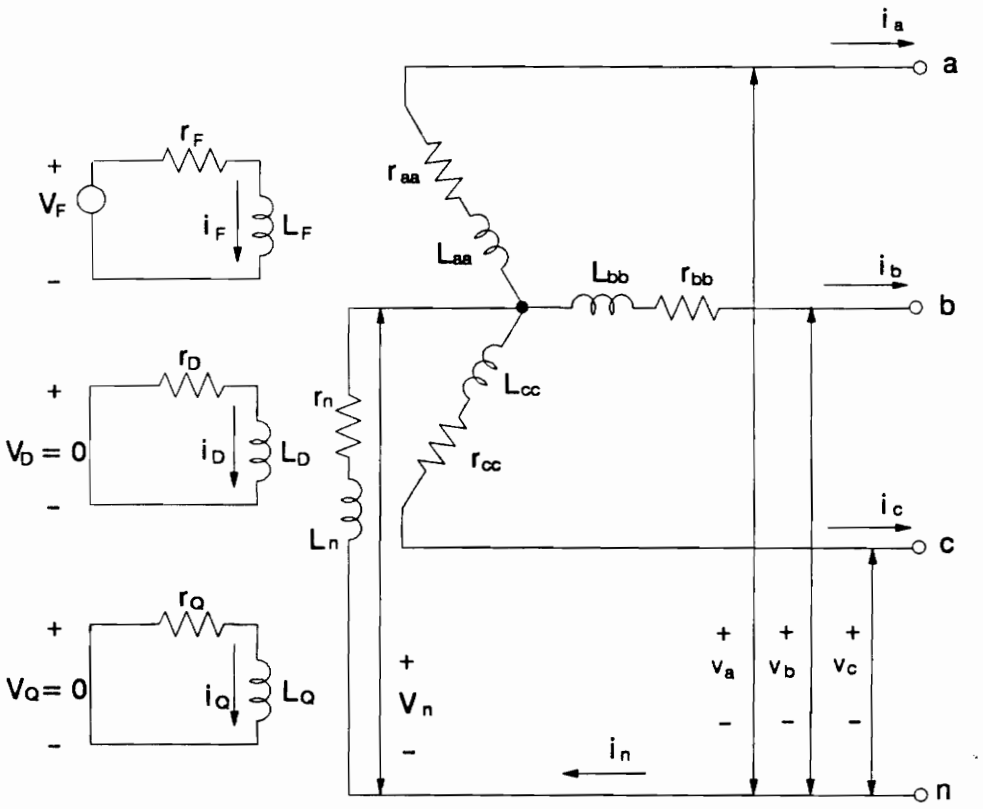



**Figure 6. Schematic diagram of a synchronous machine**

These equations are coupled with those for the mutual flux linkages. The d-axis and q-axis mutual terms may be calculated from the d and q axes terms respectively, in the flux equation (2.16).

Rewrite each self inductance as the sum of a magnetizing inductance and a leakage inductance:

$$\begin{aligned} L_d &= L_{md} + L_l \\ L_q &= L_{mq} + L_l \\ L_F &= L_{mF} + l_F \\ L_D &= L_{mD} + l_D \\ L_Q &= L_{mQ} + l_Q \end{aligned} \tag{2.24}$$

Substitubing eq. (2.24) into (2.16) yields:

$$\begin{aligned} \psi_d &= (L_{md} + L_l)i_d + KM_F i_F + KM_D i_D \\ \psi_q &= (L_{mq} + L_l)i_q + KM_Q i_Q \\ \psi_F &= KM_F i_d + (L_{mF} + l_F)i_F + M_R i_D \\ \psi_D &= KM_D i_d + M_R i_F + (L_{mD} + l_D)i_D \\ \psi_Q &= KM_Q i_q + (L_{mQ} + l_Q)i_Q \end{aligned} \tag{2.25}$$

Therefore, the portion of the armature flux due to $i_d$ which links the other d-axis circuits is $L_{md}i_d$. Similarly, the flux due to $i_q$ is $L_{mq}i_q$. Since the mutual flux linkages should be equal, setting $i_F = i_D = 0$ and $i_d = 1$ p.u., and exciting each of the other d-axis circuits in turn, gives:

$$L_{ad} \equiv L_{md} = L_{mF} = L_{mD} = KM_F = KM_D = M_R \quad (p.u.) \tag{2.26}$$

After subtracting the per unit leakage flux from the flux linking, the remaining part is the mutual flux, i.e.:

$$\psi_{ad} \equiv \psi_d - L_l i_d = \psi_F - l_F i_F = \psi_D - l_D i_D \quad (p.u.) \tag{2.27}$$

Also, setting $i_Q = 0$ and $i_q = 1$ p.u., gives:

$$L_{aq} \equiv L_{mq} = L_{mQ} = KM_Q \quad (p.u.) \tag{2.28}$$

and the q-axis mutual flux is

$$\psi_{aq} \equiv \psi_q - L_l i_q = \psi_Q - l_Q i_Q \quad (p.u.) \tag{2.29}$$

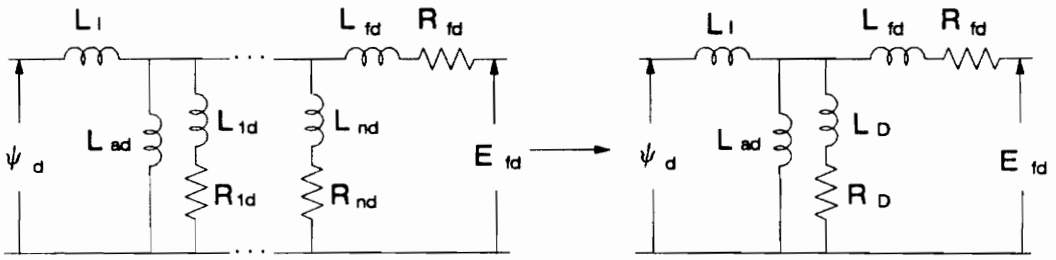
From the voltage and the mutual flux equations, the equivalent circuit for the synchronous machine in the Park's reference frame can be drawn [2, 6]. The equivalent circuit and the simulation model will be discussed in detail in the next two sections.

### 2.2.5 Equivalent Circuits

The d and q axes equivalent circuits which represent the synchronous machine are shown in Figure 7, where the elements $L_l$ represent the armature leakage inductances, $L_{ad}$ and $L_{aq}$ are the stator-rotor mutual inductances, $R_{fd}$ and $L_{fd}$ are the field resistance and leakage inductance respectively, $R_{nd}$, $L_{nd}$, $R_{nq}$ and $L_{nq}$ are damper resistances and leakage inductances (the equivalent amortisseur resistance and leakage inductance are written as $R_D$, $L_D$, $R_Q$ and $L_Q$).

Since the seventies, some studies reported in the IEEE Transaction on PAS "have shown that the classical model does not provide enough flexibility for use in system studies in those special cases when the behavior of the synchronous machine and its controls must be represented very accurately. Thus, a new model is needed for accurate simulation of solid rotor turbine-generators during transients introduced by the power system and abnormal operation." [4] In this research, a set of d-axis and q-axis equivalent circuits for turbine-generators with solid iron rotor is adopted. This is referred to as the Jackson-Winchester model [4].

d Axis Equivalent Circuit




q Axis Equivalent Circuit

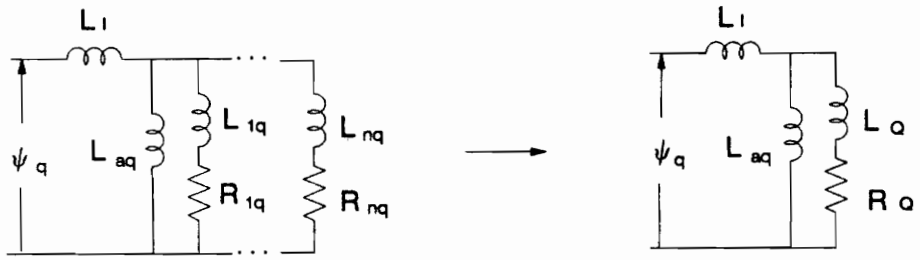



**Figure 7. Equivalent circuits of synchronous machine**

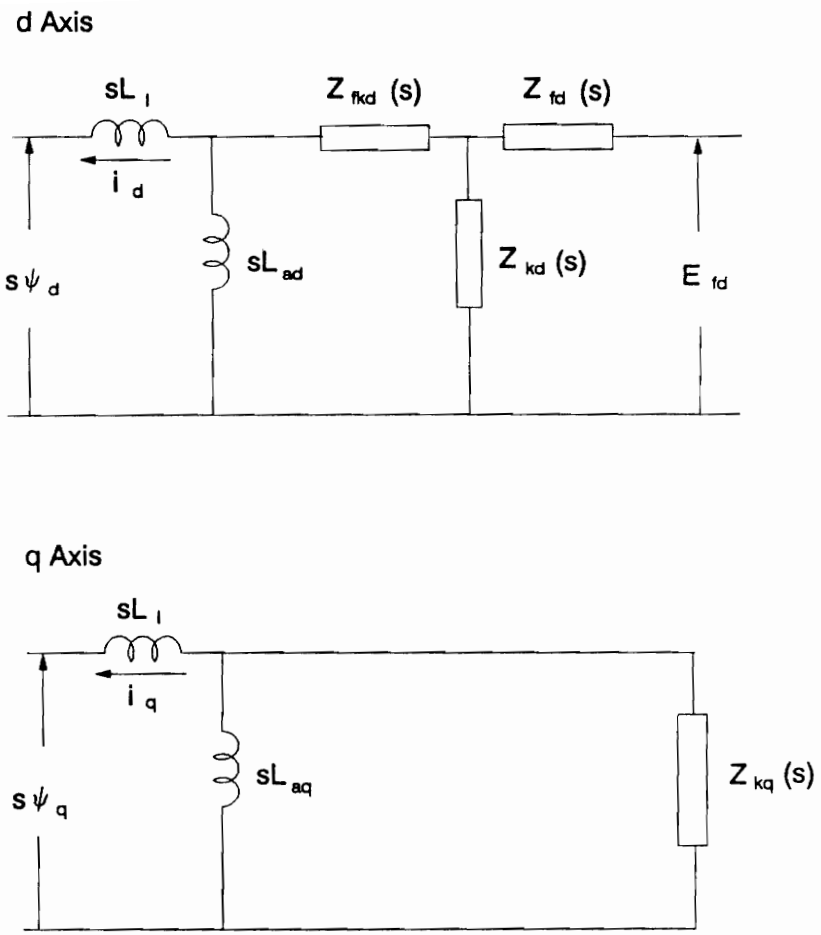



**Figure 8. Structure of equivalent circuits of Solid Iron Rotor Model**

To create a time domain simulation model, it is convenient to start with the analysis for a frequency domain machine model shown in Figure 8, where $sL_l$ represents the impedance of the armature leakage inductance as a function of the Laplace operator **s**. In a similar way, $sL_{ad}$ and $sL_{aq}$ represent the impedance of the armature-rotor mutual inductance; $Z_{fkd}(s)$ represents a mutual impedance between the field and rotor body; $Z_{fd}(s)$ is the field impedance and $Z_{kd}(s)$ and $Z_{kq}(s)$ describe the impedances of the iron and any non-excited windings on the rotor [4]. The results of a study have stated that based on the analysis of plots of the discrete-elements $Z_{fkd}(s)$, $Z_{fd}(s)$, $Z_{kd}(s)$ and $Z_{kq}(s)$, it is evdent that a discrete-element model for the solid iron rotor generator referred to as the Solid Iron Rotor Model (SIRM), which is shown in Figure 9, would be adequate for most of the studies that need an accurate machine model [4].

As the values of the resistive and inductive elements of the SIRM are known, the impedances $Z_{fd}(s)$, $Z_{kd}(s)$ and $Z_{kq}(s)$ may be written as follows:

$$Z_{fd}(s) = R_{fd} + \frac{s}{\omega_0} L_{fd} \tag{2.30}$$

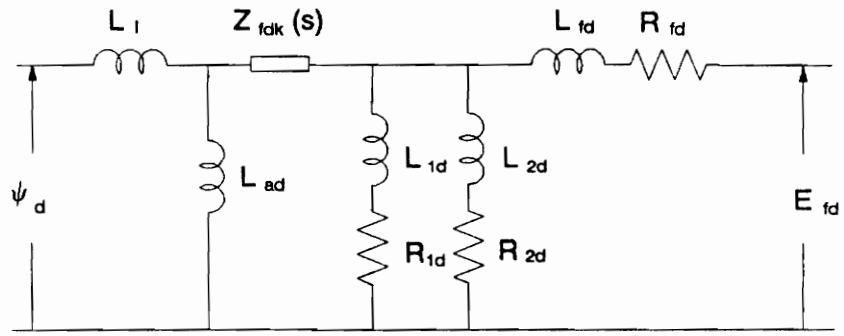
$$Z_{kd}(s) = \frac{(R_{1d} + \frac{s}{\omega_0} L_{1d})(R_{2d} + \frac{s}{\omega_0} L_{2d})}{R_{1d} + R_{2d} + \frac{s}{\omega_0} (L_{1d} + L_{2d})} \tag{2.31}$$

$$Z_{kq}(s) = \frac{(R_{1q} + \frac{s}{\omega_0} L_{1q})(R_{2q} + \frac{s}{\omega_0} L_{2q})}{R_{1q} + R_{2q} + \frac{s}{\omega_0} (L_{1q} + L_{2q})} \tag{2.32}$$

where, $\omega_0 = 377$ rad./sec..

The corresponding inductances and resistances of the SIRM are listed in Table 2.

d Axis




q Axis

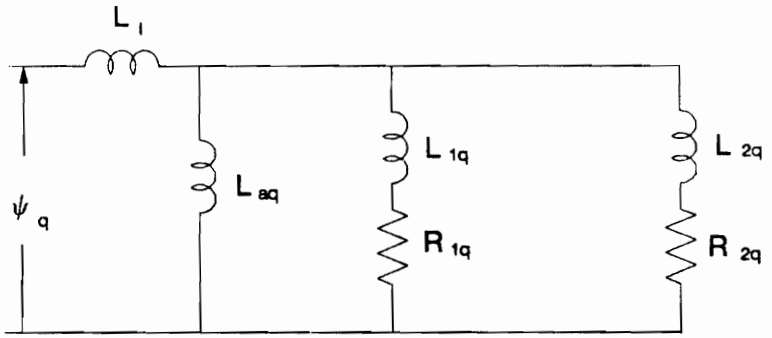



**Figure 9. Solid Iron Rotor Model equivalent circuits**

## 2.2.6 Mathematical Model

In order to derive the simulation model, it is necessary to convert the parts of the equivalent circuits to the right of $L_{ad}$ and $L_{aq}$ in Figure 9 to Thevenin equivalents, which are shown in Figure 10. The Thevenin impedance in the d-axis is $Z_{eq}(s)$ and the Thevenin voltage is $A(s)E_{fd}$, where

$$Z_{eq}(s) = \frac{Z_{fd}(s) \times Z_{kd}(s)}{Z_{fd}(s) + Z_{kd}(s)} + Z_{fdk}(s) \tag{2.33}$$

$$A(s) = \frac{Z_{kd}(s)}{Z_{fd}(s) + Z_{kd}(s)} \tag{2.34}$$

To determine the field current, another relation is required:

$$I_{fd} = A(s) \times i_{eq} + B(s) \times E_{fd} \tag{2.35}$$

where

$$B(s) = \frac{1}{Z_{fd}(s) + Z_{kd}(s)} \tag{2.36}$$

If $Z_{eq}(s)$ and $Z_{kq}(s)$ can be expressed in the following form:

$$Z_{eq}(s) = R_{eq0} + \frac{s}{\omega_0} L_{eq}(s) \tag{2.37}$$

$$Z_{kq}(s) = R_{kq0} + \frac{s}{\omega_0} L_{kq}(s) \tag{2.38}$$

then the flux linkages of the machine can be described as follows [4]:

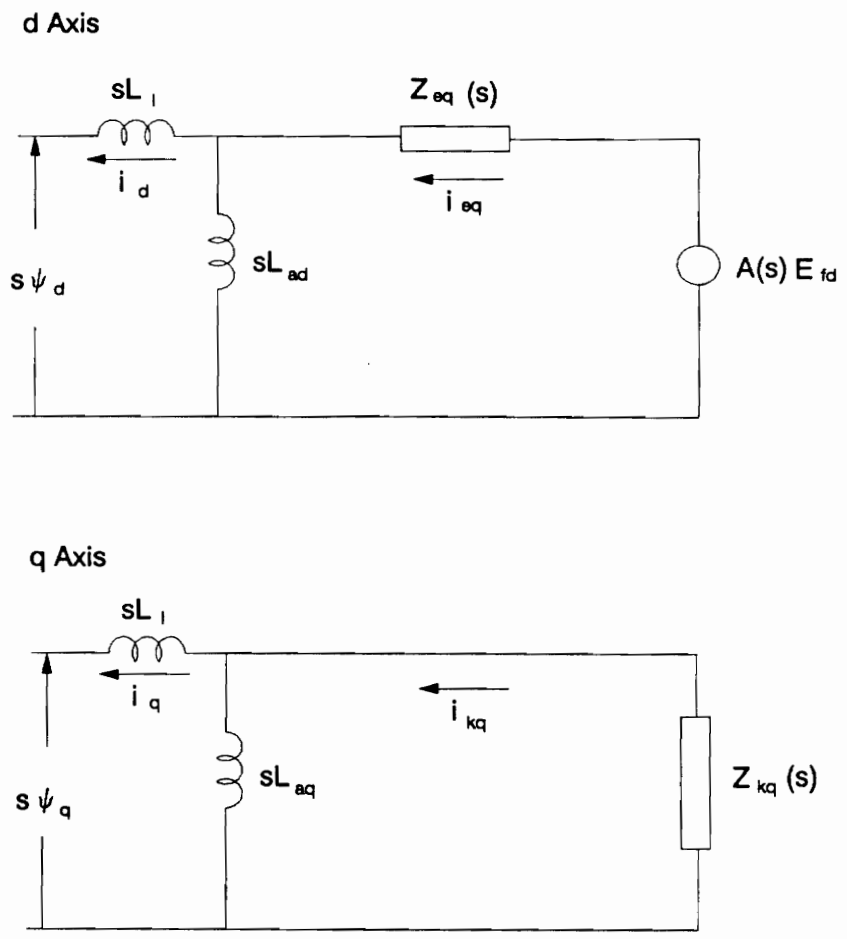



**Figure 10. Solid Iron Rotor Model equivalent circuits in Thevenin equivalent form**

$\underline{d-axis}$

$$\psi_{eq} = \psi_{ad} + i_{eq}L_{eq}(s)$$
$$\psi_{ad} = (i_{eq} - i_d)L_{ad}$$
$$\psi_d = \psi_{ad} - i_d L_l$$
$$\frac{s}{\omega_0}\psi_{eq} = A(s)E_{fd} - i_{eq}R_{eq0}$$
$$e_{fd} = \frac{R_{fd}}{L_{ad}}E_{fd}$$

$\underline{q-axis}$

$$\psi_{kq} = \psi_{aq} + i_{kq}L_{kq}(s)$$
$$\psi_{aq} = (i_{kq} - i_q)L_{aq}$$
$$\psi_q = \psi_{aq} - i_q L_l$$
$$\frac{s}{\omega_0}\psi_{kq} = -i_{kq}R_{kq0}$$

These equations can be described in block diagram form as shown in Figure 11 [4]. The interactions of the loops for the model can be decoupled by suitably alternating the transfer function of the model. The development of such a model for the d and q axes is outlined in Figure 12 and 13, respectively.

Table 3 lists the special three rotor circuit synchronous machine parameters for this research. The relationships between the equivalent circuit inductances and resistances, and the machine parameters (Shown in Figures 9 and 11) are defined in Appendix A [7].

So far, we have established a simulation model for the synchronous machine. The effects of saturation are yet to be studied. Saturation of the machine is often specified in terms of a p.u. saturation function $S_G$, which is defined in terms of the open circuit terminal voltage versus field current characteristic shown in Figure 14.

There are several methods to consider a saturation function, one of which is given in [2], which defines:

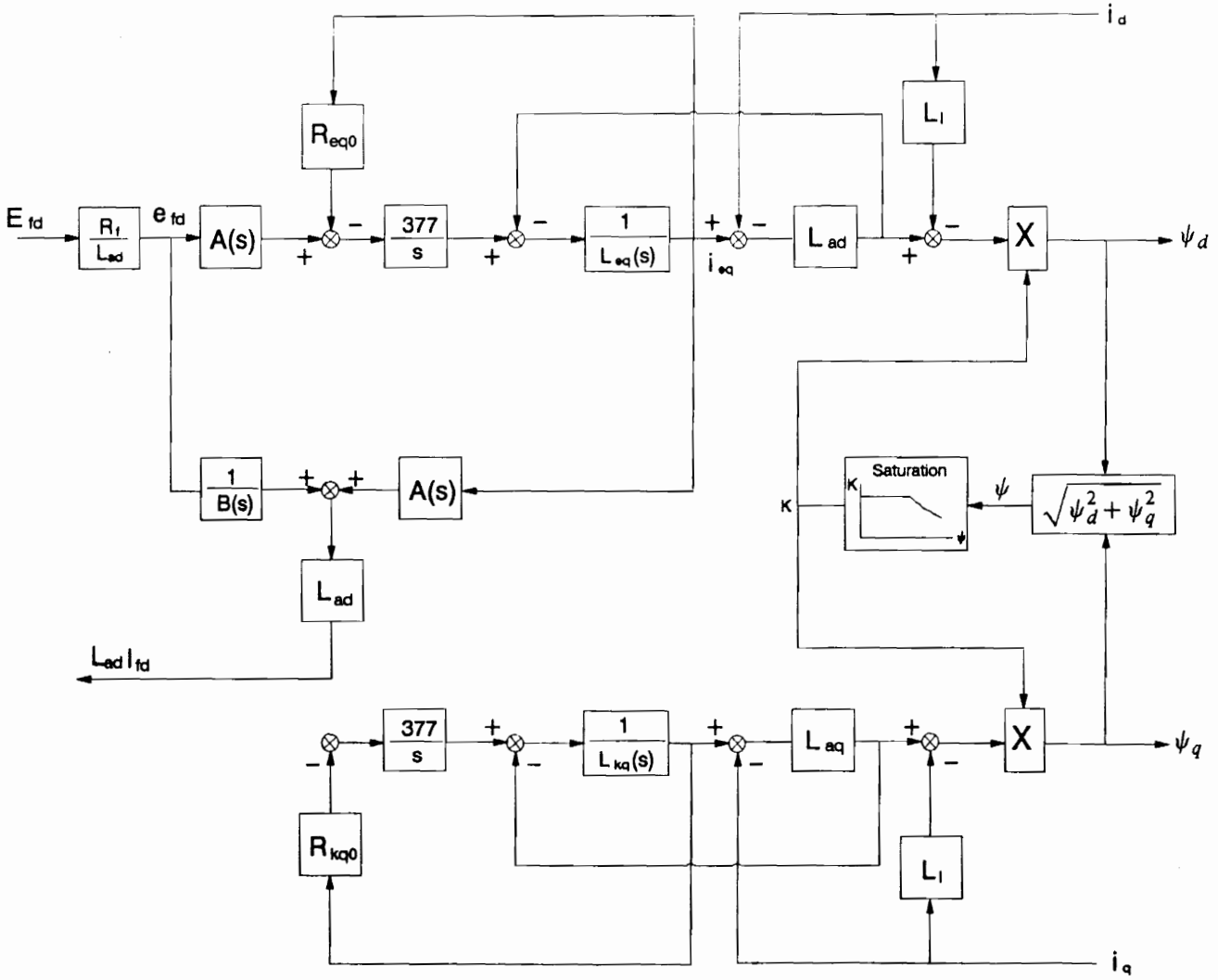



Figure 11. The block diagram for simulation of Solid Iron Rotor Model

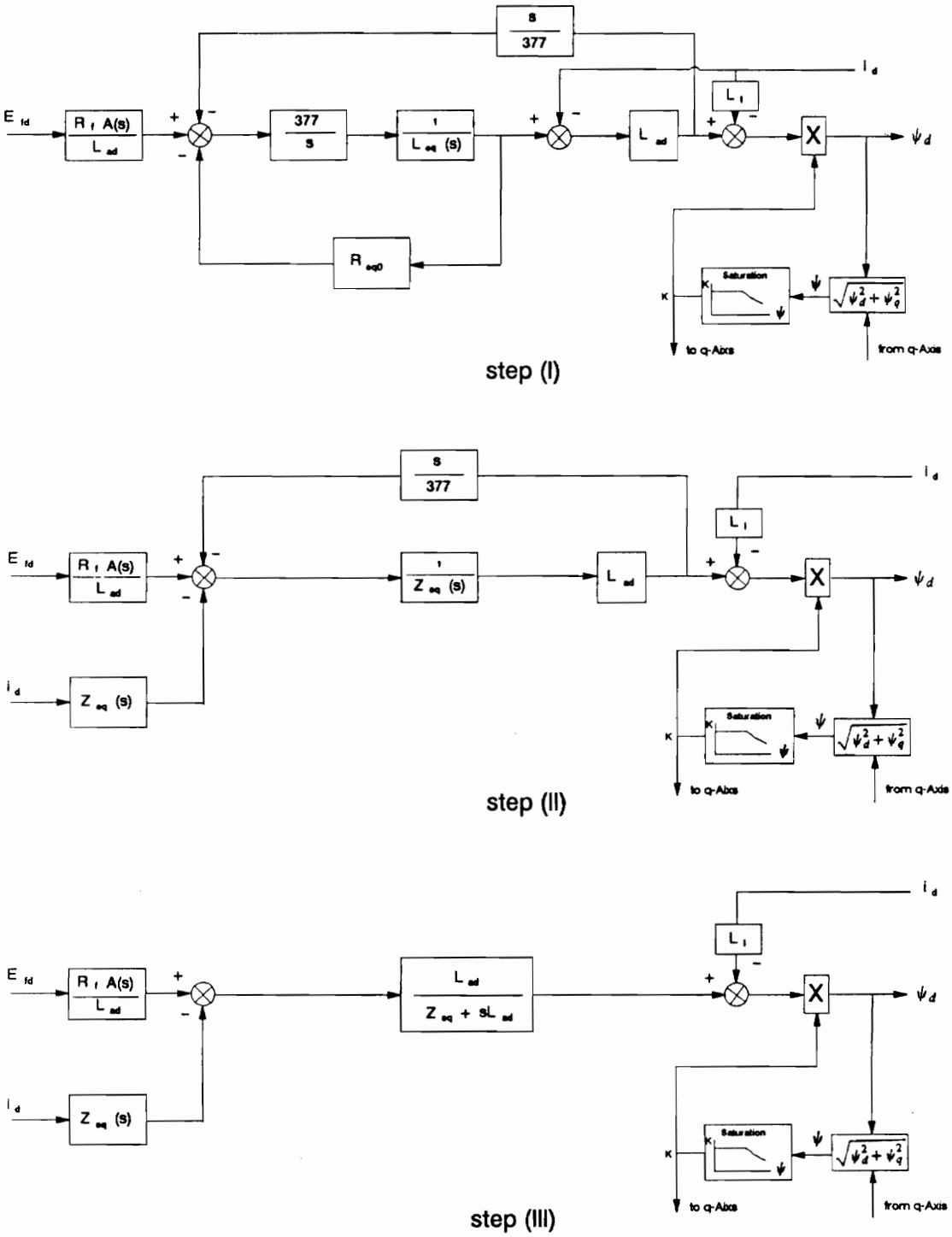



**Figure 12. Step by step derivation of d-axis transfer function**

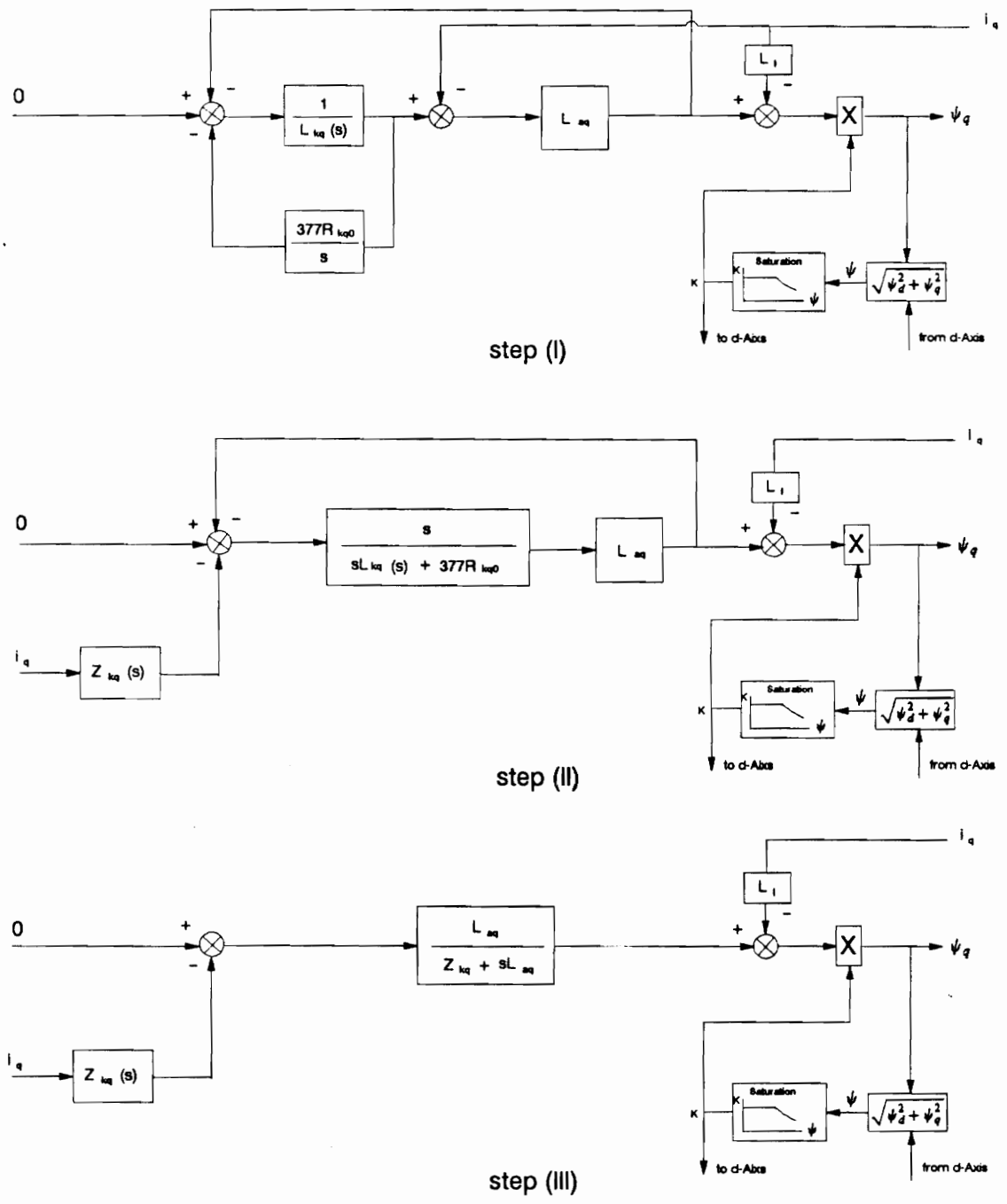



**Figure 13. Step by step derivation of q-axis transfer function**

Table 2. Typical parameters of equivalent circuits of Solid Iron Rotor Motor

| Symbol | | Values (p.u.) |
|---|---|---|
| *d – axis* | $L_l$ | 0.225 |
| | $L_{ad}$ | 2.305 |
| | $L_{fd}$ | 0.225 |
| | $R_{fd}$ | 0.34 |
| | $L_{1d}$ | 0.85 |
| | $R_{1d}$ | 1.05 |
| | $L_{2d}$ | 0.056 |
| | $R_{2d}$ | 1.25 |
| *q – axis* | $L_l$ | 0.225 |
| | $L_{aq}$ | 1.875 |
| | $L_{1q}$ | 0.884 |
| | $R_{1q}$ | 1.61 |
| | $L_{2q}$ | 0.068 |
| | $R_{2q}$ | 4.81 |

**Table 3. Typical parameters of synchronous machine**

| Symbol | | Values |
|---|---|---|
| ***d – axis*** | $T'_{d0}$ | 7.45 sec. |
| | $T''_{d0}$ | 1.00 sec. |
| | $T'''_{d0}$ | 0.08 sec. |
| | $L_d$ | 2.53 p.u. |
| | $L'_d$ | 0.43 p.u. |
| | $L''_d$ | 0.39 p.u. |
| | $L'''_d$ | 0.267 p.u. |
| | $L_l$ | 0.225 p.u. |
| ***q – axis*** | $T'_{q0}$ | 1.176 sec. |
| | $T''_{q0}$ | 0.139 sec. |
| | $L_q$ | 2.1 p.u. |
| | $L'_q$ | 0.826 p.u. |
| | $L''_q$ | 0.286 p.u. |
| | $L_l$ | 0.225 p.u. |

Machine base is 588 MVA and 22 KV.

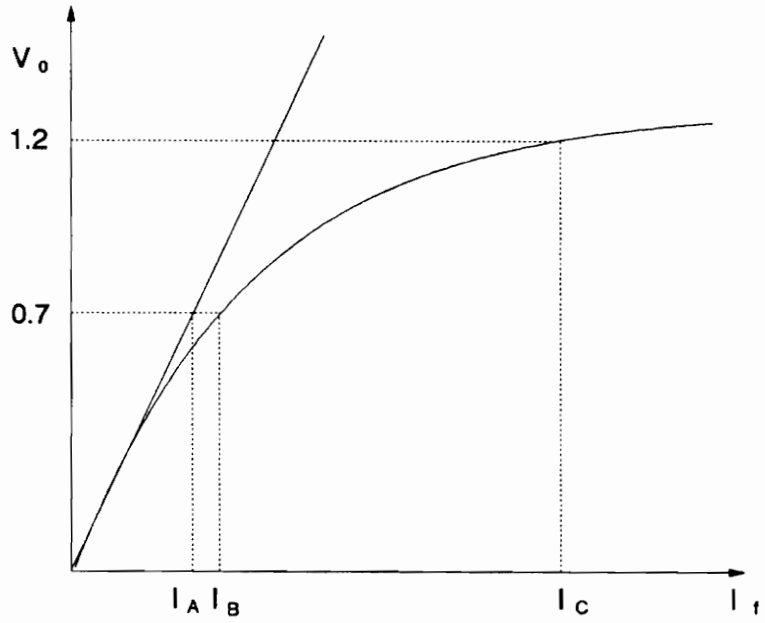



**Figure 14. Construction used for computing saturation**

$$S_G(V_T) = \begin{cases} 0 & V_T < 0.7 p.u. \\ A_G \exp(B_G V_\Delta) & V_T \geq 0.7 p.u. \end{cases} \tag{2.39}$$

where $V_\Delta = V_T - 0.7$, which is the difference between the open circuit terminal voltage and the assumed saturation threshold 0.7 p.u., and

$$A_G = S_{G1.0}^2/(1.2 \times S_{G1.2}) \tag{2.40}$$

$$B_G = 5 \ln(1.2 \times S_{G1.2}/S_{G1.0}) \tag{2.41}$$

where

$$S_{G1.0} = (I_B - I_A)/I_A$$

$$S_{G1.2} = (I_C - 1.2 \times I_A)/(1.2 \times I_A)$$

We note that the function $S_G$ is always positive, and it satisfies the defined values $S_{G1.0}$ and $S_{G1.2}$ at $V_T = 1.0$ and $V_T = 1.2$, respectively. The computed saturation function has the shape shown in Figure 15.

For this research, a typical open circuit saturation curve [8] is considered, which provides the following data:

$$S_{G1.0} = 0.1718 \qquad S_{G1.2} = 0.7028$$

Then we calculate $S_G$, $A_G$ and $B_G$ by using eqs. (2.39), (2.40) and (2.41), and get:

$$A_G = 0.035 \qquad B_G = 7.955$$

$$S_G(V_T) = 0.035 \times \exp(7.955 \times V_\Delta)$$

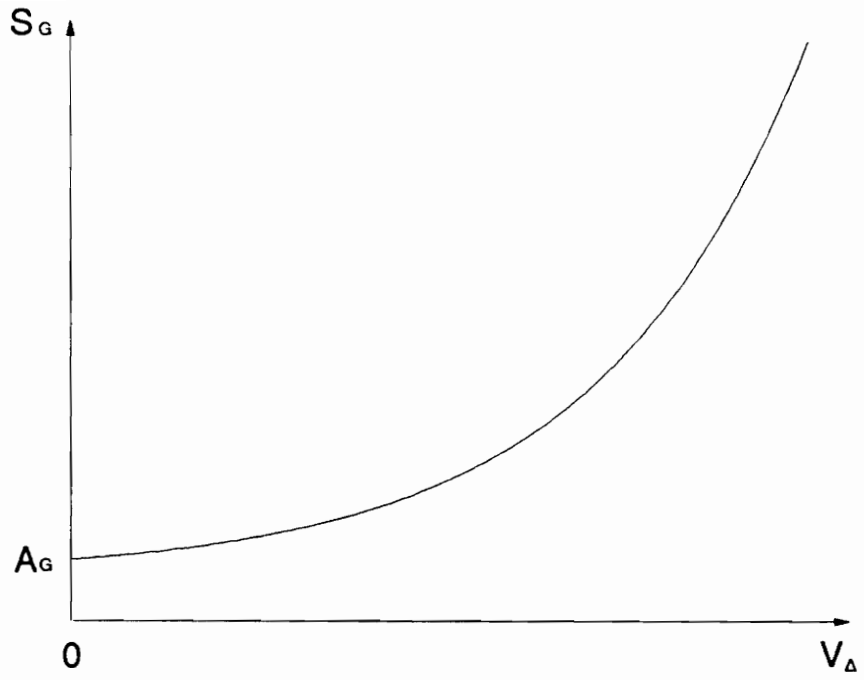



**Figure 15. The approximate saturation function $S_G$**

## 2.3 Governor-turbine Model

As it has exciter models, the Power System Engineering Committee of IEEE has also standardized "general models applicable to most turbine and speed-governing combinations." [9]. The speed-governing and turbine model shown in Figure 16 is used for this thesis. This model may represent either a mechanical-hydraulic speed-governing system or an electro-hydraulic speed-governing system by means of an appropriate selection of parameters. Here, the Westinghouse EH with steam feedback speed-governing model is finally chosen, and the simplest nonreheat turbine model is used. The typical parameters for the model are listed in Table 4.

The transfer functions and nomenclature of the model will be explained in detail as follows. One of the inputs, $\Delta\omega$, is the speed deviation resulting from the mechanical subsystem (i.e. swing equation). The other input $P_0$ (initial mechanical power) is the load reference. This initial value is combined with the increments due to speed deviation to obtain the total power at gate, $P_{GV}$, subject to the time lag, $T_3$, introduced by the servomotor mechanism. K is the total effective speed-governing system gain. There are two limit functions in the model, one introduces the rate limits $\dot{P}_{up}$ and $\dot{P}_{down}$, which limit the rate of change of power imposed by control valve rate limits; the other gives position limits $P_{max}$ and $P_{min}$, which limits power imposed by valve or gate travel. Here, we take

$$\dot{P}_{up} = 0.1 \ p.u./sec. \qquad \dot{P}_{down} = -0.1 \ p.u./sec.$$

and

$$P_{max} = 1.2 \ p.u. \qquad P_{min} = 0.9 \ p.u.$$

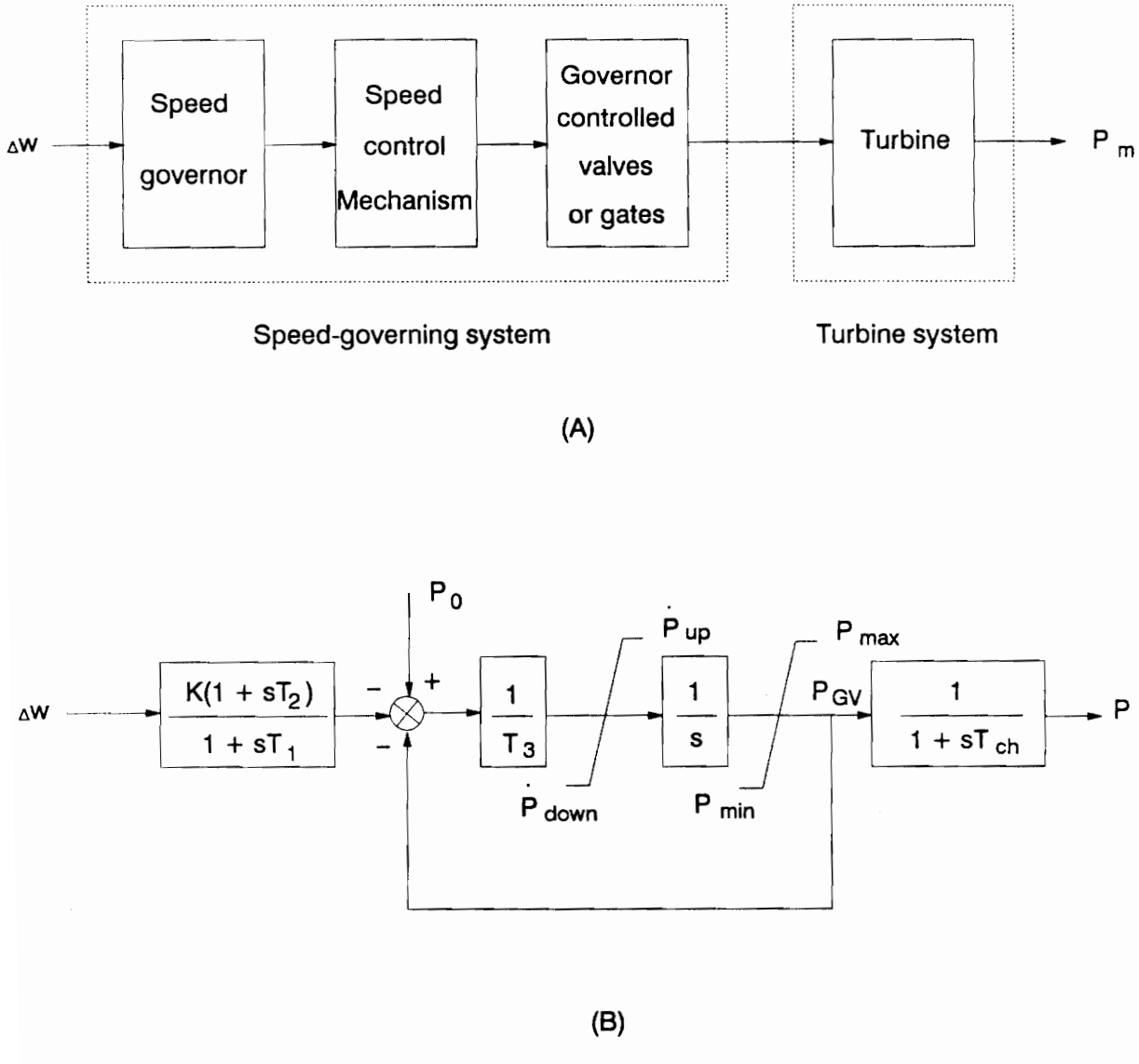



**Figure 16.** **The block diagram of electro-hydraulic speed-governing and steam turbine system** (A) Functional block diagram; (B) Approximate mathematical representation for Westinghouse EH system

**Table 4. Speed-governing system and turbine parameters for Figure 16 (B)**

| System description | Symbol | Values (sec.) |
|---|---|---|
| Westinghouse EH with | $T_1$ | 2.80 |
| steam feedback speed- | $T_2$ | 1.00 |
| governing system | $T_1$ | 0.15 |
| Nonreheat turbine system | $T_{ch}$ | 0.40 |
| | K | 100 / (% steady-state speed regulation) |

The power at gate $P_{GV}$ is fed to the turbine model, which is represented as $1 / (1 + sT_{ch})$. Then, we obtain the mechanical power $P_m$, the output of this module. Next, $P_m$ will be inputed to the mechanical subsystem, which will be discussed in detail in the next section.

## 2.4 Mechanical Model

In general, the mechanical dynamics of the synchronous machine is quite complicated. The model described here, however, intended for validation of the simulation generator dynamics, is very simple. "The turbine-generator-shaft system is modeled by a single lumped inertia. This simple model is widely used in transient stability studies" [10, 6]. And a differential damping term was also added to the mechanical dynamics model to force a rapid return to steady state following a transient.

This module is built by a second-order non-linear differential equation, known as the swing equation, which can be written in the form

$$J\frac{d^2\theta}{dt^2} + D\frac{d\theta}{dt} = T_a = T_m - T_e \tag{2.42}$$

where:

$T_a$ is the net accelerating torque in N-m.

$T_m$ is the mechanical input torque to the synchronous machine in N-m.

$T_e$ is the electrical or electromagnetic torque in N-m.

J is the lumped moment of inertia in kg-$m^2$.

D is the differential damping coefficient in Joule-sec..

t is the time in sec..

$\theta$ is the angular displacement of the rotor with respect to a stationary axis in mechanical radians (Given in eq.(2.7)).

Differentiating $\theta$ twice gives:

$$\frac{d\theta}{dt} = \omega = \omega_s + \frac{d\delta}{dt}$$

$$\frac{d^2\theta}{dt^2} = \frac{d^2\delta}{dt^2}$$

Substituting into the swing equation (2.42), gives:

$$J\frac{d^2\delta}{dt^2} + D\frac{d\delta}{dt} = T_m - T_e \qquad (N-m) \tag{2.43}$$

We recall from elementary dynamics that power is the product of torque and angular velocity and so, multiplying eq. (2.43) by $\omega$, obtains

$$J\omega\frac{d^2\delta}{dt^2} + D\omega\frac{d\delta}{dt} = P_m - P_e \qquad (W) \tag{2.44}$$

where: $P_m$ is the mechanical input power to the synchronous machine in W.

$P_e$ is the electrical power crossing its airgap in W.

In eq. (2.44), the coefficient $J\omega$ is the angular momentum of the rotor, at the synchronous speed $\omega_s$. It is denoted by M and called the inertia constant of the machine. The unit in which M is expressed is joule-sec. per mechanical radian. The coefficient M here, is not a constant in the strictest sense since $\omega$ does not equal the synchronous speed $\omega_s$ under all conditions of operation. "However, in practice, $\omega$ does not differ significantly from the synchronous speed when the machine is stable, and power is more convenient in calculations than torque" [10], therefore eq. (2.44) is preferred. For stability studies, another constant, H, related to the inertia is very often encountered. H is defined by

$$H = \frac{\textit{stored kinetic energy in megajoules at synchronous speed}}{\textit{machine rating in MVA}}$$

$$= \frac{\frac{1}{2} J\omega_s^2}{S_{mach}} \qquad (2.45)$$

$$= \frac{\frac{1}{2} M\omega_s}{S_{mach}} \qquad (MJ/MVA)$$

where $S_{mach}$ is the three phase rating of the machine in MVA. Solving for M in eq. (2.45), we can get

$$M = \frac{2H}{\omega_s} S_{mach} \qquad MJ/mech-rad. \qquad (2.46)$$

Plugging eq. (2.46) into eq. (2.44), gives

$$\frac{2H}{\omega_s} \frac{d^2\delta}{dt_2} + \frac{D\omega_s}{S_{mach}} \frac{d\delta}{dt} = \frac{P_m - P_e}{S_{mach}} \qquad (2.47)$$

In order to simplify the calculation, we redefine the damping coefficient:

$$D^* = \frac{D\omega_s}{S_{mach}}$$

Thus,

$$\frac{2H}{\omega_s} \frac{d^2\delta}{dt_2} + D^* \frac{d\delta}{dt} = P_m - P_e \qquad (p.u.) \qquad (2.48)$$

This equation leads to a very simple result. In this work, the typical inertia constant H and damping coefficient $D^*$ are taken, which are listed in Table 5.

In order to derive the simulation model, eq. (2.48) can be written as a set of two first-order differential equations by defining:

**Table 5. Mechanical parameters**

| Symbol | Values |
|---|---|
| 2H | 10.0 |
| $D^*$ | 0.08 |

$$x_1 = \delta$$
$$x_2 = \dot{x}_1 = \frac{d\delta}{dt}$$

This gives

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -D^* \frac{\omega_s}{2H} x_2 + \frac{\omega_s}{2H}(P_m - P_e) \end{aligned} \qquad (2.49-a)$$

or in matrix form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & D^* \frac{\omega_s}{2H} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\omega_s}{2H} \end{bmatrix} (P_m - P_e) \qquad (2.49-b)$$

From eq. (2.49-a) or (2.49-b), the block diagram of the mechanical model may be drawn as Figure 17. Here the mechanical power $P_m$ is the input of the mechanical subsystem. And the electrical power $P_e$ can be obtained by two ways, one by equation [2]:

$$P_e = \omega T_e = \omega \times \frac{1}{3}(i_q \psi_d - i_d \psi_q) \qquad (2.50)$$

and the other by equation [6]:

$$P_e = \frac{EV}{x} \sin \delta \qquad (2.51)$$

where $x = x'_d + x_l$

The latter equation (2.51) is used in this work, and its detail will be discussed below.

Assume the generator is modeled as a constant magnitude voltage source behind a transient reactance, then the only generator dynamics are those in the swing equation. When the constant

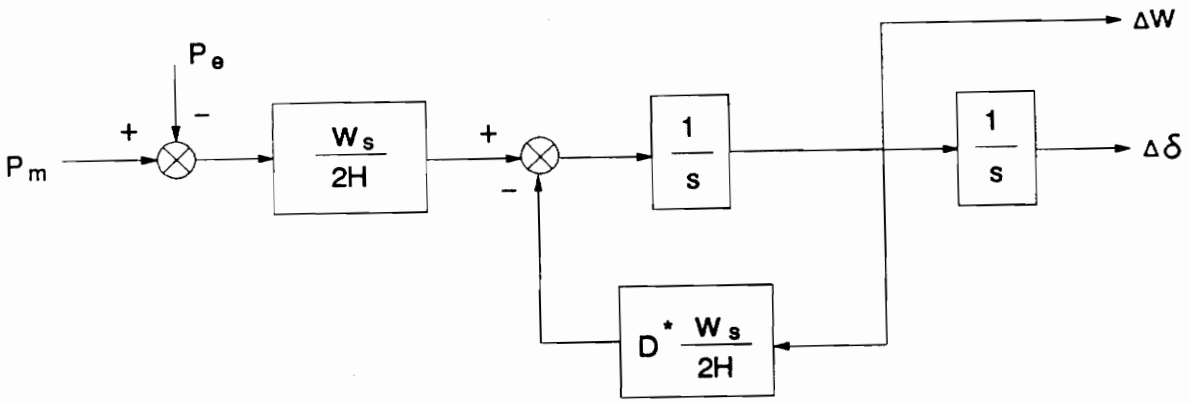



**Figure 17. The block diagram of mechanical model**

voltage behind transient reactance model is connected through a purely reactive line to an infinite bus, an equilibrium solution to the swing equation can be found. For the configuration , the electrical power $P_e$ is satisfied with eq. (2.51). Note that E is the generator voltage magnitude, V is the infinite bus voltage, $x'_d$ is the transient reactance of the generator, $x_l$ is the transmission line reactance, and $\delta$ is the angle of the generator voltage source with respect to the infinite bus voltage. At equilibrium, $\delta$ is constant and the derivatives $\frac{d^2\delta}{dt^2}$ and $\frac{d\delta}{dt}$ are zero, eq. (2.48) becomes

$$P_m - \frac{EV}{x}\sin\delta = 0 \tag{2.52}$$

Thus

$$\delta_{eq} = \sin^{-1}\left[\frac{P_m x}{EV}\right] \tag{2.53}$$

Picking some typical numbers,

$$\begin{aligned} P_m &= 1.0 \quad p.u. \\ E &= 1.0 \quad p.u. \\ V &= 1.0 \quad p.u. \\ x_l &= 0 \\ x'_d &= 0.43 \quad p.u. \qquad (\textit{Given in Table 3}) \end{aligned}$$

Finally, we obtain the equilibrium angle $\delta_{eq} = 0.4445 rad.$

# Chapter III

# Simulation Algorithm

## *3.1 Digital Non-linear System Simulation*

"As is evident from the overall transfer function, the clamp and saturation non-linearities prevent a straight forward application of linear system analysis. The so called 'direct method' of system analysis is the natural choice for real time digital simulation" [3]. This method is required to have a fixed sampling rate that is a multiple of the fundamental frequency, the reasons for which are given in the reference [3]. Many elegant numerical methods like Runge-Kutta and the trapezoidal integration are examples of direct methods. They usually employ a small time step and the result is updated after each iteration. The non-linearities are also computed and updated during the same time interval. If the sampling rate is sufficiently high, the results are close to continuous time analysis. Since the fact that the final simulation should be carried out in real time on a micro-processor, in order to solve the ordinary differential equations, a fast and efficient numerical method with easy adaptability for non-linearities is needed. Elegance and high accuracy, although desirable, are not of utmost importance because of inherent inaccuracies and loss of precision in the other

components of the simulation process, such as A/D, D/A conversion, integer arithmetic truncation, round off errors and so on.

Traditionally, discrete time control systems are implemented by simple first and second-order algorithm, a heuristically derived algorithm (pole-zero mapping), or the analytical solution to the differential equation over a time step [6]. The order 2 Runge-Kutta Algorithm, also called the trapezoidal integration method, currently does find application in real-time calculation on small computers or microprocessors [11]. Here, this algorithm is employed to work on real-time simulation. When given a solution to the differential equation at time k, it uses a simple Euler integration to predict the solution at time $k + 1$. This prediction is then improved by using the trapezoidal rule to provide the solution at time $k + 1$. It offers a very excellent combination of speed and accuracy.

## *3.2 Simulation Scheme*

Algorithmically, the trapezoidal integration method can be described as follows. Based on the linear system theory, for a linear time-invariant system, the dynamic equations are written [12]:

$$\textit{State equation:} \qquad \dot{X}(t) = AX(t) + BU(t) \tag{3.1}$$

$$\textit{Output equation:} \qquad Y(t) = CX(t) + DU(t) \tag{3.2}$$

where **X(t)** is defined as the state vector, **U(t)** as the input vector, **Y(t)** as the output vector and **A**, **B**, **C** and **D** as the coefficient matrices of the system. State equations with a zero-order hold (ZOH) process on the input vector **U** at each sampling interval results in two steps as follows:

$$X^p(k+1) = X(k) + T\dot{X}(k) \tag{3.3}$$

$$X(k+1) = X(k) + \frac{T}{2}\left[\dot{X}(k) + \dot{X}^{p}(k+1)\right] \tag{3.4}$$

where $\dot{X}(k) = AX(k) + BU(k)$ and T is the step size. This is a predictor-corrector algorithm in which the corrector is only applied once, although iteration is possible.

The output equation follows immediately as

$$Y(k+1) = CX(k+1) + DU(k+1) \tag{3.5}$$

The above equations (in matrix form) are used to describe the system differential equations and they are reminiscent of digital filter equations. This is easy to understand since digital filters are mathematically equivalent to continuous systems with sampled data inputs and outputs [3]. Note that for the real-time simulation, the current input $U(k+1)$ is required to produce the current state $X(k+1)$, and the output $Y(k+1)$. So the input must be sampled and the output produced simultaneously, and the intervening computations should take no time. In practice, if sufficient margin is left in the sample window following the output operation for the continuous systems to respond, the phase lag added by the computation time is usually not significant to the system behavior. As mentioned before, a 720 Hz sampling rate, corresponding to $T = 0.001389$ *sec.*, is used by this simulation model. The discussion so far deals with a linear time-invariant system.

On observing the exciter, synchronous machine and governor models, it is not difficult to find that they are not linear time-invariant systems, but include some non-linear functions, such as saturation and limitation. Therefore, treating non-linearities as either inputs or states is a necessary technique in the discrete non-linear generator system. Within the sampling interval, the ZOH is an ideal integrator. Each non-linearity occurring in the model can be sampled, clamped and designated as a state variable.

There is a study [3] which provides a technique to simulate the non-linear functions. As we know, during state transition, the samplers are open. "Before transition into the next sample instant, the

samplers are closed and the new sample is held during the next interval. This introduces a second set of equations which account for the resetting of the states at the sampling or the 'jump' instants. A jump matrix **J** is chosen to satisfy the following equation at the sampling instant." [3]

$$X^{+}(k+1) = J \times X^{-}(k+1) \tag{3.6}$$

where the superscripts + and − denote the values immediately after and before the sampling instant. At the sampling instant, none of the continuous time variables change instantly but the non-linearity clamps do change.

The digital simulation for each subsystem will be explained in section 3.3, 3.4, 3.5 and 3.6.

## *3.3 Discretization of Exciter Model*

The expanded block diagram of the exciter subsystem is shown in Figure 18.

The system coefficient matrix **A** and jump matrix **J** are written as follows:

$$\dot{X}(k) = \begin{bmatrix} -1/T_a & -K_f/T_fT_e & 1 & 0 & 0 \\ 0 & -K_e/T_e & 0 & 1 & -1 \\ 0 & K_f/T_fT_fT_e & -1/T_f & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} X(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta V_T(k)$$

and

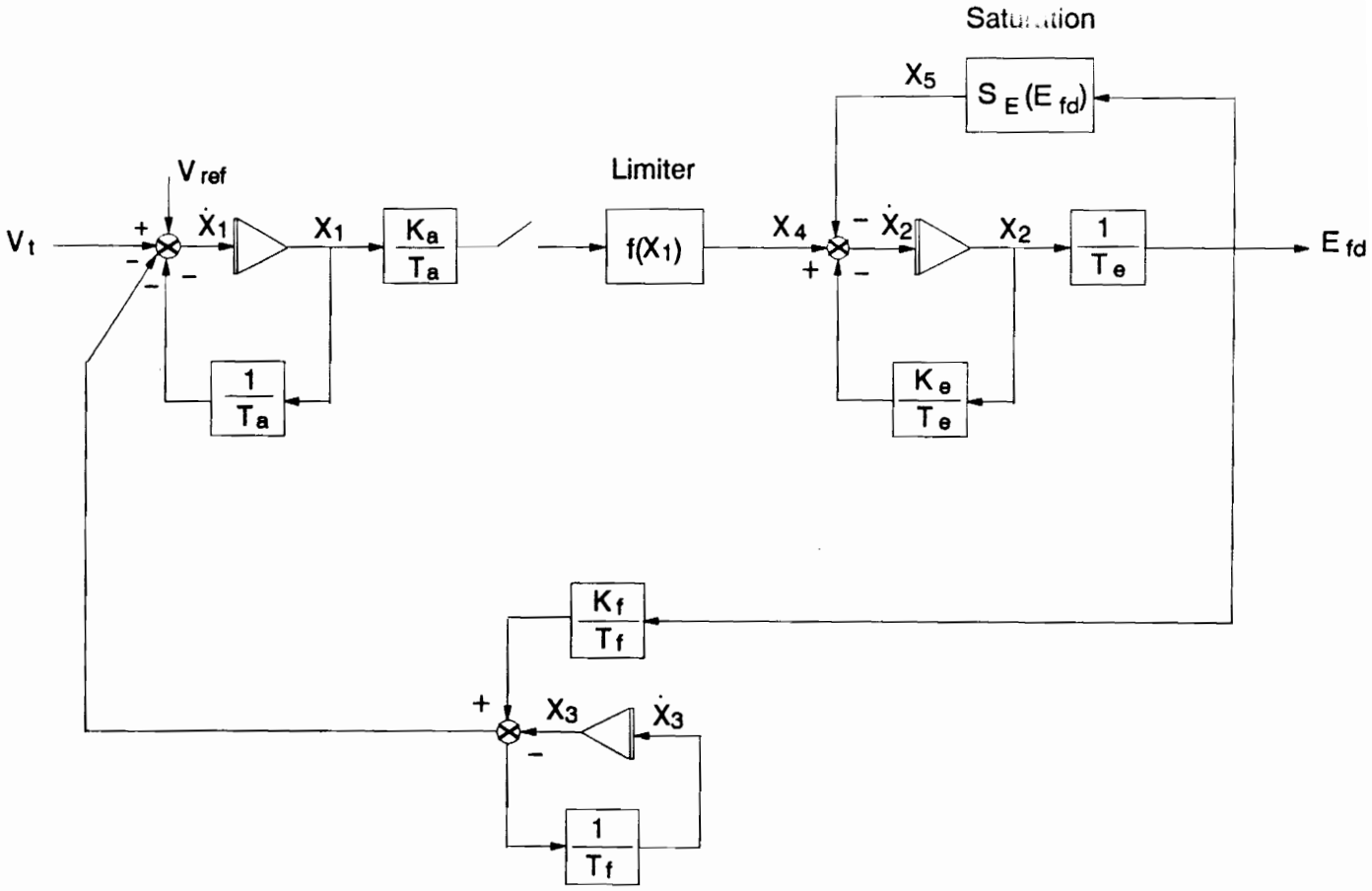



**Figure 18. Non-linear, realization block diagram of the IEEE Type-I excitation system**

$$X^{+}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ (K_a/T_a)\, f(X_1) & 0 & 0 & 0 & 0 \\ 0 & S_E(X_2)/T_e & 0 & 0 & 0 \end{bmatrix} X^{-}(k)$$

where the functions $f(X_1)$ and $S_E(X_2)$ represent the saturation and the limiter non-linearities in the **J** matrix. These are the functions [1] of state variables $X_1$ and $X_2$ respectively. For the rotating rectifier exciter with static voltage regulator having the constants shown in Table 1, the system matrix is given as

$$\dot{X}(k) = \begin{bmatrix} -50.00 & -0.0375 & 1 & 0 & 0 \\ 0.00 & -1.2500 & 0 & 1 & -1 \\ 0.00 & 0.0375 & -1 & 0 & 0 \\ 0.00 & 0.00 & 0 & 0 & 0 \\ 0.00 & 0.00 & 0 & 0 & 0 \end{bmatrix} X(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta V_T(k)$$

Expanding the subsystem difference equations:

$$\dot{X}_1(k) = -1/T_a \times X_1(k) - K_f/(T_f T_e) \times X_2(k) + X_3(k) + (V_{ref} - V_T(k))$$
$$\dot{X}_2(k) = -K_e/T_e \times X_2(k) + X_4(k) - X_5(k)$$
$$\dot{X}_3(k) = K_f/(T_f T_f T_e) \times X_2(k) - 1/T_f \times X_3(k)$$

The jump equations valid at the sampling instants are:

---

[1] The details for the two functions are described in Chapter II.

$$X_4{}^+(k) = \begin{cases} -7.3 & X_1(k) \le X_{1min} \\ X_4{}^-(k) & X_{1min} < X_1(k) < X_{1max} \\ 7.3 & X_1(k) \ge X_{1max} \end{cases}$$

$$X_5{}^+(k) = S_E(X_2(k) / T_e)$$

where

$$X_{1max} = V_{rmax} / (K_a/T_a)$$
$$X_{1min} = V_{rmin} / (K_a/T_a)$$

Using the trapezoidal integration algorithm, the following steps are taken to compute the exciter model performance:

Step 1: Pick starting values for $X(0) = X_0$ at $t = 0$ *i.e.* $k = 0$.

Step 2: Calculate the derivatives of state variables $\dot{X}(k)$ at $t = T \times k$ by eq. (3.1).

Step 3: Calculate the predicting state variables $X^p(k+1)$ at $t = T \times (k+1)$ by eq. (3.3).

Step 4: Find the derivatives of predicting state variables $\dot{X}^p(k+1)$ at $t = T \times (k+1)$ by eq. (3.1).

Step 5: Update the state variables $X(k+1)$ at $t = T \times (k+1)$ by eq. (3.4).

Repeat steps 2 through 5 for the next sampling interval, where all the calculation must be completed within the given sampling interval for the real-time simulation. At the end of each step, the output $E_{fd}$ follows immediately as

$$E_{fd}(k+1) = 1/T_e \times X_2(k+1)$$

## 3.4 Discretization of Synchronous Machine Model

The realization block diagrams of the synchronous machine are shown in Figures 19 and 20, which were derived from Figures 12 and 13; and these have the typical constants shown in Tables 2 and 3.

It is obvious that the simulation model for d-axis of the synchronous machine is considered as a seventh order subsystem, and that for q-axis as a third order system. In order to increase simulation speed and save computer space, both the d-axis and q-axis models are separated into three and two parts, respectively.

Expanding the state differential equations for part I of d-axis model:

$$\begin{aligned} \dot{X}_{61}(k) &= X_{62}(k) \\ \dot{X}_{62}(k) &= -8.36 \times X_{61}(k) - 7.8 \times X_{62}(k) + E_{df}(k) \end{aligned}$$

and the relevant output equation:

$$Y_1(k+1) = 0.539 \times X_{61}(k+1) + 0.443 \times X_{62}(k+1) + 0.028 \times E_{fd}(k+1)$$

For part II of d-axis model:

$$\begin{aligned} \dot{X}_{71}(k) &= X_{72}(k) \\ \dot{X}_{72}(k) &= -8.36 \times X_{71}(k) - 7.8 \times X_{72}(K) + i_d(k) \end{aligned}$$

and

$$\begin{aligned} Y_2(k+1) = &-(4.32 \times X_{71}(k+1) + 6.68 \times X_{72}(k+1)) + 0.74 \times i_d(k+1) \\ &+ 0.044 \times [i_d(k+1) - i_d(k)] / T \end{aligned}$$

And for part III of d-axis model:

$$\dot{X}_{81}(k) = X_{82}(k)$$
$$\dot{X}_{82}(k) = X_{83}(k)$$
$$\dot{X}_{83}(k) = -0.77 \times X_{81}(k) - 7.93 \times X_{82}(k) - 8.12 \times X_{83}(k) + Y_3(k+1)$$

where

$$Y_3(k+1) = Y_1(k+1) + Y_2(k+1)$$

Finally, the output for d-axis is:

$$\psi_d^-(k+1) = 8.36 \times X_{81}(k+1) + 7.8 \times X_{82}(k+1) + X_{83}(k+1) - 0.225 \times i_d(k+1)$$

where the superscript – denotes the value before dealing with saturation.

Next, expanding the state differential equations and the output equations:

$$\dot{X}_9(k) = -6.76 \times X_9(k) + i_q$$
$$Y_4(k+1) = -20 \times X_9(k+1) + 4.16 \times i_q + 0.63 \times [i_q(k+1) - i_q(k)] / T$$

and

$$\dot{X}_{101}(k) = X_{102}(k)$$
$$\dot{X}_{102}(k) = -4.19 \times X_{101}(k) - 8.91 \times X_{102}(k) - Y_4(k+1)$$

Finally, the output equation of q-axis is as follows:

$$\psi_q^-(k+1) = 6.54 \times X_{101}(k+1) + 0.97 \times X_{102}(k+1) - 0.225 \times i_q(k+1)$$

Thus the flux linkage of the synchronous machine $\psi^-$ ( before dealing with saturation) is given as:

$$\psi^{-} = \sqrt{\psi_d^{-2} + \psi_q^{-2}}$$

Similar to the simulation of the exciter model, the state equations and output equations can be computed by the trapezoidal integration algorithm, where the procedure is exactly the same as that described in Section 3.3. Remember until now we have not considered the saturation non-linearity function $S_G(V_T)$, which is defined in Section 2.2.6, for the machine model. The saturation can also be dealt with the specified technique which is mentioned in Section 3.2. Here, the relevant 'jump' term is written as follows:

$$\psi^{+}(k+1) = \begin{cases} \psi_I(k+1) - S_G(V_T(k+1) - 0.7) \times \psi^{-}(k+1) & V_T(k+1) > 0.7 \ p.u. \\ \psi_I(k+1) & V_T(k+1) \leq 0.7 \ p.u. \end{cases}$$

where the superscripts + and − denote the value after and before dealing with saturation at the same sampling window, and

$\psi_I$ = Flux linkage when saturation effect is neglected

$\psi$ = Flux linkage when saturation is considered

## *3.5 Discretization of Governor-turbine Model*

As we know, the specified block diagram for the EH speed-governing system and nonreheat steam turbine has been given in Figure 16 (B) in the previous chapter, and the relevant parameters are listed in Table 4. The non-linear realization block diagram for this model is drawn in Figure 21.

Like the calculation for the exciter and synchronous machine models, the trapezoidal integration algorithm may be performed to solve both the state difference equations and the output equations

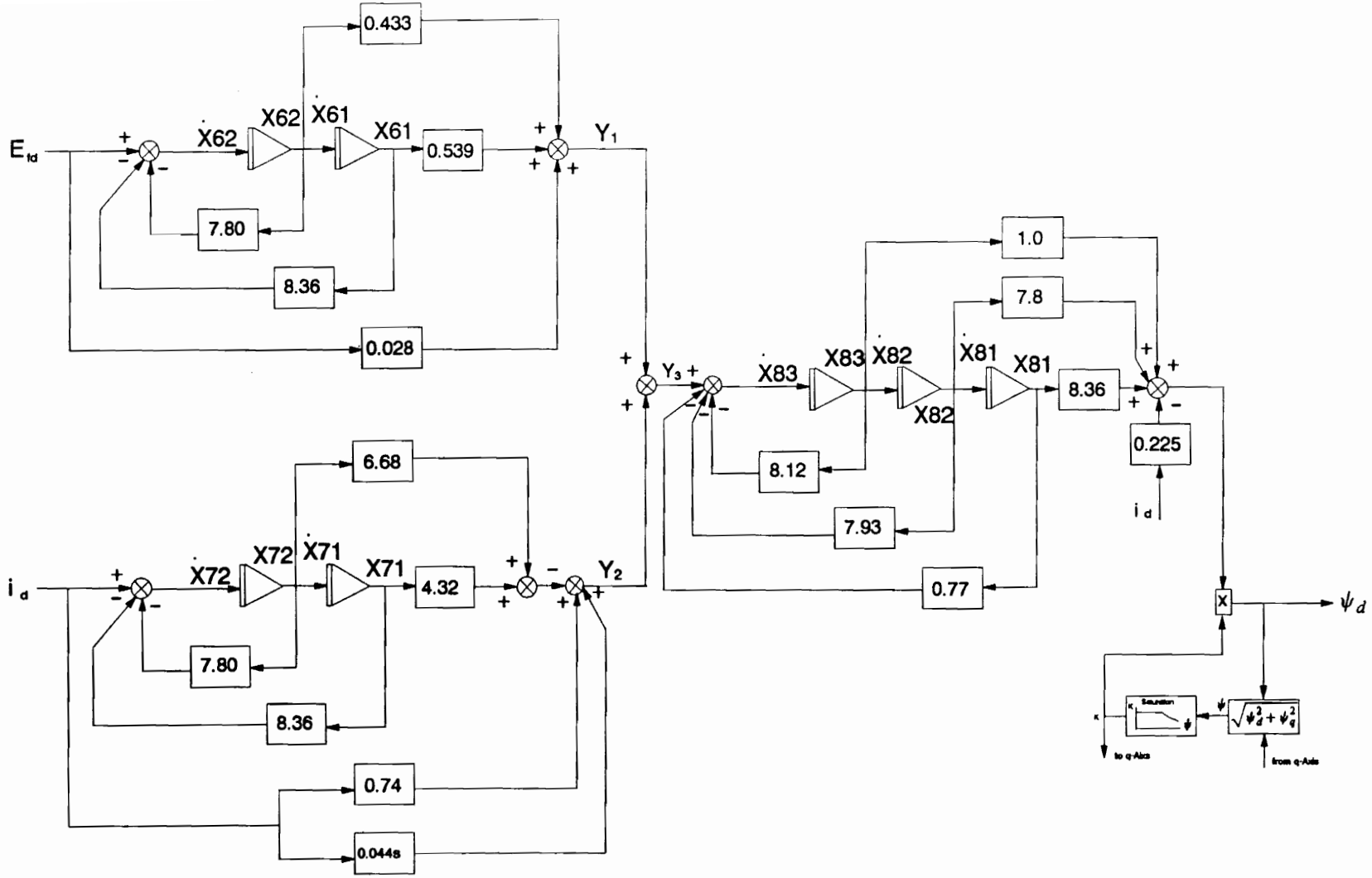



Figure 19. Non-linear, realization block diagram of Solid Iron Rotor Model for d-axis

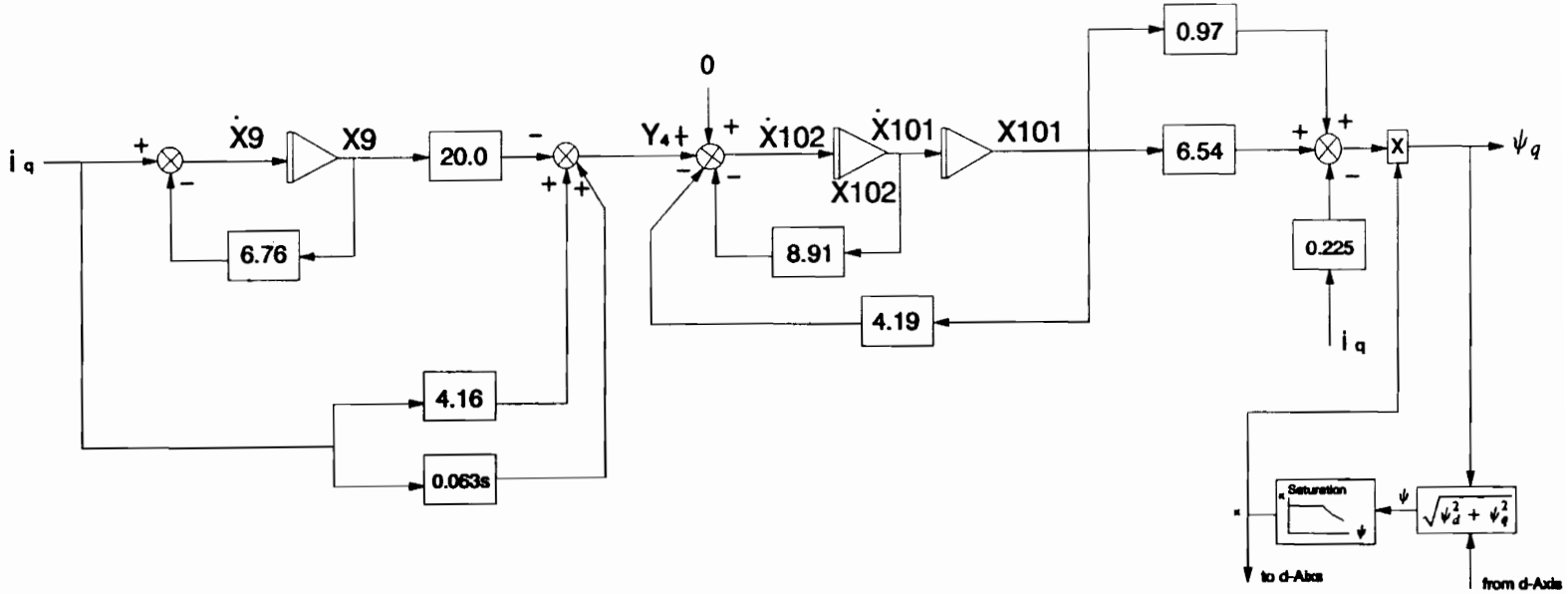



Figure 20. Non-linear, realization block diagram of Solid Iron Rotor Model for q-axis

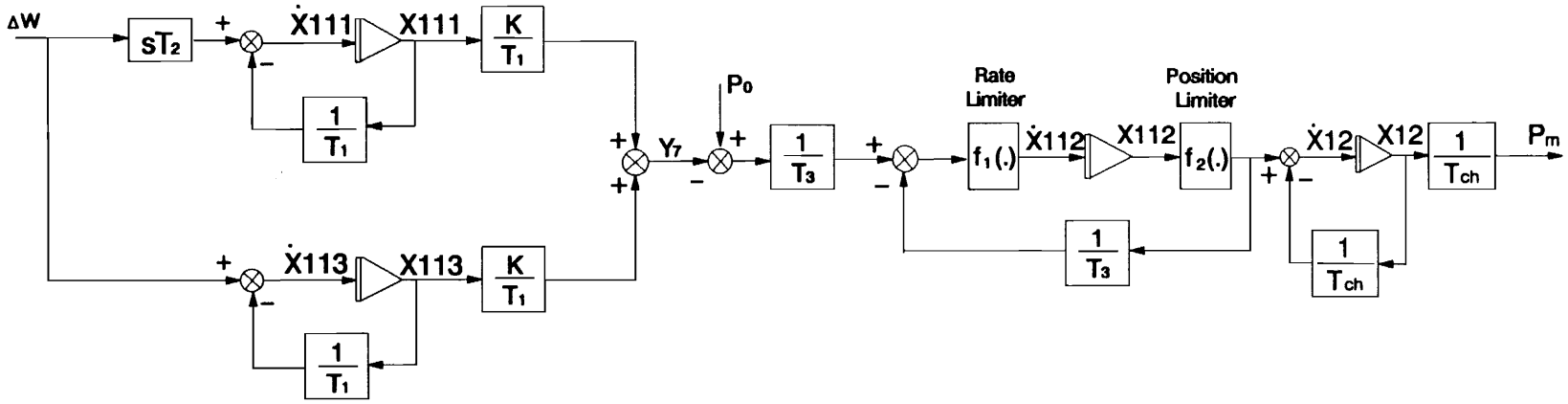



Figure 21. Non-linear, realization block diagram of the Westinghouse EH speed-governor & turbine system

for the governor-turbine, where the procedure is exactly the same as that described in Section 3.3. Based on Figure 21, the relevant equations are written as follows:

$$
\begin{aligned}
\dot{X}_{111}(k) &= -1/T_1 \times X_{111}(k) + T_2 \times (\dot{\delta}(k) - \dot{\delta}(k-1))/T \\
\dot{X}_{113}(k) &= -1/T_1 \times X_{113}(k) + \dot{\delta}(k) \\
Y_7(k+1) &= K/T_1 \times (X_{113}(k+1) + X_{113}(k+1)) \\
\dot{X}_{112}(k) &= -1/T_3 \times X_{112}(k) + 1/T_3 \times (P_0 - Y_7(k+1)) \\
\dot{X}_{12}(k) &= -1/T_{ch} \times X_{12}(k) + X_{112}(k+1) \\
P_m &= 1/T_{ch} \times X_{12}(k+1)
\end{aligned}
$$

where the values of $T_1$, $T_2$, $T_3$, $T_{ch}$ and $K$ are listed in Table 4.

In Figure 21, both non-linear functions $f_1(.)$ and $f_2(.)$ represent the change of rate limiter and position limiter, respectively [2]. These are treated by the 'jump' term which satisfies the following equations at the sampling instant,

$$
f_1(.) = \begin{cases} \dot{P}_{down} & \dot{X}_{112}(k) \le \dot{P}_{down} \\ \dot{X}_{112}(k) & \dot{P}_{down} < \dot{X}_{112}(k) < \dot{P}_{up} \\ \dot{P}_{up} & \dot{X}_{112}(k) \ge \dot{P}_{up} \end{cases}
$$

where $\dot{P}_{up}$ and $\dot{P}_{down}$ have been defined in the previous chapter. And

$$
f_2(.) = \begin{cases} P_{\min} & X_{112}(k) \le P_{\min} \\ X_{112}(k) & P_{\min} < X_{112}(k) < P_{\max} \\ P_{\max} & X_{112}(k) \ge P_{\max} \end{cases}
$$

where $P_{\max}$ and $P_{\min}$ have already been specified previously.

---

[2] The definition of these non-linear functions are given in Chapter II.

## 3.6 Discretization of Mechanical Model

This system is linear. The realization block diagram of the model is shown in Figure 22, which is derived from Figure 17. The mechanical constants for this model have been given in Table 5.

Note that

$$X_{131}(k) = \delta(k)$$
$$X_{132}(k) = \dot{\delta}(k)$$

we rewrite the discretized system equations as:

$$\dot{X}_{131}(k) = X_{132}(k)$$
$$\dot{X}_{132}(k) = -D^* \frac{\omega_s}{2H} \times X_{132}(k) + \frac{\omega_s}{2H} [P_m(k) - P_e(k)]$$

and

$$X_{131}(k+1) = X_{131}(k) + \frac{T}{2} [\dot{X}_{131}(k+1) + \dot{X}_{131}(k)]$$
$$X_{132}(k+1) = X_{132}(k) + \frac{T}{2} [\dot{X}_{132}(k+1) + \dot{X}_{132}(k)]$$

where $X_{131}(k+1)$ represents the torque angle $\delta$, and $X_{132}(k+1)$ is the speed derivative, denoted by $\Delta\omega$ or $\dot{\delta}$ at $t = T \times (k+1)$.

As was done while dealing with exciter, synchronous machine and governor- turbine models, the state equations and output equations of this subsystem can be solved by the trapezoidal rule too, whose steps have been explained in detail in Section 3.3.

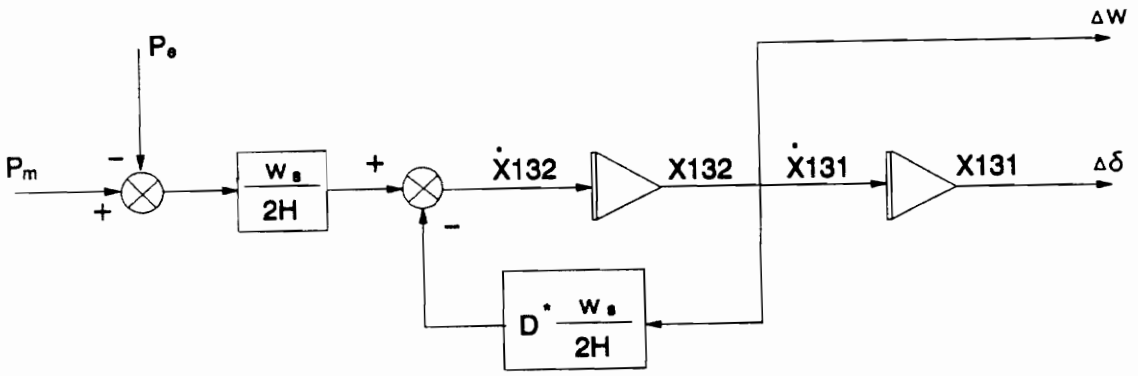



**Figure 22. Non-linear, realization block diagram of the mechanical system**

Section 3.3 through 3.6 have described how to perform the numerical calculation by the trapezoidal integration algorithm for the whole discrete generator model. In this work, first the digital computer simulation is implemented by a FORTRAN simulation program (See Appendix B), i.e. an off-line simulation on a Microvax computer. However, we lay emphasis on the digital computer real-time simulation, the implementation of real-time simulation on the Motorola MC68020, and the software organization which will be discussed in Chapter V in detail.

# Chapter IV

# Transformation of Input and Output

Many of the common forms of the generator equations treat machine armature currents $i_d$ and $i_q$ as inputs and machine terminal voltages $V_d$ and $V_q$ as outputs, where $i_d$, $i_q$, $V_d$ and $V_q$ are variables in the d-q domain. However, the real currents, measured from either the real power system or the simulating system, are of three phase components $i_a$, $i_b$ and $i_c$, and the most important components of the simulation scheme are the three phase voltages $V_a$, $V_b$ and $V_c$, which form the output of the generator simulation model.

The well known Park's transformation in machine theory can easily solve this problem. Transformation from the phase a, b, c domain to the d-q domain involves Park's transform (PT), and conversely, involves inverse Park's transform (IPT). The simulation model of the generator system is as shown in Figure 23 [3].

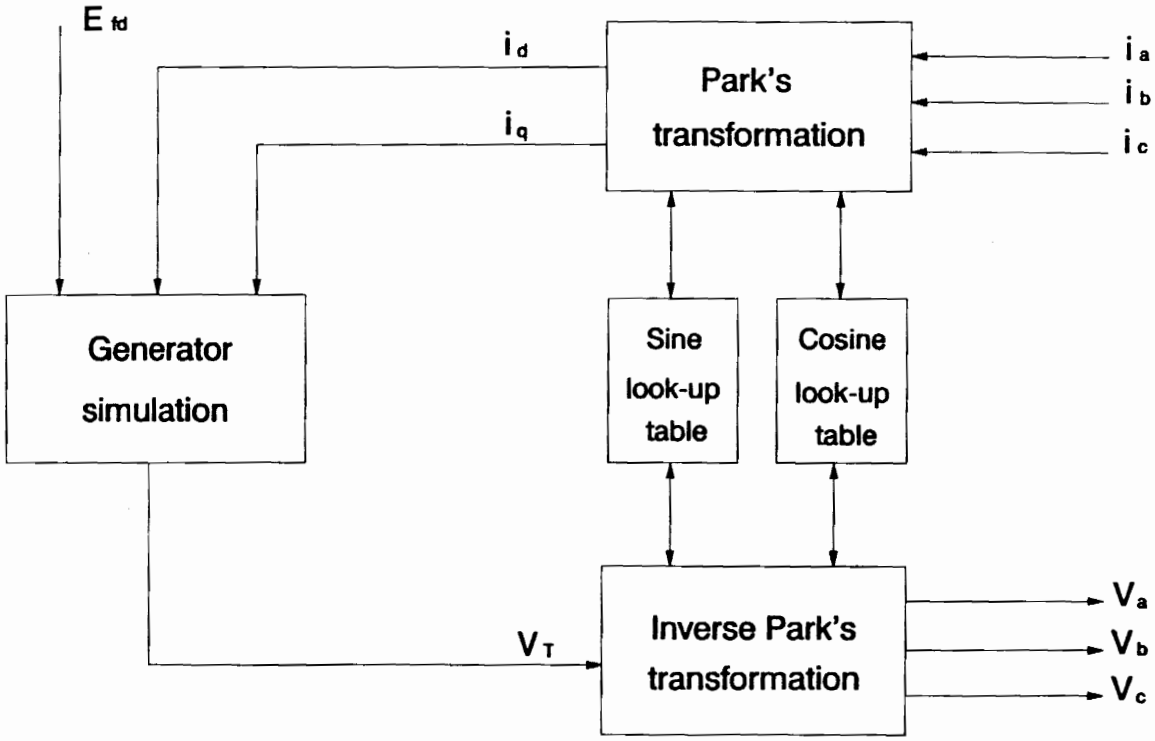



**Figure 23. Typical generator simulation scheme**

## 4.1 Park's and Inverse Park's Transformation

In this simulation system, the Park's transformation is used to generate the direct and quadrature axes currents $i_d$, $i_q$. This transformation is given by:

$$\begin{bmatrix} i_0 \\ i_d \\ i_q \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \cos(\theta) & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ \sin(\theta) & \sin(\theta - 2\pi/3) & \sin(\theta + 2\pi/3) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{4.1}$$

where [3]

$$\theta = \omega t + \delta + \frac{\pi}{2} = (\omega_s + \Delta\omega)t + \delta + \frac{\pi}{2} \tag{4.2}$$

The sample values of three phase currents $i_a$, $i_b$, $i_c$ are obtained from either the real power system or a power system simulator at each 1 / 720 second, which is suitably scaled. Moreover, the transformation is based on per unit (p.u.).

The inverse Park's transformation is applied to produce the machine terminal three phase voltage waveforms. The transformation is given by:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} 1 & \cos(\theta) & \sin(\theta) \\ 1 & \cos(\theta - 2\pi/3) & \sin(\theta - 2\pi/3) \\ 1 & \cos(\theta + 2\pi/3) & \sin(\theta + 2\pi/3) \end{bmatrix} \begin{bmatrix} V_0 \\ V_d \\ V_q \end{bmatrix} \tag{4.3}$$

Since we have got the machine terminal voltage $V_T$, a slight improvisation can be made here to be used in the transformation. As we know,

---

[3] In this work, set $\omega_{ref} = \omega_s = 60\ Hz$.

$$
\begin{aligned}
V_d &= V_T \sin\delta \\
V_q &= V_T \cos\delta
\end{aligned}
\tag{4.4}
$$

Omitting the zero sequence voltage component, normalizing and rewriting eq. (4.3):

$$
\begin{aligned}
V_a &= V_d \cos\theta + V_q \sin\theta \\
V_b &= V_d \cos(\theta - 2\pi/3) + V_q \sin(\theta - 2\pi/3) \\
V_c &= V_d \cos(\theta + 2\pi/3) + V_q \sin(\theta + 2\pi/3)
\end{aligned}
\tag{4.5}
$$

Plugging eq. (4.3) into (4.4), gives:

$$
\begin{aligned}
V_a &= V_T \sin(\theta + \delta) \\
V_b &= -\frac{V_a}{2} - \frac{\sqrt{3}}{2} V_T \cos(\theta + \delta) \\
V_c &= -V_a - V_b
\end{aligned}
\tag{4.6}
$$

This equation is used to generate the three phase outputs voltages, where $V_T$ is suitably scaled, so that 1 p.u. $V_T$ in d-q domain generates 1 p.u. machine output (in sinusoidal form) in the phase a-b-c domain. However this transformation has the following limitations:

- The zero sequence quantities are omitted and so that zero sequence can not be obtained.
- The inverse Park's transform has been manipulated to accept $V_T$ instead of $V_d$ and $V_q$.

It should be noted that the inverse Park's transform may be implemented without any of the above limitations, if all of the quantities $V_d$, $V_q$ and $V_0$ are available [3].

## 4.2 Microprocessor Implementation

Implementing eq. (4.1) and (4.6) requires finding sines and cosines of the cumulative rotor angle $\theta$ (i.e. $\left[\omega t + \delta + \frac{\pi}{2}\right]$) ,which changes with time. Sine and cosine look up tables from 0 to $\frac{\pi}{2}$ radians with a resolution of 0.001 radians were developed.[4] There are a total of 1572 entries in each table. A single entry takes one word of memory, so each table occupies slightly over 3K bytes of memory. The calculation rate of the Park's transform and inverse Park's transform modules (PT & IPT) is 720 Hz. This corresponds to 30° of a 60 Hz waveform. In each time interval, the values of $i_d$ and $i_q$ are computed by the PT module, and the values of $V_a$, $V_b$ and $V_c$ are calculated and stored in memory by the IPT module [3].

The look up tables are stepped through with an angle change of 30° each calculation time. A count of the total angle, $\theta$, is maintained in each time interval. Whenever both of PT and IPT modules are executed one time, the count of $\theta$ is incremented by a value which is sum proportional to the 30° advance in phase and changes in $\omega$ and $\delta$. Here this stepping interval (SI) is regarded as the angle between samples on the current or voltage wave.

No. of entries in each table ( 0 - $\frac{\pi}{2}$ rad.) = 1572 words
Stepping interval for an angle $\frac{\pi}{6}$ rad. (i.e. 30° ) = 524 words

When the frequency is keeping at precisely $\omega_s$ (60 Hz), the stepping interval is 524 words of memory corresponding to 30°. However, if the frequency deviates from the synchronous frequency, i.e. $\Delta f$ or $\Delta\omega$ has some value, the stepping interval will change. For example, for $\Delta f = 2\,Hz$ i.e. $\Delta\omega = 4\pi\ rad./sec.$, the new stepping interval is:

$$524 + \frac{524}{30} \times 4\pi \times \frac{180}{\pi} \times 0.001389 \simeq 542\ words$$

[4] The programs which develop the sine and cosine look up tables are listed in Appendix B.

which corresponds to an angle of 31°. For this simulation model, the actual rotor angular speed $\omega$ is obtained from the mechanical system, and then the calculation for finding the stepping interval can be done during each time interval.

Changes in $\delta$ may be determined similarly. Assume that $\delta$ is varying from one calculation cycle to another. Since $\delta$ just changes the phase but not the frequency, an offset which is proportional to $\delta$ will be added to the stepping interval as the phase changes, while, the offset corresponding to the $\delta$ in the previous calculation cycle should be subtracted. In each calculation cycle, the real torque angle $\delta$ could also be found from the mechanical system.

For example, at calculation cycle $k$, i.e. at time $t = T \times k$, $\theta$ is:

$$\theta_k = (\omega t + \delta + \frac{\pi}{2})_k$$

If the new stepping interval due to changes in $\omega$ and $\delta$ is $\gamma$, then $\theta$ at calculation cycle $k + 1$ has the value:

$$\begin{aligned} \theta_{k+1} &= (\omega t + \delta + \frac{\pi}{2})_k + \gamma \\ &= \theta_k + \gamma \end{aligned}$$

Since the look up tables are stored in memory sequentially, the sines and cosines would be obtained by reading the value at a particular memory location with an offset from the starting address of the table. Since the two tables contain sine and cosine values only for angles lying in the first quadrant, certain logic incorporated in the software would determine the quadrant and hence the sine and cosine of $\theta$.

The steps for finding the particular memory location for $\sin\theta$ and $\cos\theta$ from the look up tables in any time interval are as shown below:

1. Let a setting of $(C500)_{16}$ be the starting address of the sine table, and $(D500)_{16}$ be set as the starting address of cosine table.
2. Perform a logical calculation to determine the quadrant of $\theta$ and transfer $\theta$ to the first quadrant, if it is greater than $\frac{\pi}{2}$ *rad.*.
3. Find the memory location for the particular angle $\theta$ ($0 \leq \theta \leq \frac{\pi}{2}$ *rad.*):

   The memory location of $\sin(\theta) = (C500)_{16} + (2000 \times \theta)_{16}$

   The memory location of $\cos(\theta) = (D500)_{16} + (2000 \times \theta)_{16}$

   Save the two addresses in two data registers $D_{i1}$ and $D_{i2}$, respectively.
4. Transfer the two memory locations from the two data registers, $D_{i1}$ and $D_{i2}$, to two address registers $A_{i1}$ and $A_{i2}$, separately, then read the value of $\sin(\theta)$ and $\cos(\theta)$ from these specified address registers.
5. Do a logical calculation to obtain the real $\sin(\theta)$ and $\cos(\theta)$ for $0 \leq \theta \leq 2\pi$ *rad.*.

This logic is flowcharted in Figures 24, 25, 26, and 27. It is implemented by subprograms PT and IPT, the source codes for which are listed in Appendix B.

Both the PT and IPT modules execute at 720 Hz, and their inputs to PT and IPT are clamped by a control flag at 720 Hz. So the outputs of PT, $i_d$, $i_q$, and the outputs of IPT, $V_a$ , $V_b$ and $V_c$ are also given at the same frequency. Moreover, $V_a$, $V_b$ and $V_c$ are sent to D/A converters at exact sampling point. Also, in order to get smoother output voltage waveforms, the values of $V_a$, $V_b$ and $V_c$ at other three points in each time interval are needed, so that the three phase voltages are actually outputed at 2880 Hz.

In view of the fact that the simulation must be carried out in real-time, fast, simple and efficient numerical methods like the interpolation and extrapolation algorithms are developed to calculate these output points within each time interval. We can either directly interpolate or extrapolate the non-linear output function, or treat the output as a piecewise linear function for each time interval, and then implement interpolation and extrapolation. The former would result in a smoother and more precise sinusoidal waveform, but take longer execution time, while, the latter may complete

calculation faster, but which causing the sinusoidal waveform is not smooth enough, especially in the peak area of the waveform. Mathematically, both linear and non-linear interpolation and extrapolation algorithms can be explained as follows.

1. **Interpolation and extrapolation by piecewise linear function**

Assume that the output voltage waveforms are piecewise linear (i.e. a spline of degree 1 [11]) for each time step. By referring to Figure 28 and using the point-slope of a line, we obtain

$$S_k(t) = V_k + m_k(t - t_k) \tag{4.7}$$

where $m_k$ is the slope of the line and is therefore given by the formula

$$m_k = \frac{V_{(k+1)} - V_k}{t_{(k+1)} - t_k} = \frac{V_{(k+1)} - V_k}{T}$$

Taking phase A as an example, during simulation cycle k, we may get new output $V_{a(k+1)}$ (also denoted as $V_a(t_{k+1})$) from IPT module. The calculation completes at time $t_c$, where $t_k < t_c < t_{k+1}$, and the outputs at the other three points occurs at time $t_k + \frac{iT}{4}$ for $i = 1, 2, 3$. If $(t_k + \frac{iT}{4}) < t_c$, find $V_a(t_k + \frac{iT}{4})$ by extrapolation

$$V_a(t_k + \frac{iT}{4}) = \frac{i}{4}[V_{ak} - V_{a(k-1)}] + V_{ak} \tag{4.8}$$

where $V_{a(k-1)}$ is IPT output at previous sampling point. And if $(t_k + \frac{iT}{4}) \geq t_c$, by interpolation calculate $V_a(t_k + \frac{iT}{4})$

$$V_a(t_k + \frac{iT}{4}) = \frac{i}{4}[V_{a(k+1)} - V_{ak}] + V_{ak} \tag{4.9}$$

As seen above, a logical operation is required to determine which algorithm (extrapolation or interpolation) should be used for calculation of these intermediate output points. This operation is performed by setting an output control flag, 'outputflg', while executing the simulation

module. At the starting point of a sampling window, i.e. $t = t_k$, set outputflg = 0, and at $t = t_c$, reset outputflg = 1. This is implemented in each time interval.

2. **Interpolation and extrapolation by sine function**

Similar to the linear interpolation and extrapolation, taking phase A as an example, during simulation cycle k, we can use $V_{a(k+1)}$, $V_{ak}$ and $V_{a(k-1)}$ to implement extrapolation or interpolation. By referring to Figure 29, assume

$$V_{a(k-1)} = V \sin\theta \tag{4.10}$$

$$\begin{aligned} V_{ak} &\simeq V \sin(\theta + 30°) \\ &= \frac{\sqrt{3}}{2} V \sin\theta + \frac{1}{2} V \cos\theta \\ &= \frac{\sqrt{3}}{2} V_{a(k-1)} + \frac{1}{2} V \cos\theta \end{aligned} \tag{4.11}$$

$$V_{a(k+1)} \simeq V \sin(\theta + 60°) \tag{4.12}$$

From eq.(4.11), we get

$$V \cos\theta = 2V_{ak} - \sqrt{3}\, V_{a(k-1)} \tag{4.13}$$

Now we may find the other three outputs at position $\theta + 37.5°$, $\theta + 45°$ and $\theta + 52.5°$ by extrapolation

$$\begin{aligned} V_{ak7.5} &= V \sin(\theta + 37.5°) \\ &= V \sin\theta \cos 37.5° + V \cos\theta \sin 37.5° \\ &= V_{a(k-1)} \cos 37.5° + (2V_{ak} - \sqrt{3}\, V_{a(k-1)}) \sin 37.5° \\ &= V_{a(k-1)}(\cos 37.5° - \sqrt{3} \sin 37.5°) + V_{ak}(2 \sin 37.5°) \end{aligned} \tag{4.14}$$

On the analogy of eq.(4.14)

$$\begin{aligned} V_{ak15} &= V\sin(\theta + 45°) \\ &= V_{a(k-1)}(\cos 45° - \sqrt{3}\sin 45°) + V_{ak}(2\sin 45°) \end{aligned} \tag{4.15}$$

$$\begin{aligned} V_{ak22.5} &= V\sin(\theta + 52.5°) \\ &= V_{a(k-1)}(\cos 52.5° - \sqrt{3}\sin 52.5°) + V_{ak}(2\sin 52.5°) \end{aligned} \tag{4.16}$$

Comparing the piecewise linear extrapolation and interpolation algorithm with the non-linear extrapolation algorithm, we find that the non-linear algorithm can result in more exact and smooth output waveforms, and the execution time is reasonable. In this simulation, the non-linear extrapolation algorithm is applied, which is implemented by subroutine 'EXTRAPOLATE' (See Appendix B).

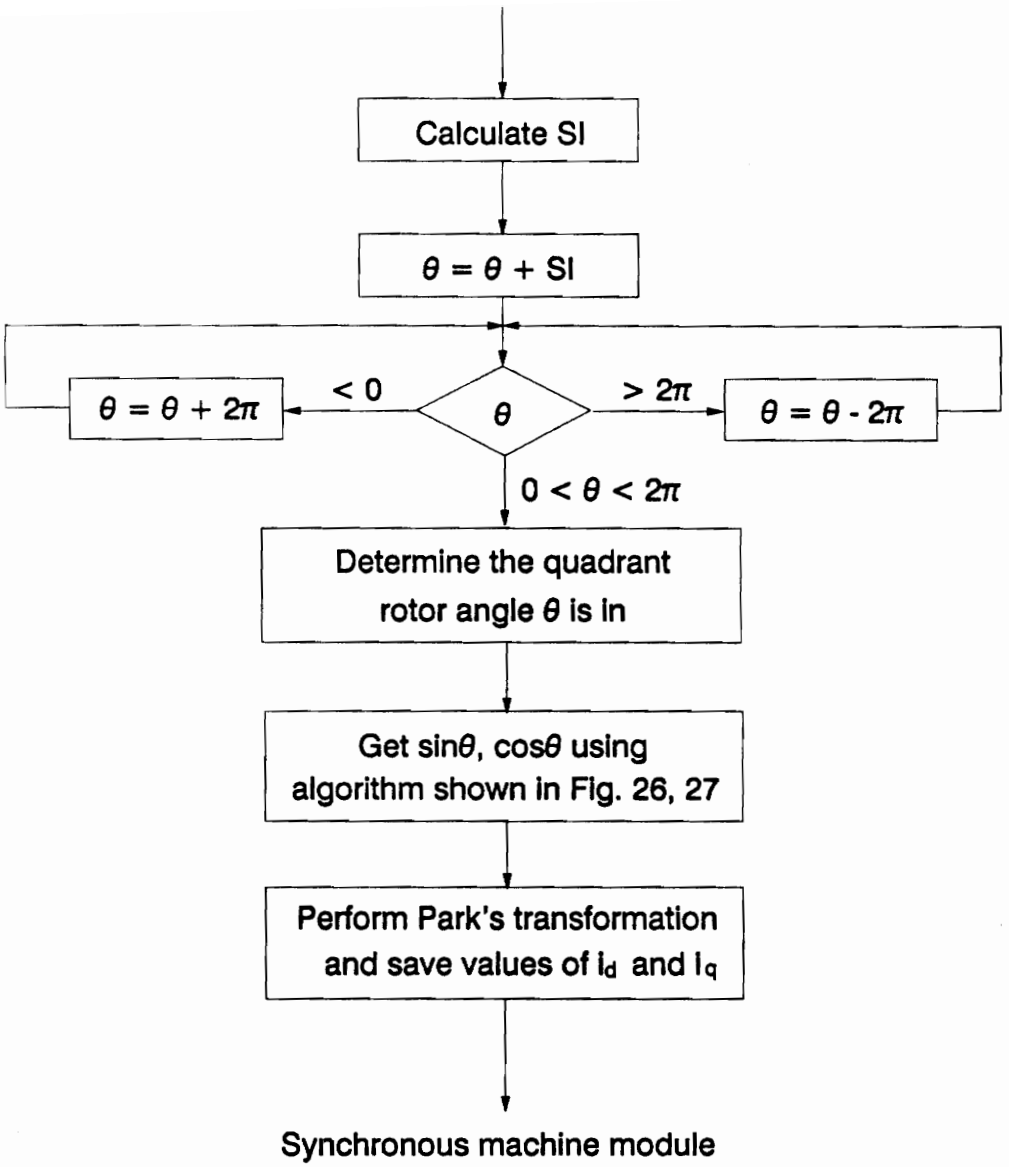



Figure 24. Block diagram of Park Transformation module (PT)

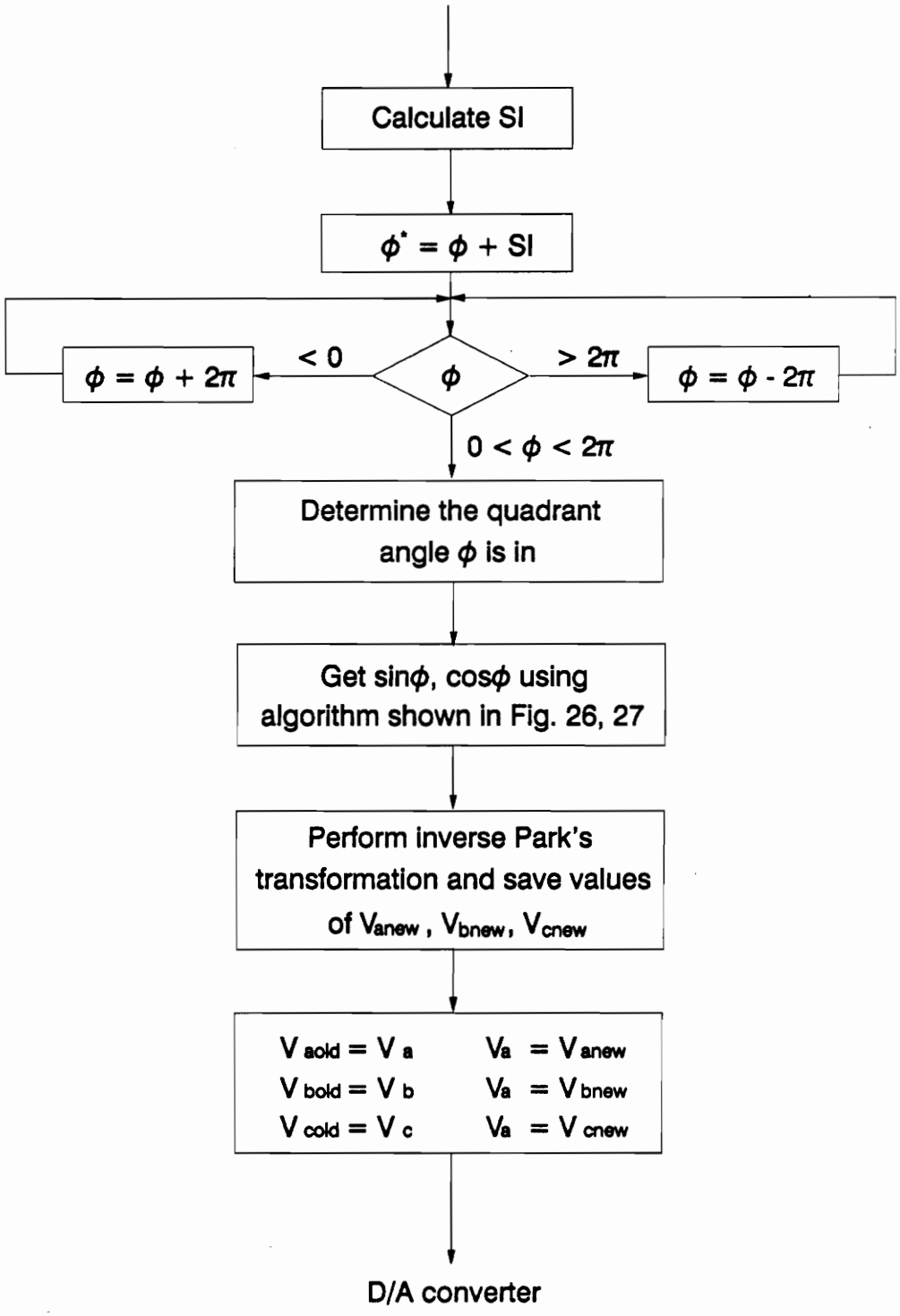



**Figure 25. Block diagram of Inverse Park Transformation module (IPT)**

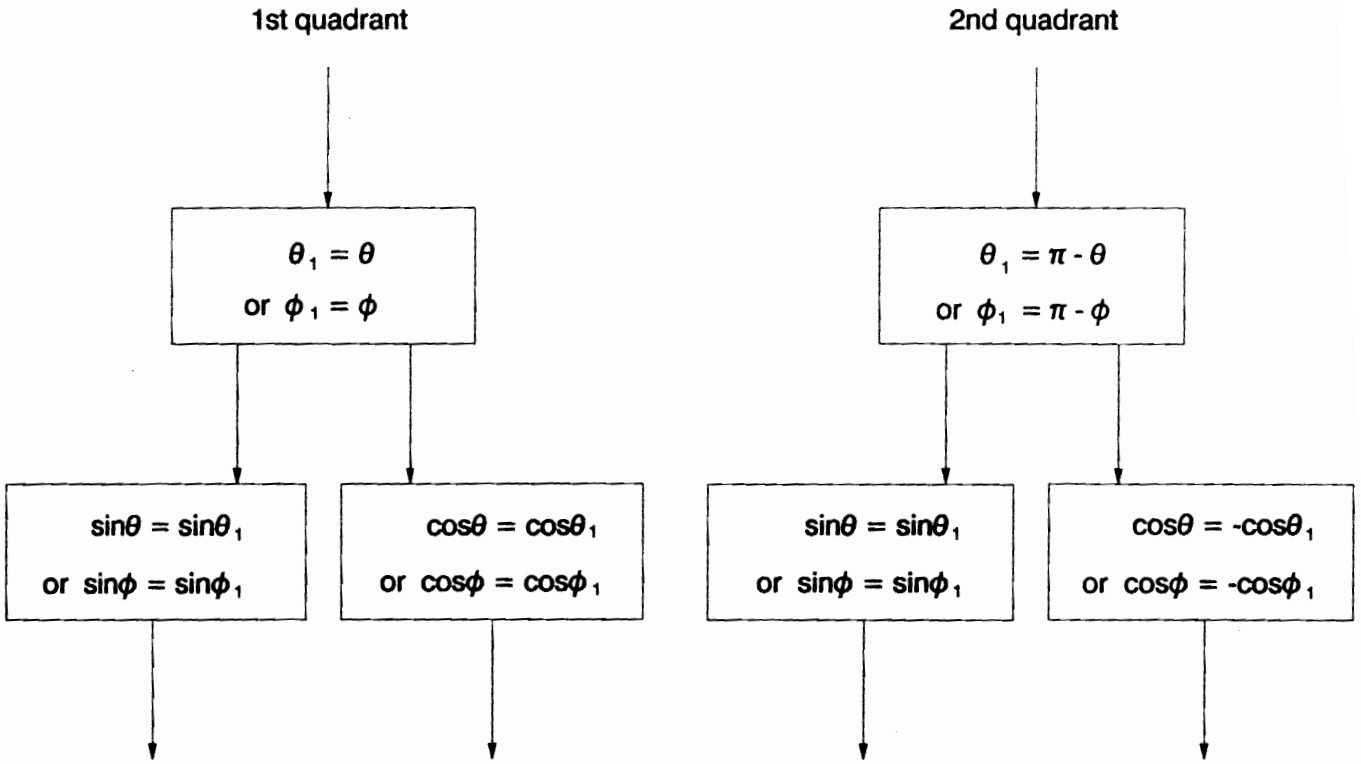



**Figure 26. Table look up $\theta$ or $\phi$ in 1st & 2nd quadrant**

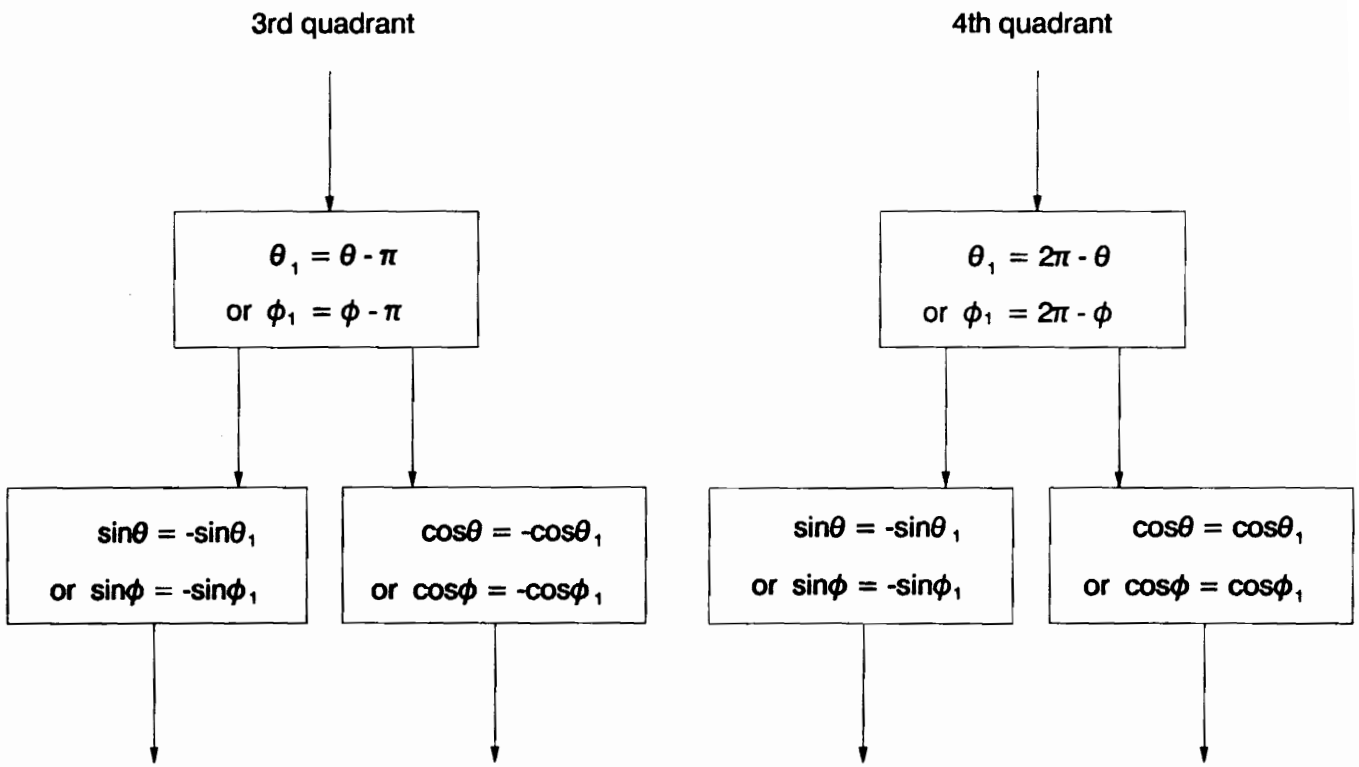
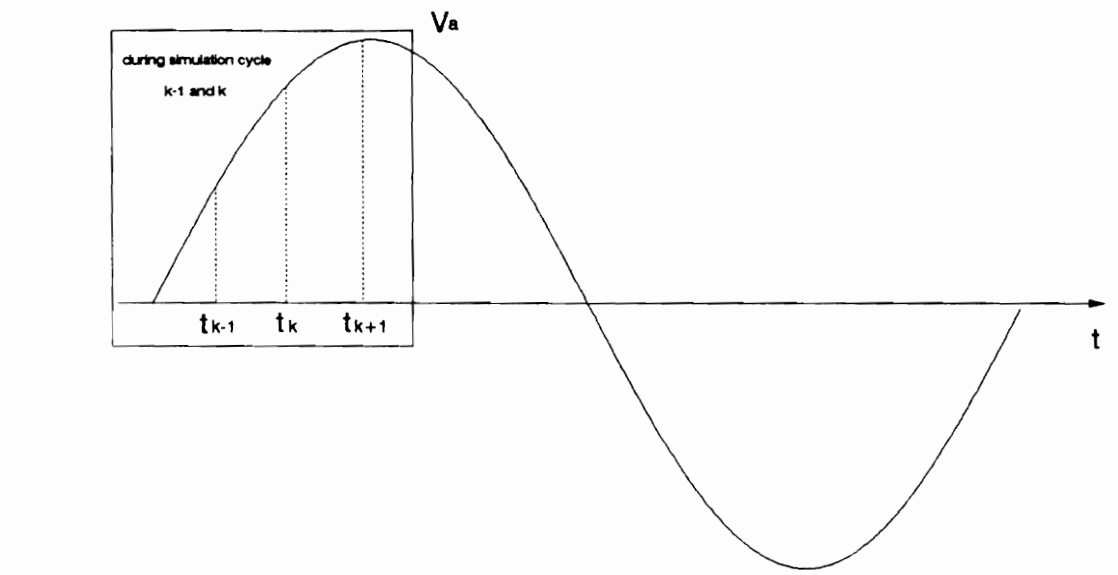



Figure 27. Table look up $\theta$ or $\phi$ in 3rd & 4th quadrant



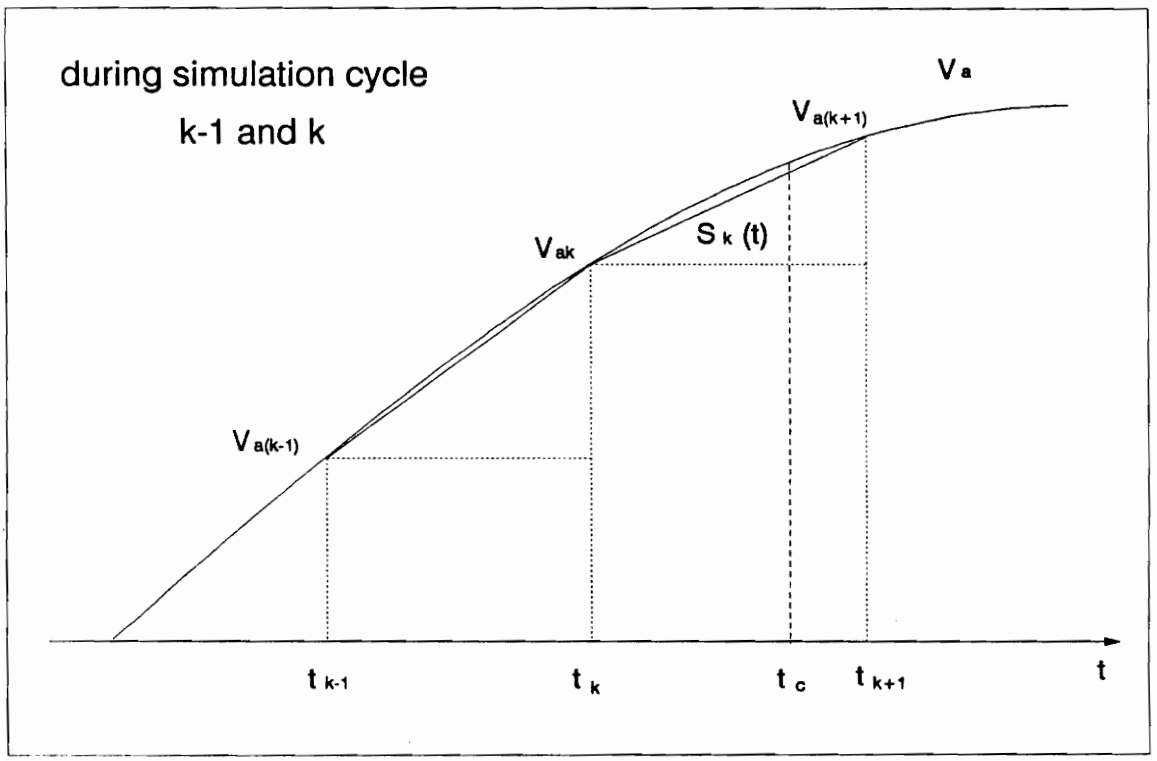




**Figure 28. The extrapolation for a piecewise linear function**

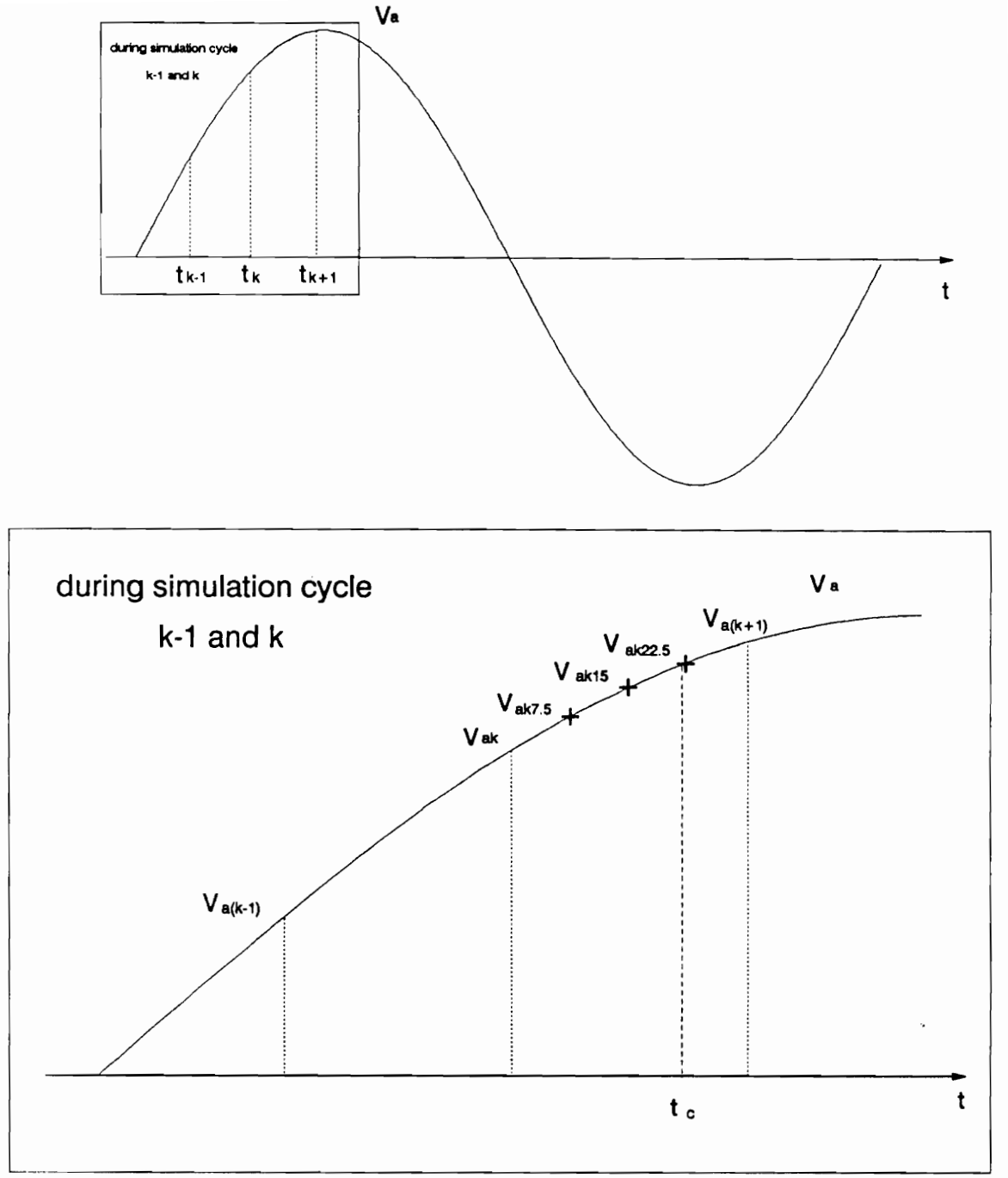



**Figure 29. The extrapolation for a sine function**

# Chapter V

# Implementation and Simulation Results

This chapter discusses the microprocessor implementation of the discretized generator model and the simulation results. There are many possible strategies for implementing the model, each with its own cost benefit tradeoffs. In this work, the real-time simulation of the discretized generator model is implemented by software on the Motorola MC68020.

## *5.1 Hardware Organization*

The hardware for this simulation consists of the following:

1. MicroVAX host computer and VT220 terminal.
2. Motorola MC68020 microcomputer board.
3. Two data translation 12 bit, 16 channel A/D converter, 2 channel D/A converter boards (DT 1401).

4. Three Crown PS-400 watt audio power amplifiers.
5. Tektronix 2221 60 MHz digital storage oscilloscope.
6. Three signal conditioners and three two-stage RC filters.

All the boards are VME compatible making interface easy because of the wide acceptance of this bus in the industry. In addition to the above, this simulation system includes of transformer model, transmission line model and load model. The host computer is supported by the VAX/VMS operating system and the Digital Command Language (DCL), and software for this host includes a Micro Assembler and a FORTRAN compiler. The hardware is organized as shown in Figure 30.

## *5.2 Software Organization*

"Microprocessor programming for real-time, time-constrained applications is very much different from ordinary programming. It forces the programmer to be efficient often at the expense of neatness and order." [3] Most real-time programs are not structured, since subroutine calls and jumps take up too much time. Difficulty in deciphering real-time software is compounded by the fact that it is in assembly language.

In this application, the processor is interrupted at a frequency of 2880 Hz, which is set by programming the on board MC68901 timer. The interrupts, which occur at equal intervals of 347.2 $\mu$s, form the basic calculation time frame. There are two different sampling rates in the simulation. The exciter, synchronous machine, governor-turbine, mechanical and Park's transformation modules are executed for every 1388.8 $\mu$s (720 Hz). The inverse Park's transformation and extrapolation subprogram are also solved once every 1388.8 $\mu$s (720 Hz), while the values of $V_a$, $V_b$ and $V_c$ are calculated for subintervals and sent to D/A converters (DAC) at the rate of 2880 Hz. Note that the outputs, $V_a$, $V_b$ and $V_c$, resulting from inverse Park's transformation are transferred to the DAC at

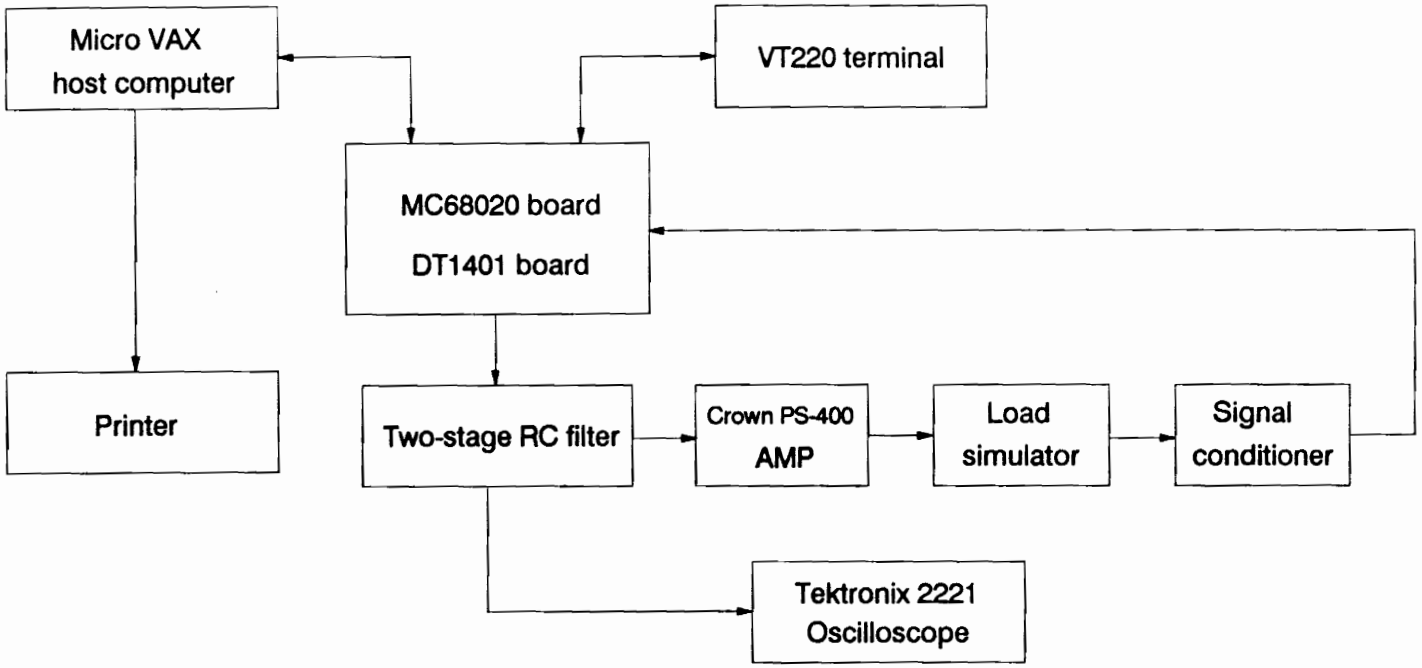



**Figure 30. Hardware organization**

exactly the sampling point. Therefore the effective sampling rate for the inverse Park's transformation and interpolation & extrapolation, i.e. output module, is 2880 Hz. The software organization is shown in Figure 31.

The software consists of the following modules and a description of each is given below:

**The Main Module:** This module (which is named as 'Generator Model' and listed in Appendix B) is made up of the exciter system, the synchronous machine system, the governor-turbine system, the mechanical system (i.e. swing equation), the Park's and inverse Park's transformation models, and the extrapolation subroutine. In addition to the above programs, four functional look up tables (Sine, Cosine, Exponential and square root) and a machine saturation subroutine are embraced in the main module.

The development and calculation of the particular memory location for both sine and cosine look up tables have been discussed in the previous chapter. Similarly, an exponential look up table from 0 to 5 with a resolution of 0.002 and a square root table from 0 to 16 with a resolution of 0.004 are also developed. There are a total of 2500 entries in the exponential table and 4000 entries in the square root table. One entry occupies one word of memory. The computation for determining the specific location for exponential and square root look up tables in a time step is as explained below:

1. Let the setting of $(9100)_{16}$ be the starting address of the exponential table, and $(A500)_{16}$ be as the starting address of the square root table.
2. Find the memory location for the specific state variable $X_2$ and machine terminal voltage $V_T$:

   the memory location of $\exp(X_2) = (9100)_{16} + (1000 \times X_2)_{16}$

   the memory location of $\sqrt{V_T} = (A500)_{16} + (500 \times V_T)_{16}$

   and then store these memory addresses in two data registers, $D_{i1}$ and $D_{i2}$, respectively.
3. Transfer the memory locations from $D_{i1}$ and $D_{i2}$ to two address registers $A_{i1}$ and $A_{i2}$, separately, then read the value of $\exp(X_2)$ from $A_{i1}$ and $\sqrt{V_T}$ from $A_{i2}$.

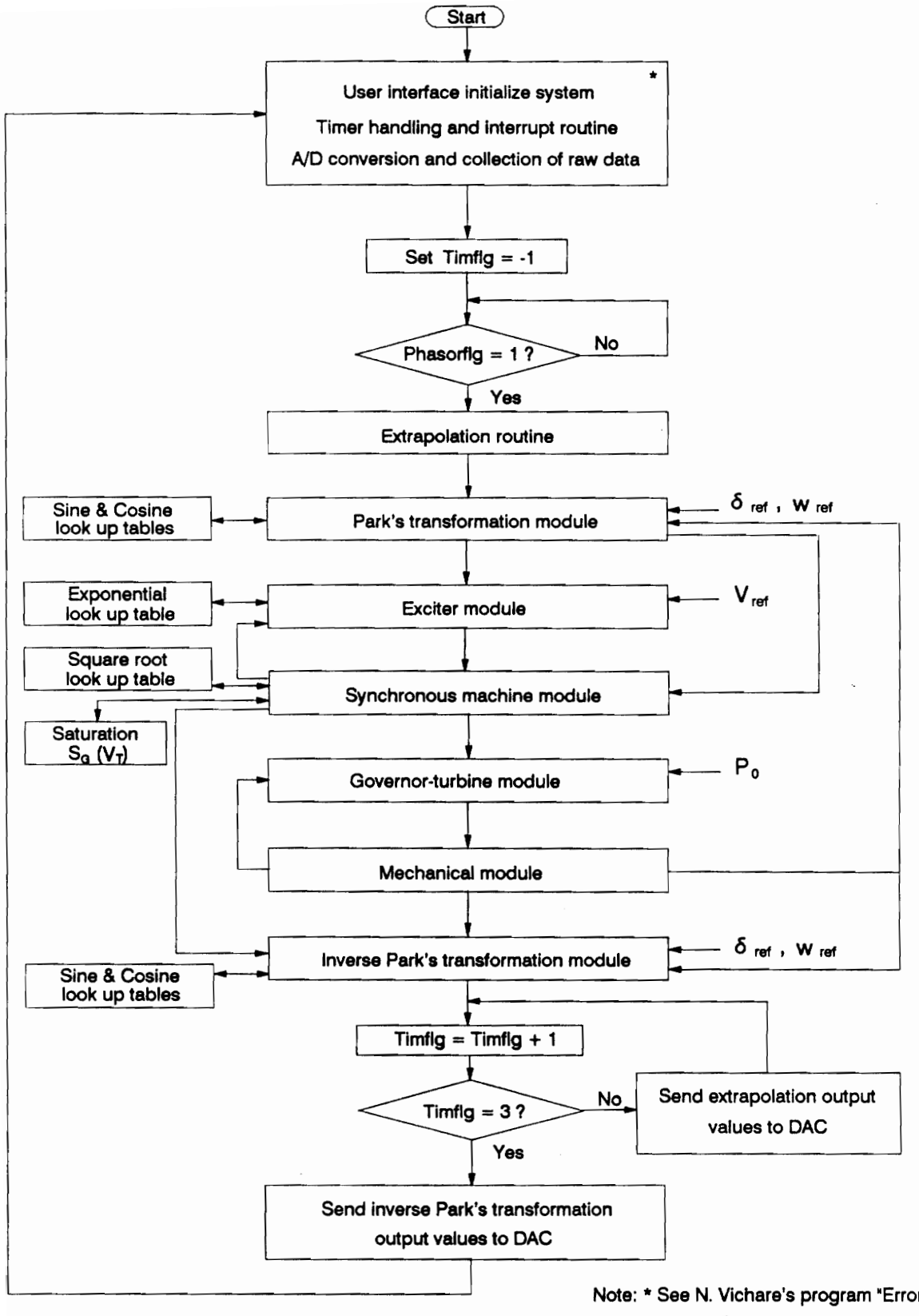



**Figure 31. Software organization**

All the digital calculation takes place in this module, and it is executed once every 1388.8 $\mu$s (720 Hz). The inputs to this module are:

1. The phase a, b and c current samples.
2. The reference point values, which include $V_{ref}$, $P_0$, $\omega_{ref}$ and $\delta_{ref}$. For this simulation, in the main module, set

$$V_{ref} = 1.04\ p.u. \qquad P_0 = 1.09\ p.u.$$
$$\delta_{ref} = 0.4445\ rad. \qquad \omega_{ref} = 377\ rad./sec.$$

The outputs of this module are balanced three phase voltages which represent the output of the generator. The algorithms used in this model are given in complete detail earlier in Chapter III and IV, respectively.

As shown in Figure 31, the interrupts occur at regular intervals at the rate of 2880 Hz. They are set as 0, 1, 2 and 3 respectively in one calculation time frame which spans over four interrupts giving a sampling rate of 720 Hz, i.e. four output points would be given in one time interval. In each time step, as #3 is serviced (i.e. TIMGLG = 3 [5]), a control flag 'Phasorflg' is set up enabling the main simulation module to be executed. If Phasorflg = 1, the main program 'generator model' is started to be executed, and if Phasorflg = 0, the main program is held at the beginning point. During the real-time simulation, Phasorflg = 1 occurs at exactly the sampling point for each time interval (1388.8 $\mu$s). The main module starts from memory location $(8000)_{16}$. Subroutine EXTRAPOLATE is executed first, it extrapolates three points of $V_a$, $V_b$ and $V_c$ in a 60 Hz sine wave, where the input is two points measured at the rate of 720 Hz and the output is three points calculated at 2880 Hz, and then saves them in consecutive memory locations starting at VEXTRA (See program list in Appendix B). In order to compensate the 10° the analog interface [6], subroutine EXTRAINPUT is called to extrapolate the sample inputs $i_a$, $i_b$ and $i_c$, which are located in reserved

---

[5] See program 'Error Model Generator' in Appendix B in detail.

[6] Analog interface will be discussed in Section 5.3.

storage space IA, IB and IC (See Appendix B). Next the subprogram PT transfers the calculated inputs, which named $i_{a10}$, $i_{b10}$ and $i_{c10}$, from phase a-b-c domain to $i_d$ and $i_q$ in d-q domain by Park's transformation. Then module ES estimates the field voltage $E_{fd}$, and SMS (which includes two parts, d-axis and q-axis models) calculates the armature flux linkage $\psi$ and the terminal voltage of the machine, $V_T$. And then subprogram GTS is run to find mechanical power $P_m$. The swing equation, MS, is then solved. Finally, IPT transfers $V_T$ to phase values $V_a$, $V_b$ and $V_c$ by inverse Park's transformation and stores them in consecutive memory locations starting at VABCNEW (Appendix B). The procedure of execution for the whole simulation model can be seen in Figure 31.

**The Interrups and I/O Module:**[7] : The module (which is named 'Error Model Generator' and listed in Appendix B) consists of the main program, A/D conversion (ADC) and collection of raw data subroutine, phasor calculation subroutine and timer handling routine, which may generate the rate of 2880 Hz for D/A conversion (DAC), but call ADC and phasor calculation at the rate of 720 Hz. Timer handling routine is also used for waiting cycles. First the I/O module performs the initialization of all the peripherals. The interrupt module services interrupt numbers 0, 1, 2 and 3. They output the values of $V_a$, $V_b$ and $V_c$ calculated by IPT or EXTRAPOLATE to the right D/A channels. They also load the starting address of the next service routine into the level 5 interrupt vector. This ensures that they are cyclically executed. As shown in Figure 31, the flag 'Phasorflg' is set up while servicing #3 and this enables to start a calculation cycle for the main module. Actually, the author did not work on developing the interrupt and I/O module, but only called the program 'Error Model Generator' as a subroutine to implement real-time simulation at the sampling rate of 720 Hz [8].

---

[7] This module is programmed by Nitin Vichare, Research Assistant at Power System Lab, VPI.

[8] See Appendix B and Ref. [3 & 19] for further information on interrupt, timer handling, ADC & DAC and I/O routines.

## 5.3 Analog Interface

The microcomputer generated sinusoidal waveform at the output of the D/A converter is as shown in Figure 32. Because the samples of the waveform are output at 2880 Hz, a 60 Hz sine wave would have 48 sample points in each cycle. A zero order hold process is in effect between the samples. Based on the analysis of FFT, the waveform is with a significant frequency component at 2880 Hz, which can be seen on the digital storage oscilloscope. A two-stage RC filter with a cut-off frequency of 382 Hz was used to remove those higher frequency component. The transfer function of a two-stage RC filter is given by:

$$H(j\omega) = \frac{1}{1 + j\omega(R_1C_1 + R_2C_2 + R_1C_2) - \omega^2(R_1C_1R_2C_2)} \tag{5.1}$$

Figure 33 shows a two-stage RC filter circuit with this transfer function and a cut-off frequency of 382 Hz. The bode plots of $H(j\omega)$ are given in Figure 34. As can be seen, the cut-off frequency is indeed 382 Hz, and the phase lag at the fundamental power frequency (60 Hz) is about 10°.

Based on Figure 1, we know that the input signals $i_a$, $i_b$ and $i_c$ depends on the machine terminal voltage and the load, which are represented as:

$$\dot{i}_{ph} = \dot{v}_{ph} / Z_{load}$$

where $Z_{load} = R_{load} + jX_{load}$. It is obvious that the phase shift of output voltage should cause the same phase lag for input current. In order to compensate the phase shift, a extrapolating calculation for input $i_a$, $i_b$ and $i_c$ is asked before performing Park's transformation. Taking phase A as an example, at the starting point of simulation cycle k, we can read the input $i_{ak}$ from I/O module, and then use $i_{ak}$ and the previous input $i_{a(k-1)}$ to preform extrapolation. Assume

$$i_{a(k-1)} = I\sin(\phi) \tag{5.2}$$

$$i_{ak} = I\sin(\phi + 30°) \tag{5.3}$$

Now we may calculate the input current with considering the compensation of the 10° phase lag by

$$\begin{aligned} i_{ak10} &= I\sin(\phi + 40°) \\ &= I\sin\phi\cos 40° + I\cos\phi\sin 40° \\ &= i_{a(k-1)}(\cos 40° - \sqrt{3}\sin 40°) + isunak(2\sin 40°) \\ &\simeq -0.3473 \times i_{a(k-1)} + 1.2856 \times i_{ak} \end{aligned} \tag{5.4}$$

In this work, the extrapolation algorithm is implemented by subroutine 'EXTRAINPUT' ( See Appendix B ), and then $i_{ak10}$, $i_{bk10}$ and $i_{ck10}$ are regarded as the input of Park's transformation.

## *5.4 Computational Workload*

At each step, the generator model performs:

- Park's transformation on incoming sample data $i_a$, $i_b$ and $i_c$.
- Recursive update of state equations for exciter, generator, governor- turbine and mechanical modules.
- Inverse Park's transformation, extrapolation and interpolation on output voltage $V_T$.

The arithmetic work required to perform each of these operations is shown in Table 6. In addition to the arithmetic operations, enough time margin must remain in the processing window to allow for any processing overhead, such as data movement and shift, logical operation, A/D conversions, interrupts and timer handling, scaling of input and output quantities and data collection.

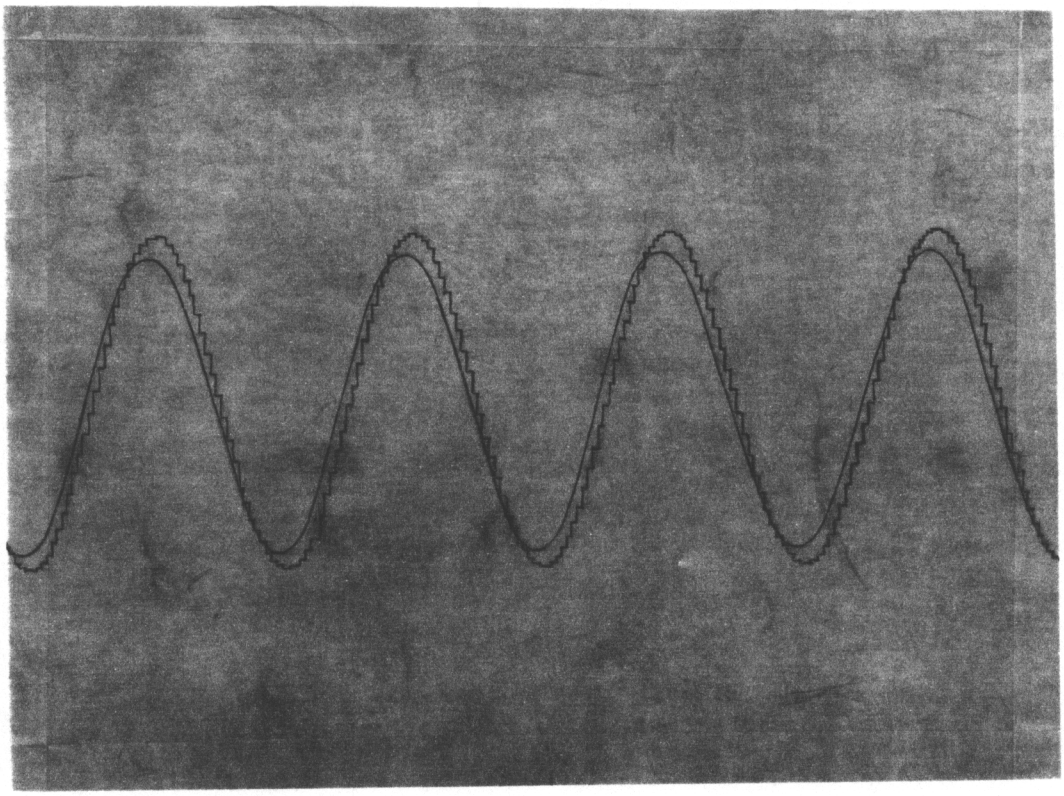


**Figure 32. Pre and post filtered waveforms**

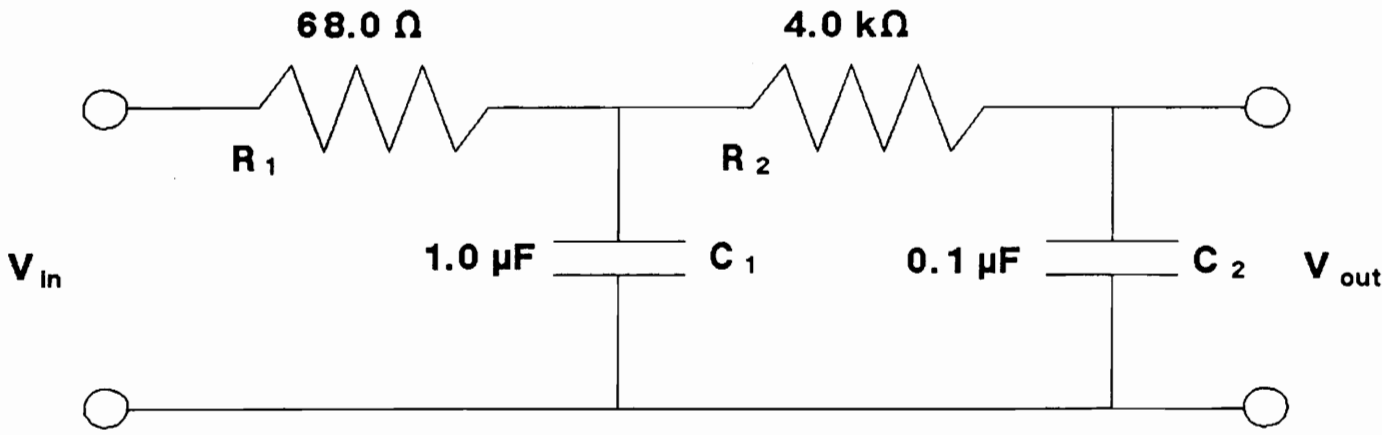



Figure 33. Two-stage RC filter with a cut-off frequency of 382 Hz

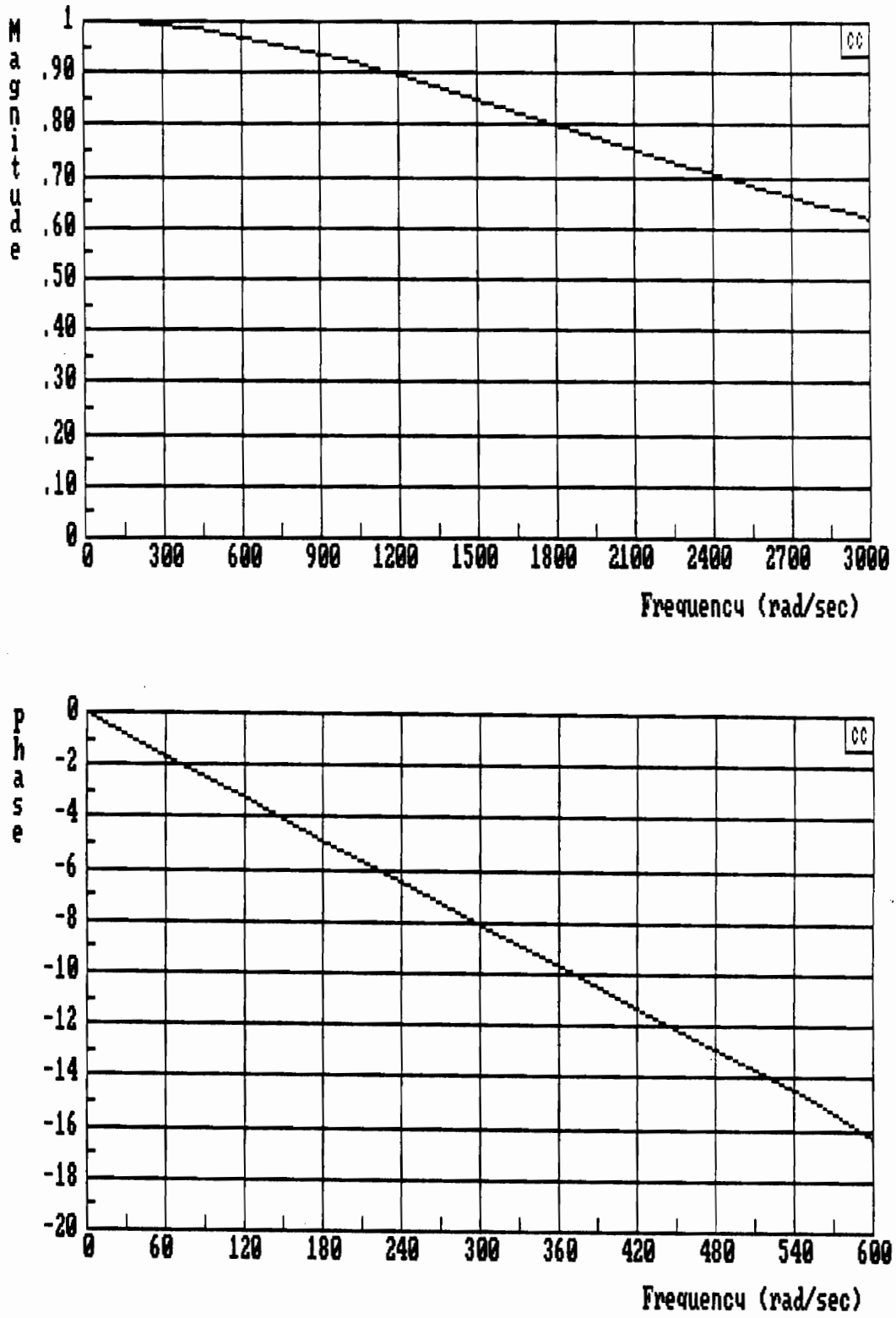



Figure 34. Bode plots of transfer function H(jω)

The prototype simulation model is designed to run at a sampling rate of 720 Hz, i.e. the processing window during which all of the above operations must be carried out is 1388.8 $\mu$s long. Given currently available microprocessor technology, it is possible to implement the model generator by using a commercial microprocessor. An excellent performance 32-bit microprocessor, Motorola 68020 is well supported in Virginia Tech's Power System Laboratory. Approximate operation times, in clock cycle, are listed in Table 7 [6, 13, 15 and 18] for two possible system configurations.

Based on Table 6 and Table 7, the minimum time needed to perform the arithmetic computations for one time interval of the generator model can be computed. This minimum time for each configuration is given in Table 8. However, for an additional 60 % added to the processing time by the overhead, the floating configuration will not be able to meet the 1388.8 $\mu$s time limit. Since the Motorola MC68020 is to be used, the generator model has to be implemented in fixed point arithmetic.

## *5.5 Simulation Results*

The system being simulated is that shown in Figures 2 and 30, the generator terminal phase voltages are amplified by the power amplifiers and fed to the external load models, and then the load currents $i_a$, $i_b$ and $i_c$, as input signals, are sent through three signal conditioners back to the machine. This section reports the results of validation simulation for the model generator with the external load $Z_l = 4.98 + j0.0$ p.u..

As described in sections 4.2, 5.2 and 5.3, a phase of the computer generated voltage waveform (48 samples/cycle) and its filtered version is shown in Figure 32. The time delay between the two waveforms is due to the zero order hold and filter delay. The three phase output voltage waveforms generated by the microprocessor are shown in Figure 35.

Table 6. Arithmetic Work Per Timestep

| Function | Add | | Subtract | | Multiply | Divide | Sin/Cos |
|---|---|---|---|---|---|---|---|
| Size | Word/Byte | Long | Word/Byte | Long | - | - | - |
| Extrapola-tion | 3* | | | | 6* | | |
| Park's Transform | 5 | 3 | 5 | 1 | 11 | | 4 |
| Recursions | 53 | 32 | 25 | 6 | 119 | 3 | |
| Inverse Park's transform | 3 | 6 | 9 | 4 | 2 | | 16 |
| Extrapola-tion | 9* | | | | 18* | | |
| Total | 73 | 41 | 39 | 11 | 156 | 3 | 20 |

Note: * MC68881 Floating-point arithmetic

**Table 7. Arithmetic Operation Times**

| System | Add/Sub | Multiply | Divide | Sin/Cos |
|---|---|---|---|---|
| MC68881 Floating-Point | 72 (S) | 92 (S) | 124 (S) | 410 (S) |
| Coprocessor * | 76 (X) | 96 (X) | | |
| Fixed Point Arithmetic | 4 (B/W)<br>6 (L) | 70 | 158 | 138** |

Notes:
* S -- Single precision real
X -- Extended precision real
** Assume table look up

**Table 8. Arithmetic execution times**

| System | Cycles | Microseconds ($\mu$s) | |
|---|---|---|---|
| | | 20 MHz | 25 MHz |
| MC68881 Floating-Point Coprocessor | 34732 * | 1736.6 | 1389.3 |
| Fixed Point Arithmetic ** | 16402 | 820.1 | 656.1 |

Note: * The arithmetic operation times are based on single precision real (S).

** Subroutine 'EXTRAPOLATE' and 'EXTRAINPUT' are implemented on MC68881 floating-point coprocessor, and the arithmetic operation times are based on extended precision real (X).

An easy way of evaluating the real-time simulation is to compare it with off-line computations for a known set of inputs [3].

### 5.5.1 Open-loop Test

A reference of 1 p.u. is set at the reference input and then the simulation started. The value of the terminal voltage $V_T$ is output to a D/A converter and the trace may shown on a scope. This is compared with the same test conducted off-line on the simulation model. The output results are shown in Figure 36. The trace (a) is the trajectory calculated off-line at a sampling rate of 720 Hz in the microcomputer. The other trace (b) is the real-time solution as calculated by using fixed-point arithmetic on the microprocessor and output through a D/A converter. The slight difference in trajectories are due to the truncation and quantization errors.

### 5.5.2 Closed-loop Test

The value of the machine terminal voltage $V_T$ calculated by the program is fed to the exciter model, which is compared with the reference set point $V_{ref}$, and this error signal is as the input of this simulation test. The reference voltage is set to 1 p.u.. Similarly, the simulation results are compared with the off-line simulation as shown in Figure 37. A 720 Hz sampling rate was used to carry out both real-time and off-line simulation results.

### 5.5.3 Constant Voltage Test

To verify the governing action and the equilibrium analysis, the synchronous machine electrical dynamics were replaced by a constant voltage magnitude. For this test, set

$$V_T = 1.05\ p.u. \qquad P_0 = 1.05\ p.u.$$

Under these conditions, the dynamics in this model are the mechanical dynamics represented by the swing equation (2.43) and the governing action described in section 2.3. Recall that the equilibrium angle for this system is $\delta_{eq} = 0.4445$ radians.

The results of the electrical power $P_e$ versus time, the mechanical power $P_m$ versus time, the rotor speed oscillation $\Delta\omega$ versus time and the torque angle $\delta$ versus time are shown in Figure 38.

The resulting equilibrium angles did not match the value predicted in section 2.4. The reason for this error is that the electrical power delivered by the model generator to the external system is not constant. The power function is only being sampled at 720 Hz, as a result, much of the information content of the power function is lost. This loss of information by the model generator causes the errors in the equilibrium angle.

### 5.5.4 Exciter Subsystem Tests

The exciter subsystem functions as a closed loop system in the model generator. It senses the changes in the machine terminal voltage $V_T$ due to external disturbances and performs a corrective action upon the error with the reference. The exciter is also tested for both real-time and off-line simulations under open-loop and closed-loop conditions. For open-loop test, a reference of 1 p.u. is set at the reference input and the simulation begun, and for closed-loop test, the reference is also set to 1 p.u., the terminal voltage is negative feedback from the the synchronous subsystem to this subsystem. The real-time simulation results are again compared with the off-line simulation as shown in Figures 39 and 40. A 720 Hz sampling rate was used for both real-time and off-line simulations. The difference in trajectories is due to the truncation and quantization errors.

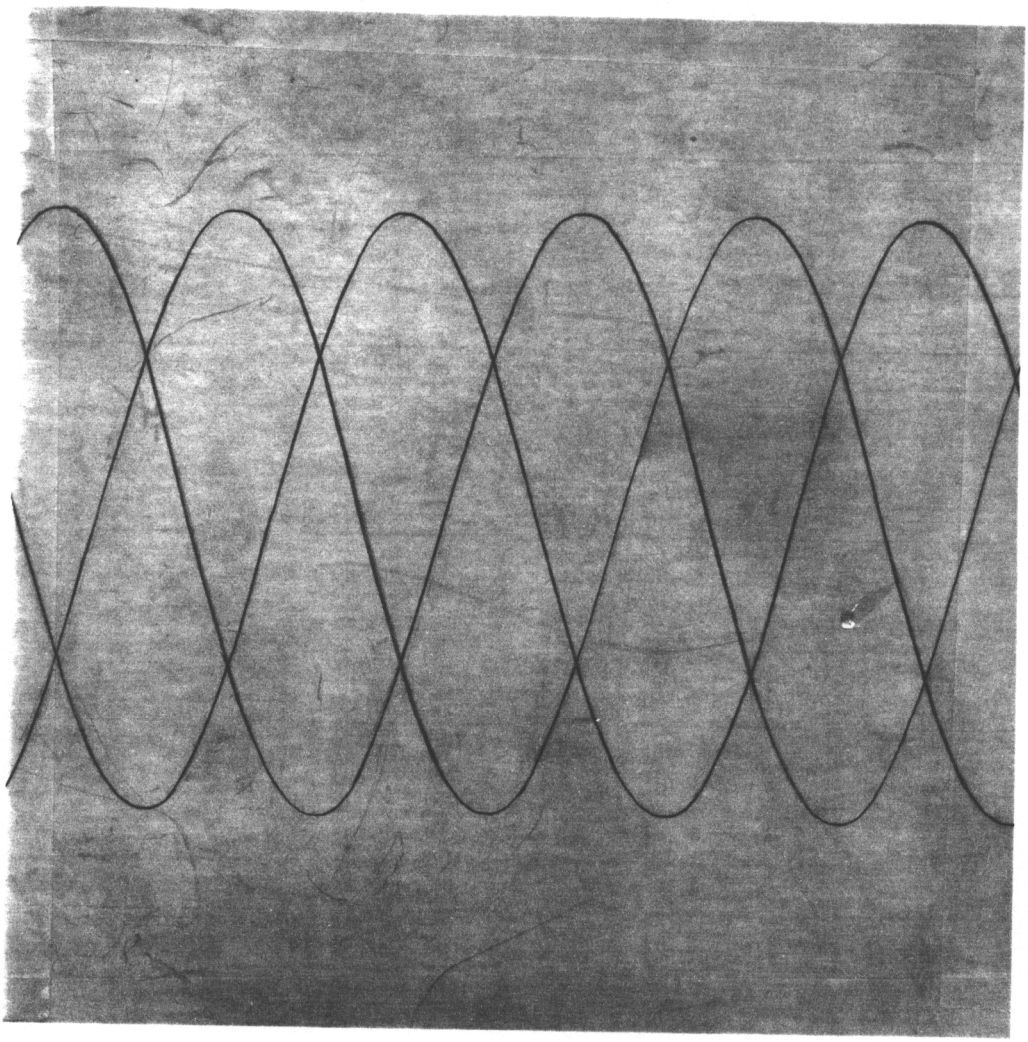
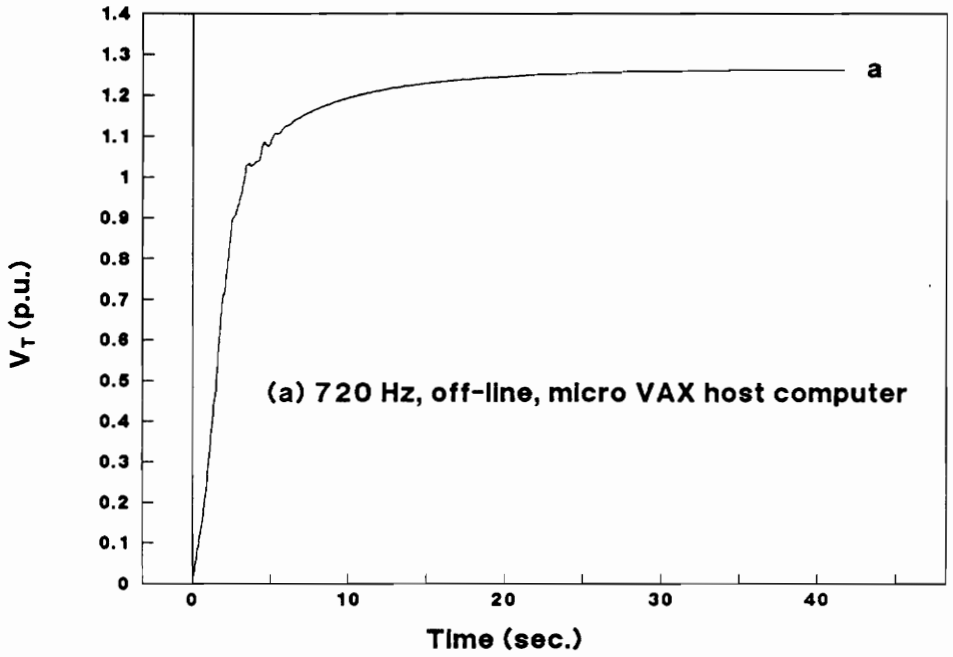


Figure 35. Microprocessor generated three phase voltage waveforms



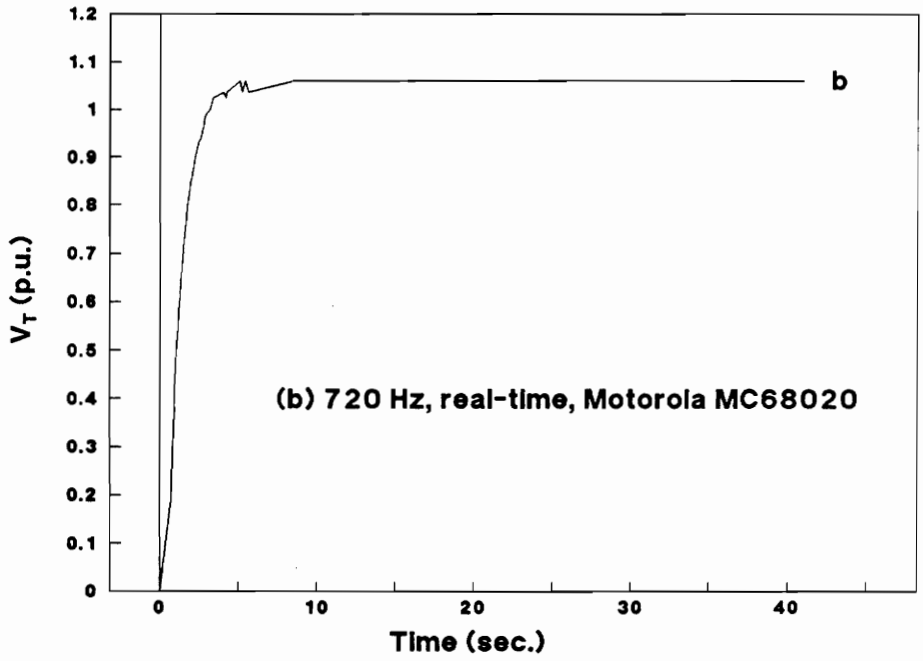




Figure 36. Trajectory of $V_T$ for unit step input (open-loop)

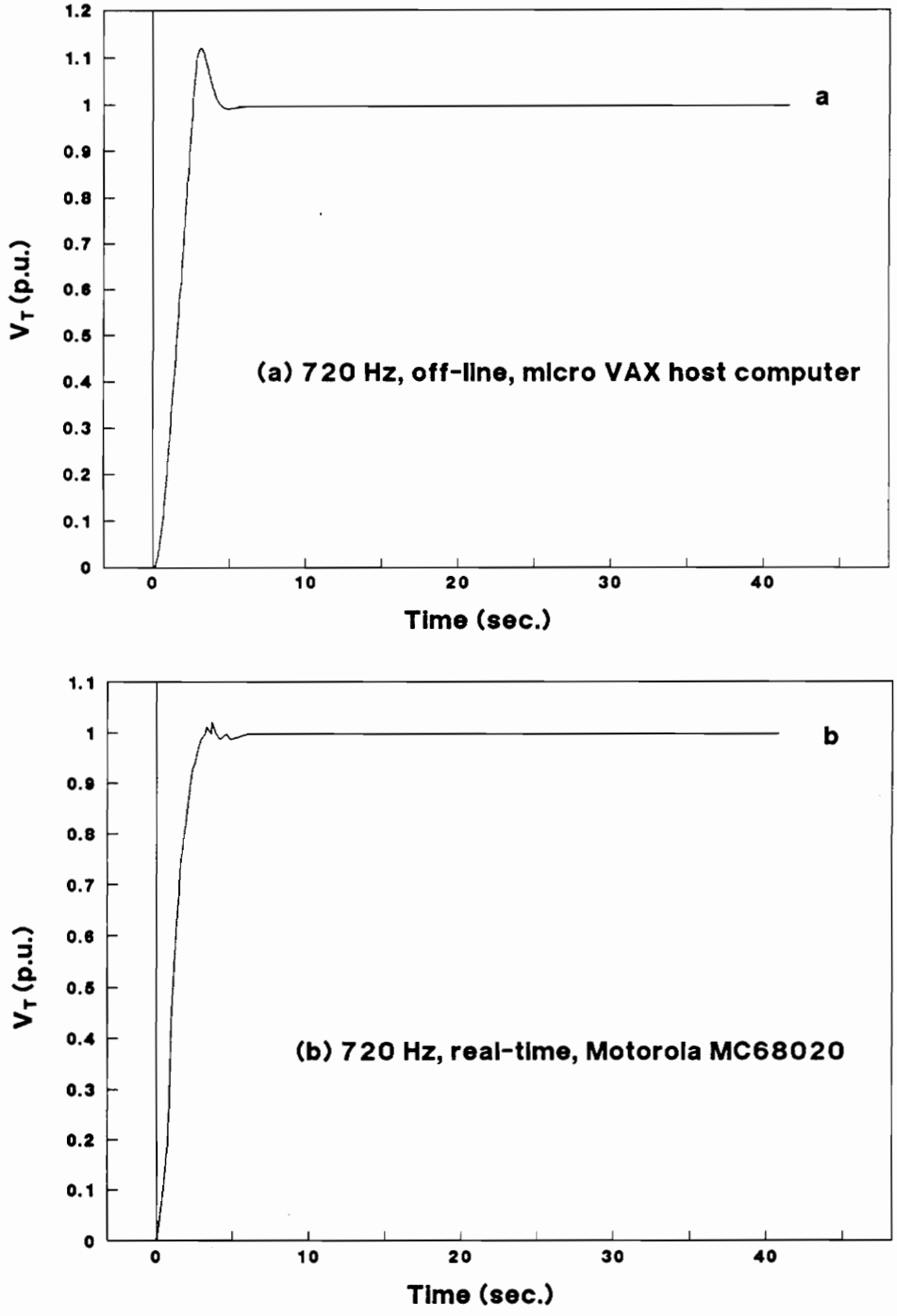



Figure 37. Trajectory of $V_T$ for unit step input (closed-loop)

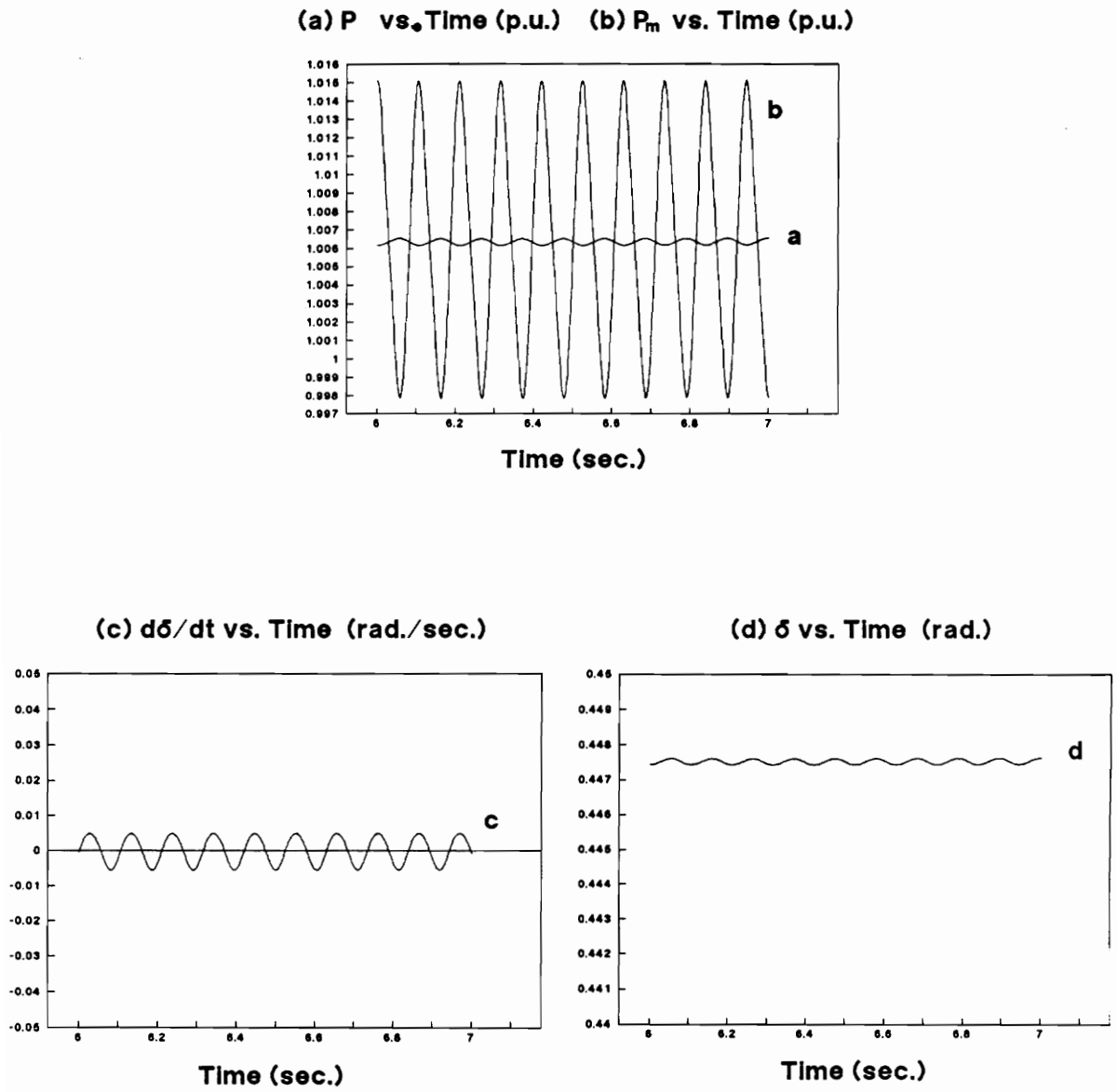
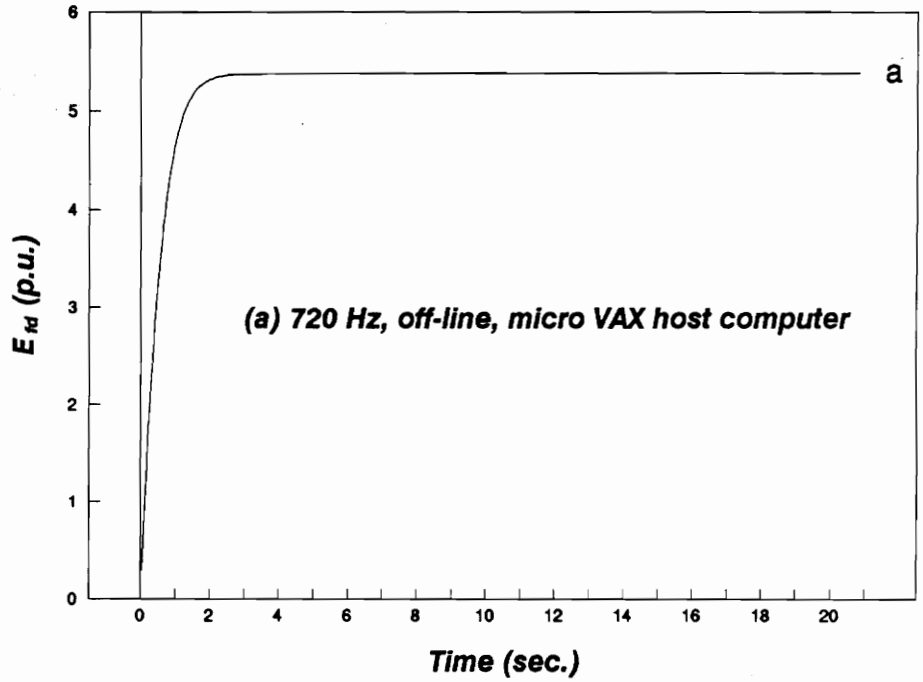



Figure 38. Plots for constant voltage test



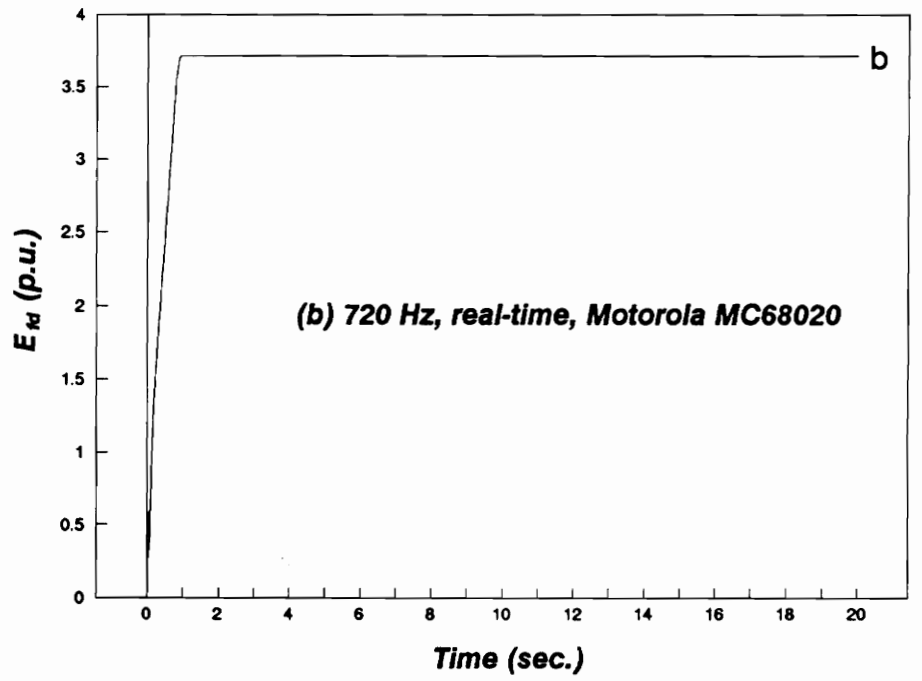




Figure 39. Trajectory of $E_{FD}$ for unit step input (open-loop)

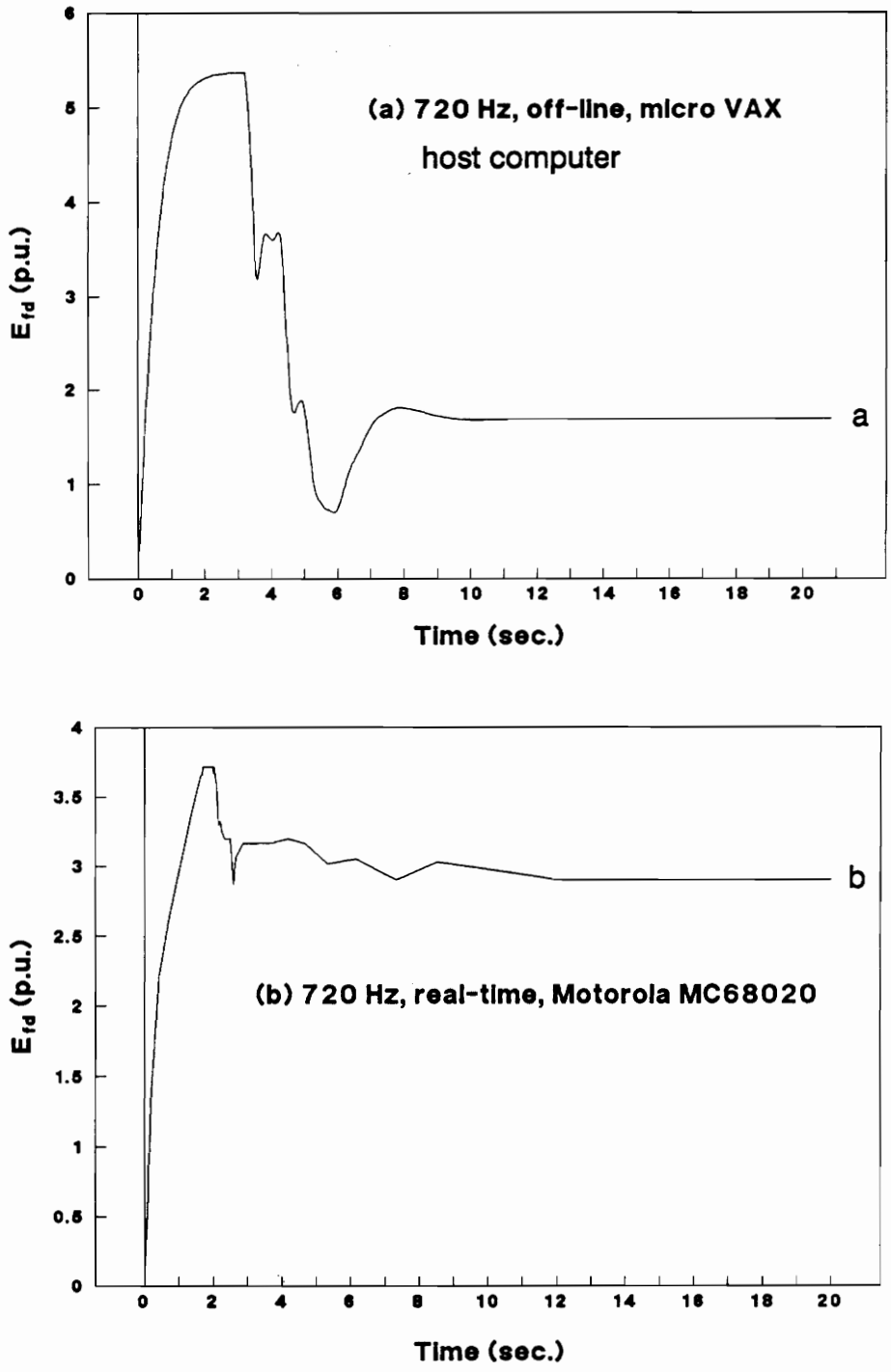



Figure 40. Trajectory of $E_{FD}$ for unit step input (closed-loop)

# Chapter VI

# Conclusions and Suggestions for Future Research

## *6.1 Conclusions*

A real-time digital simulation of the generator model on Motorola MC68020 was developed in this thesis. This could offer much more flexibility than the TNA, and a much more comprehensive and accurate representation than the existing generator simulation model.

The following conclusions and observations may be written from the implementation of the real-time digital simulation scheme:

1. The trapezoidal integration algorithm, i.e. second order Runge-Kutta numerical method, was used in solving the difference equations of the simulation system. Due to real-time digital simulation in fixed point arithmetic, the solution trajectories and values are slightly different from those of the off-line computer simulation results as explained in Chapter V.

2. A new direct and quadrature axis dynamic model for simulating solid iron rotor generators was implemented on Motorola MC68020 in real-time. Techniques for obtaining the coefficients of the model in terms of a quite detailed model of the machine impedance are presented.
3. The IEEE Type-I excitation system was modeled on Motorola MC68020 in real-time. Reference is made to previous studies which have used this model.
4. Since the machine swing equation and prime mechanical input power are modeled, a variable frequency voltage source was developed. This is a significant improvement over the constant frequency, voltage behind reactance presentation of a synchronous machine.
5. Park's and inverse Park's transformation were programmed by means of two functional look-up tables on Motorola MC68020. Due to the extrapolation for three points between two sampling points outputted at the rate of 720 Hz, the three phase voltages are actually outputted at 2880 Hz.
6. Minimum amount of external hardware is used as all the modeling is done in software.

## *6.2 Suggestions for Further Research*

This thesis attempted to show that this real-time digital simulation of generator model is really technically and economically feasible, and considerably comprehensive. As far as we know, this is probably the first time that anyone has implemented such a comprehensive simulation for the generator model in real-time on a microprocessor. However, a lot of improvements may be done to perfect this simulation scheme. Some suggestions can be made as follows:

1. The off-line simulation results are correct, while the real-time work couldn't confirm the input signals $i_a$, $i_b$ and $i_c$ from signal conditioner. Further test is needed in signal conditioner, A/D conversion and Park's transformation model.

2. Most likely the error from real-time simulation is due to signal conditioner and excitation subsystem coded by word size (16 bits). The exciter model is particularly important since it forms one of the control inputs to the synchronous machine. An improved excitation model with higher precision (long word size) is required for further detailed real-time simulation.
3. Implement a more complete turbine model, such as the tandem compound or the cross compound double reheat turbine model.
4. Add in the boiler subsystem, since the boiler and turbine are closely coupled and are often considered as a single subsystem.
5. Simulate the exciter by the IEEE Type-III excitation system. In fact, some static systems cannot be represented by Type-I, because the generator terminal current is used with potential as the excitation source.
6. Increase the precision of numerical computations. This can be done with a math coprocessor which has floating point capability and can be operated in a parallel fashion, so that some subsystems may be simulated by fixed point arithmetic, and some by floating point arithmetic, base on the accuracy requirements.[9] Of course, the arithmetic operation times should be considered first.

In summary, the suggestions for enhancing the performance of the simulator are possible and could be explored for further improvements in the computational speed, accuracy and memory requirements, while, from both hardware and software point of view, a thorough and comprehensive technical evaluation should be done to justify the additional effort.

---

[9] In this work, the subroutines which extrapolate the output voltages and the input currents have used the floating point arithmetic.

# References


[ 1] IEEE Committee Report, "Computer Representation of Excitation Systems", IEEE T-PAS, Vol PAS-87, June 1968

[ 2] Anderson, P.M., Fouad, A.A., "Power System Control and Stability", Vol 1, Iowa State University Press, Ames, Iowa

[ 3] Jayakumar, Anthony, "Real-time Digital Simulation of the Synchronous Machine System", Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1984

[ 4] Schulz, Richard P., Jones, William D., Ewart, Donald N., " Dynamic Models of Turbine Generators Derived from Solid Rotor Equivalent Circuits", IEEE T-PAS, May/June, 1973

[ 5] Olive, David W., " Digital Simulation of Synchronous Machine Transients", IEEE T-PAS, Aug., 1968

[ 6] MacMinn, Stephen Richard, "A Digital Model Generator for Real-time Simulation", Cornell University, Aug., 1984

[ 7] deMello, F.P., Hannett, L.N., "Validation of Synchronous Machine Models and Derivation of Model Parameters from Test", The IEEE Power Engineering Society, 1983 Winter Meeting, Feb. 2, 1983

[ 8] Salon, S. J., "Obtaining Synchronous Machine Parameters from Test", The IEEE Power Engineering Society, 1983 Winter Meeting, Feb. 2, 1983

[ 9] IEEE Committee Report, "Dynamic Models for Steam and Hydro Turbines in Power System Studies", IEEE T-PAS, Nov./Dec., 1973

[10] Stevenson, William D., Jr., " Elements of Power System Analysis", 4th Edition, McGraw-Hill Book Company, N.Y., 1982

[11] Cheney, Ward, Kincaid, David, "Numerical Mathematics and Computing", 2nd Edition, the University of Texas at Austin

[12] Kuo, Benjamin C., "Automatic Control Systems", 3rd Edition, University of Illinois at Urbana-Champaign

[13] Harman, Thomas L., Lawson, Barbara, " The Motorola MC 68000 Microprocessor Family: Assembly Language, Interface Design, and System Design", University of Houston, Clear Lake

[14] Gill, Arthur, Corwin, Edward, Logar, Antonette, "Assembly Language Programming for the 68000", Prentice-Hall, Inc., Englewood Cliffs, NJ

[15] "MC68020 32-Bit Microprocessor User's Manual", 2nd Edition, Prentice-Hall, Inc., Englewood, NJ

[16] "User Manual for DT1401 Series", Data Translation, Inc.

[17] "MC68901 Multi-Function Peripheral", Jan., 1984

[18] "MC68881 Floating Point Coprocessor User's Manual", 1st Edition, Motorola Inc. 1985

[19] "Power System Phasor Measurement Unit User's Manual", Virginia Polytechnic Institute and State University, 1989

[20] Phadke, A.G. "Introduction to Power System Transients", IEEE Tutorial Course on Digital Simulation of Electrical Transient Phenomena.

[21] Phadke, A.G., Thorp, J.S. "Computer Relaying for Power System", Research Studies Press LTD, Taunton, Somerset, England, 1988

# Appendix A

# Inductances and Time Constants of Machine

**d-Axis**

$$T'_{d0} = \frac{L_{ad} + L_{fd}}{r_{fd}}$$

$$T''_{d0} = \frac{L_{1d} + \dfrac{L_{ad}L_{fd}}{L_{fd} + L_{ad}}}{r_{1d}}$$

$$T'''_{d0} = \frac{L_{2d} + \dfrac{1}{\dfrac{1}{L_{ad}} + \dfrac{1}{L_{fd}} + \dfrac{1}{L_{2d}}}}{r_{2d}}$$

$$L_d = L_l + L_{ad}$$

$$L'_d = L_l + \frac{1}{\frac{1}{L_{ad}} + \frac{1}{L_{fd}}}$$

$$L''_d = L_l + \frac{1}{\frac{1}{L_{ad}} + \frac{1}{L_{fd}} + \frac{1}{L_{1d}}}$$

$$L'''_d = L_l + \frac{1}{\frac{1}{L_{ad}} + \frac{1}{L_{fd}} + \frac{1}{L_{1d}} + \frac{1}{L_{2d}}}$$

**q-Axis**

$$T'_{q0} = \frac{L_{aq} + L_{1q}}{r_{1q}}$$

$$T''_{q0} = \frac{L_{aq} + \frac{L_{1q}L_{2q}}{L_{1q} + L_{2q}}}{r_{2q}}$$

$$L_q = L_l + L_{aq}$$

$$L'_q = L_l + \frac{1}{\frac{1}{L_{aq}} + \frac{1}{L_{1q}}}$$

$$L''_q = L_l + \frac{1}{\frac{1}{L_{aq}} + \frac{1}{L_{1q}} + \frac{1}{L_{2q}}}$$

# Appendix B

# Program listings

```
*****************************************************************
*                                                               *
*   THE MAIN SIMULATION MODULE, WHICH INCLUDES THE EXCITER      *
*   SYSTEM, THE SYNCHRONOUS MACHINE SYSTEM, THE GOVERNOR-       *
*   TURBINE SYSTEM, THE MECHANICAL SYSTEM, THE PARK'S AND       *
*   INVERSE PARK'S TRANSFORMATION MODELS AND THE EXTRAPOLA-     *
*   TION SUBROUTINE.                                            *
*                                                               *
*****************************************************************
*
        TTL     'GENERATOR MODEL'
        LLEN    110
EXPTBL  EQU     $9100
SQRTBL  EQU     $A500
SINTBL  EQU     $C500
COSTBL  EQU     $D500
        XDEF    IDLE,VEXTRA,VABCNEW,VABC,VABCOLD
        XDEF    PSID1,PSIQ1,PSI,PSIL,FIRST,LAST
        XREF    VREAL,VIMAG
        XREF    SIGNALDATA,PHASORFLG,EXTRAPOLATE
*
        ORG     $8000
*
IDLE:   CMP.W   #1,PHASORFLG
        BNE     IDLE
        MOVE.W  #0,PHASORFLG
*
        JSR     EXTRAPOLATE
*
*       ***************************************
*       *         PARK'S TRANSFORMATION       *
*       ***************************************
*
        JSR     PARK
*
*       ***************************************
*       *          EXCITATION SUBSYSTEM       *
*       ***************************************
        MOVE    X2,D0    ;X2 IN D0
        MOVE    X3,D1    ;X3 IN D1
        MULS    #$A,D0   ;AKF/TE*X2 IN D0
        SUB     D1,D0    ;AKF/TE*X2-X3=FB
*                        ;  IN D0
        MOVE.L  #$6000,D2 ;VREF=1.04
        SUB.L   D0,D2     ;VREF-FB IN D2
        MOVE    VT,D3
        CMPI    #$40,D3
        BGT     SETIN
        MOVE    #$40,D3
SETIN   ASL.L   #8,D3
        SUB.L   D3,D2 ;UE=VREF-FB-VT
*
        MOVE.L  X1,D3     ;X1 IN D3
        MOVE    #$FFCE,D4 ;TAA=-50 IN D4
```

```
        MOVE.L  D3,D5
        ASR.L   #8,D3
        ASR.L   #1,D3
        MULS    D4,D3
        ASR.L   #1,D2
        ADD.L   D3,D2    ;DXK=A*X+UE IN D2
        MOVE    #$FEE4,D4 ;A*T IN D4
        MULS    D2,D4     ;DXK*A*T IN D4
        ASR.L   #8,D4
        ASR.L   #4,D4
        ASL     #1,D2     ;2*dxk in d2
        ADD     D2,D4     ;2*dxk+a*t*dxk
*                         ;in d4
        MULS    #$5B,D4   ;(d4)*t in d4
        ASR.L   #8,D4
        ADD.L   D4,D5     ;x1 in d5
*                        ;EQ.to CALL MEM
        MOVE.L  D5,X1     ;X1 BE STORED
        ASR.L   #6,D5
        MULS    #$4E20,D5 ;X4 = KA/TA*X1
        ASR.L   #2,D5     ;  IN D5
        ASR.L   #8,D5
        MOVE.L  D5,D6     ;X4 IN D6
        CMPI.L  #$74D,D6
        BGT     LOOP      ;X4 > 7.3
        CMPI    #$F8B3,D6 ;GOTO LOOP
        BLT     LOOP1     ;X4 < -7.3
        BRA     LOOP2     ;GOTO LOOP1
*
LOOP    MOVE    #$74D,D6  ;X4=7.3
        BRA     LOOP2
*
LOOP1   MOVE    #$F8B3,D6 ;X4=-7.3
        BRA     LOOP2
*
LOOP2   MOVE    D6,D2     ;X4 IN D2
        MOVE    D6,X4
        MOVE    X2,D3     ;X2 IN D3
        MOVE    #$FEC0,D4 ;TEE=-1.25
        CLR     D5        ;IN D4
        JSR     MEM1      ;X2 IN D5
        MULS    #$B1,D5   ;AFTER CALL
        SUB.B   #$7F,D5
        CMPI.B  #0,D5
        BLE.B   EXC
        ADD     #$100,D5
EXC     ASR.L   #8,D5
        MULS    #1000,D5
        SUB.B   #$7F,D5
        CMPI.B  #0,D5
        BLE.B   EXC1
        ADD     #$100,D5
EXC1    ASR.L   #8,D5      ;D5 IS INTEGER
        AND.L   #$FFFE,D5  ;D5 IS EVEN
        ADD     #$9100,D5
```

```
        MOVEA.L D5,A0
        MOVE    (A0),D5   ;EXP(Q/TE*X2K)
        MULS    #$19,D5   ;*P=X5K IN D5
        SUB.B   #$7F,D5
        CMPI.B  #0,D5
        BLE.B   EXC2
        ADD     #$100,D5
EXC2    ASR.L   #8,D5
*
        SUB     D5,D6     ;X4-X5=US IN D6
        MOVE    #$FEC0,D4 ;TEE IN D4
        MOVE    X2,D3     ;X2 IN D3
        CLR     D5
        MOVE    D6,D2     ;US IN D2
        JSR     MEM1      ;CALL MEM1
        MOVE    D5,X2     ;X2K BE SAVED
        MULS    #$140,D5  ;EFD=X2K/TE
        ASR.L   #8,D5     ;IN D5
        MOVE    D5,EFD
*
        MULS    #$7,D5    ;UF=EFD*AKF IN
        ASR.L   #8,D5     ;  D5
        MOVE    #$ff00,D4;TF IN D4
        MOVE    X3,D3
        MOVE    D5,D2     ;UF IN D2
        JSR     MEM       ;CALL MEM
*                         ;X3K IN D5
        MOVE    D5,X3     ;X3 BE SAVED
*       ********************************************
*       *          GENERATOR D--AXIS SUBSYSTEM     *
*       ********************************************
        MOVE    X61,D0    ;X61 IN D0
        MOVE    X62,D1    ;X62 IN D1
        MOVE    #$F7A4,D2;A21 IN D2
        MOVE    #$F833,D3;A22 IN D3
        MOVE    EFD,D5    ;EFD IN D5
        NEG     D5
*
        JSR     MULM1     ;CALL MULM1
        MOVE    D0,X61    ;X61 BE SAVED
        MOVE    D1,X62    ;X62 BE SAVED
        MULS    #$8A,D0 ;.539*X61 IN D0
        MULS    #$71,D1 ;.443*X62 IN D1
        MOVE    EFD,D6
        MULS    #$7,D6  ;.028*EFD IN D6
        ASR.L   #4,D6
        ADD.L   D0,D1
        ASR.L   #8,D1
        ASR     #2,D1
        ADD     D1,D6     ;Y1 IN D6
        MOVE    D6,A0     ;Y1 IN A0
*
        MOVE    X71,D0     ;X71 IN D0
        MOVE    X72,D1     ;X72 IN D1
        MOVE    #$F7A4,D2 ;A21 IN D2
```

```
        MOVE    #$F833,D3 ;A22 IN D3
        MOVE    AID1,D5
        MOVEA   D5,A1
        NEG     D5
*
        JSR     MULM       ;CALL MULM
        MOVE    D0,X71
        MOVE    D1,X72
        MOVE    A1,D2      ;ID1 IN D2
        MOVEA   D2,A3      ;ID1 IN A3
        MULS    #$BD,D2   ;.74*ID1 IN D2
        MOVEA   AID,A2    ;ID IN A2
        SUBA    A2,A1     ;ID1-ID IN A1
        MOVE    A1,D3     ;ID1-ID IN D3
        MULS    #$1FAD,D3;.044*(dID/dT)
*                         ;IN D3
        MULS    #$452,D0 ;4.32*X71 IN D0
        MULS    #$6AE,D1 ;6.68*X72 IN D1
        ADD.L   D0,D1
        ASR.L   #6,D1
        ADD.L   D3,D2
        SUB.L   D1,D2     ;Y2 IN D2
        ASR.L   #4,D2
        ADD     A0,D2     ;Y3=Y1+Y2 IN D2
*
        MOVE.L  X81,D0
        MOVE.L  X82,D1
        MOVE    X83,D3
        MOVE.L  D0,D4     ;MAKE COPY X8
        MOVE.L  D1,D5
        MOVE    D3,D6
        ASR.L   #8,D0
        ASR.L   #2,D0
        MULS    #$C5,D0   ;.77*X81 IN D0
        ASR.L   #8,D0
        ASR.L   #6,D1
        MULS    #$7EE,D1 ;7.93*X82 IN D1
        ASR.L   #8,D1
        MULS    #$81F,D3 ;8.12*X83 IN D3
        ASR.L   #8,D3
        ADD     D0,D1
        ADD     D1,D3
        ASL.L   #2,D2
        SUB     D3,D2     ;DXK(3,1) IN D2
*
        MOVE    D6,D0     ;X83 IN D0
        MULS    #$5B,D0   ;.001389*X83
        ASR.L   #8,D0
        ASR.L   #3,D0     ;T/2*X83 IN D0
        ADD.L   D5,D0     ;(X82+T/2*X83)
        ASR.L   #6,D0
        MULS    #$5B,D0
        ASR.L   #4,D0
        ADD.L   D4,D0     ;XK81 IN D0
*
```

```
        MOVE    D2,D7     ;COPY DXK(3,1)
        MULS    #$5B,D2   ; IN D7
        ASR.L   #8,D2
        ASR.L   #8,D2
        ASR     #1,D2
        ADD     D6,D2     ;DXK(2,1)+T/2
        MULS    #$5B,D2 ;*DXK(3,1) IN D2
        ASR.L   #8,D2
        ASR.L   #2,D2
        ADD.L   D5,D2     ;XK82 IN D2
        MULS    #$1FD,D7 ;2*DXK(3,1)
        ASR.L   #8,D7    ;+T*A33*DXK(3,1)
        ASR.L   #6,D5
        MULS    #$46,D5
        ASR.L   #8,D5     ;T*.77*DXK(1,1)
        ASR.L   #8,D5     ; IN D5
*
        MOVE    D6,D4    ;COPY X83 IN D4
        MULS    #$2D,D4 ;7.93*T*DXK(2,1)
        ASR.L   #8,D4     ; IN D4
        ASR.L   #4,D4
        SUB     D5,D7
        SUB     D4,D7
        MULS    #$5B,D7
        ASR.L   #8,D7
        ASR.L   #8,D7
        ADD     D6,D7      ;XK83 IN D7
        MOVE    A3,D1      ;AID1 IN D1
        MULS    #$39,D1    ;.225*AID1
        ASR.L   #2,D1
        MOVE.L  D0,X81
        MOVE.L  D2,X82
        MOVE    D7,X83
        ASR.L   #8,D0
*
        ASR.L   #2,D0
        MULS    #$85C,D0   ;8.36*XK81
        ASR.L   #8,D0
        ASR.L   #6,D2
        MULS    #$7CC,D2   ;7.8*XK82
        ASR.L   #8,D2
        ADD.L   D0,D2
        ADD     D7,D2
        SUB     D1,D2      ;PSID1 IN D2
        ASR     #6,D2
        MOVEA   D2,A0      ;PSID1 IN A0
        MOVE    D2,PSID1   ;SAVE PSID1
*
*       ********************************************
*       *          GENERATOR Q--AXIS SUBSYSTEM     *
*       ********************************************
*
        MOVE    AIQ1,D2
        MOVE    X9,D3
        MOVE    #$F954,D4 ;-6.67 IN D4
```

```
        MOVE    D2,D6    ;COPY IQ1 IN D6
        JSR     MEM2       ;CALL MEM2
        MOVE    D5,X9      ;X9 BE STORED
        MOVE    AIQ,D4
        MOVE    D6,D7     ;COPY IQ1 IN D7
        SUB     D4,D6      ;IQ1-IQ
        MULS    #$2D5B,D6 ;.063*(IQ1-IQ)
        ASR.L   #8,D6      ;/T
        MOVEA   D7,A4     ;COPY IQ1 IN A4
        MOVEA   D4,A5      ;AIQ IN A5
        MULS    #$428,D7  ;4.16*AIQ1
        ASR.L   #8,D7
*
        MULS    #$14,D5
        ASR.L   #7,D5
        ADD     D7,D6
        SUB     D5,D6       ;Y4 IN D6
        MOVE    D6,D5
        MOVE    X101,D0
        MOVE    X102,D1
        MOVE    #$FBCF,D2 ;A2(2,1)=-4.19
        MOVE    #$F717,D3 ;A2(2,2)=-8.91
*
        JSR     MULM        ;CALL MULM
        MOVE    D0,X101
        MOVE    D1,X102
        MOVE    A4,D2       ;IQ1 IN D2
        MULS    #$39,D2     ;.225*IQ1
        NEG.L   D2
        MULS    #$F8,D1     ;.97*X102
        ASR.L   #4,D1
        MULS    #$68A,D0    ;6.54*X101
        ASR.L   #4,D0
        ADD.l   D1,D0
        ASR.L   #2,D0
        ADD.L   D0,D2       ;PSIQ1 IN D2
        ASR.L   #8,D2
        MOVEA   D2,A1       ;PSIQ1 IN A1
        MOVE    D2,PSIQ1    ;SAVE PSIQ1
*
        JSR     $9000       ;CALL SATUR
        MOVE    DDEL,D3
        MULS    #$AE,D3
        ASR.L   #8,D3
        ASR.L   #8,D3
        ADD     #$1000,D3 ;(377+DDEL)
*                         ;/377 IN D3
        MOVE    PSID,D6   ;PSID IN D6
        MOVE    PSIQ,D7   ;PSIQ IN D7
        MOVE    A3,AID    ;ID1=ID
        MOVE    A4,AIQ    ;IQ1=IQ
        MOVE    D1,PSID   ;PSID1=PSID
        MOVE    D2,PSIQ   ;PSIQ1=PSIQ
        SUB     D1,D6     ;PSID-PSID1 IN D6
        SUB     D2,D7     ;PSIQ-PSIQ1 IN D7
```

```
        MULS    #$1E8E,D6 ;DIVIDED BY
        ASR.L   #8,D6     ;T*377
        ASR.L   #4,D6
        MULS    #$1E8E,D7
        ASR.L   #8,D7
        ASR.L   #4,D7
        MULS    D3,D1  ;WK*PSID1 IN D0
        ASR.L   #4,D1
        MULS    D3,D2  ;WK*PSIQ1 IN D2
        ASR.L   #4,D2
        MOVEA   D3,A6   ;WK IN A6
        MOVE    A3,D3   ;ID1 IN D3
        MOVE    A4,D4   ;IQ1 IN D4
        MULS    #$A,D3  ;Ra*ID1
        MULS    #$A,D4  ;Ra*IQ1
        SUB.L   D4,D1   ;WK*PSID1-Ra
        ASR.L   #8,D1   ;*IQ1 IN D1
        ADD.L   D3,D2   ;Ra*ID1+WK
        ASR.L   #8,D2   ;*PSIQ1 IN D2
        ADD     D1,D7   ;VQK1 IN D7
        SUB     D2,D6   ;VDK1 IN D6
*
        MOVE    D7,D5
        MOVE    D6,D4
        MULS    D5,D7   ;VQK1^2
        MULS    D4,D6   ;VDK1^2
        ADD.L   D6,D7   ;VQK1^2+VDK1^2
        ASR.L   #8,D7   ;IN D7
*
        MULS    #500,D7
        ASR.L   #8,D7     ;D7 IS INTEGER
        AND.L   #$FFFE,D7 ;D7 IS EVEN
        ADD     #$A500,D7
        MOVEA.L D7,A2
        MOVE    (A2),D7   ;VT IN D7
        MOVE    D7,VT     ;SAVE VT
*
*       ***********************************************
*       *          GOVERNOR & TURBINE SUBSYSTEM       *
*       ***********************************************
*
        MOVE    X111,D3
        MOVE.L  DDEL1,D2
*
        JSR     MEM3
        ADD     X111,D4  ;X111 IN D4
        MOVE    D4,X111  ;SAVE X111
*
        MOVE    X113,D3
        MOVE.L  DDEL1,D2
        SUB.L   DDEL,D2
        ASR.L   #8,D2
        ASR.L   #2,D2
        MULS    #$2D0,D2
        ASL.L   #8,D2
```

```
        ASL.L   #2,D2
        JSR     MEM3     ;(DDEL1-DDEL)/T
        ADD     X113,D4  ;= INPUT
        MOVE    D4,X113   ;SAVE X113
        ADD     X111,D4   ;(X111+X113)*K
        MULS    #$11DB,D4; /T1 = Y7
        ASR.L   #8,D4     ;K=50,  T1=2.8
        ASR.L   #3,D4
        MOVE    D4,DDEL3
*
        SUB     #$1150,D4 ;P0 = 1.09
        NEG     D4        ;P0-Y7 IN D4
        MULS    #$6AB,D4  ;(P0-Y7)/T3
        ASR.L   #8,D4
        MOVE    D4,D6     ;(Po-y7)/T3
*                         ; IN D4,D6
        ASR.L   #4,D4
        MOVE    D4,D2     ;INPUT IN D2
        MOVE    X112,D3
        MOVE    #$F955,D4 ;-1./T3 IN D4
*
        JSR     MEM1      ;CALL MEM1
        MOVE    D5,X112   ;SAVE X112
        MULS    #$6AB,D5  ;X112/T3
        ASR.L   #4,D5
        SUB     D6,D5     ;(Po-y7)/t3-
        NEG     D5        ;x112/T3=Y5
        ASR.L   #4,D5     ; IN D5
*
        CMPI    #$1A,D5   ;IF Y5 > .1
        BGT     LUP       ;GOTO LUP
*
        CMPI    #$FFE6,D5 ;IF Y5 < -.1
        BLT     LDOWN     ;GOTO LDOWN
        BRA     NOR
*
LUP     ASR.L   #4,D6
        SUB     #$1A,D6   ;(P0-y7)/t3-.1
        MULS    #$266,D6  ;(D6)*T3
        ASR.L   #8,D6
        ASR.L   #4,D6
        MOVE    D6,X112   ;SAVE X112
        BRA     NOR
*
LDOWN   ASR.L   #4,D6
        ADD     #$1A,D6   ;(P0-Y7)/T3+.1
        MULS    #$266,D6  ;(D6)*T3
        ASR.L   #8,D6
        ASR.L   #4,D6
        MOVE    D6,X112    ;SAVE X112
*
NOR     MOVE    X112,D2   ;X112 IN D2
        CMPI    #$133,D2  ;IF X112 > 1.2
        BGT     MAX       ;GOTO MAX
        CMPI    #$E6,D2   ;IF X112 < .9
```

```
        BLT     MIN         ;GOTO MIN
        BRA     MID        ;.9< X112 <1.2
*                           ; GOTO MID
MAX     MOVE    #$133,D2   ;SET X112=1.2
        BRA     MID
*
MIN     MOVE    #$E6,D2    ;SET X112 =.9
*
MID     MOVE    X12,D3
        MOVE    #$FD80,D4  ;-1/Tch IN D4
*
        JSR     MEM         ;CALL MEM
        MOVE    D5,X12     ;SAVE X12
        MULS    #$280,D5   ;Pmk1=X12/Tch
*
        MOVE.L  DDEL1,D2
        ASR.L   #8,D2
        MULS    #$AE,D2    ;DDEL1/377.
        ASR.L   #8,D2
        ASR.L   #8,D2
        ADD.L   #$10000,D2
        ASR.L   #4,D2
        DIVS    D2,D5    ;Pmk1/Wk1=Tmk1
        ASL.L   #4,D5
        AND.L   #$FFFFF,D5
        MOVE.L  D5,TM1       ;SAVE TM1
*
*       ****************************************
*       *          MECHANICAL SUBSYSTEM        *
*       ****************************************
*
        MOVE    VT,D2
        MULS    VT,D2      ;VT^2IN D2
        ASR.L   #8,D2
        MULS    #$2536,D2  ;VT^2/XDD IN
        ASR.L   #8,D2      ;D2
*
        MOVE.L  DEL,D3
        ASR.L   #8,D3
        ASR.L   #4,D3
        CMPI    #0,D3
        BGE     PE         ;IF DEL < 0
        ADDI    #$6488,D3  ;DEL + 2*PI
        BRA     PE1
*
PE      CMPI    #$6488,D3
        BLE     PE1        ;IF DEL>2*PI
        SUBI    #$6488,D3  ;DEL - 2*PI
*
PE1     CMPI    #$1922,D3  ;IF DEL>PI/2
        BGT     PE2        ;GOTO PE2
        JSR     SSIN     ;SIN(DEL) IN D3
        BRA     PEC
PE2     CMPI    #$3244,D3  ;IF DEL > PI
        BGT     PE3        ;GOTO PE3
```

```
        SUBI    #$3244,D3 ;PI-DEL IN D3
        NEG     D3
        JSR     SSIN    ;SIN(DEL) IN D3
        BRA     PEC
*
PE3     CMPI    #$4B66,D3 ;IF DEL>1.5*PI
        BGT     PE4       ;GOTO PE4
        SUBI    #$3244,D3 ;DEL-PI IN D3
        JSR     SSIN
        NEG     D3       ;SIN(DEL) IN D3
        BRA     PEC
*
PE4     SUBI    #$6488,D3
        NEG     D3     ;1.5*PI<DEL<2*PI
        JSR     SSIN
        NEG     D3      ;SIN(DEL) IN D3
*
PEC     MULS    D2,D3     ;Pe IN D3
        ASR.L   #4,D3     ;SET Pe = Te (P.U.)
        MOVE.L  D3,TE1    ; SAVE Te1
*
        SUB.L   D3,D5     ;Tm-Te IN D5
        ASR.L   #1,D5
        MULS    #$25B3,D5 ;(D5)*377/AJ (AJ=2H)
        MOVE.L  DDEL1,D3
        MOVE.L  D3,DDEL   ;DDEL1=DDEL
*
        CMP.L   #$150000,TIME
        BGT     SWING
        ADD.L   #$5B,TIME
        BRA     INV
*
SWING   ASR.L   #8,D3
        ASR.L   #2,D3
        MULS    #$FCFC,D3 ;DDEL*AJJ IN D3
        ASL.L   #1,D3     ;(AJJ=-D*377/2H=3.016)
        ADD.L   D3,D5     ;DDEL*AJJ+(D5) IN D5
        ASR.L   #8,D5
        ASR.L   #1,D5
*
        MOVE    #$FF6A,D4 ;AJJ*.001389
        MULS    D5,D4     ;*(D5) IN D4
        ASR.L   #8,D4
        ASL.L   #5,D5     ;2*(D5) IN D5
        ADD.L   D5,D4     ;2*(D5)+(D4)
        ASR.L   #4,D4
        MULS    #$5B,D4
        ASR.L   #7,D4     ;(D4)*T/2
        ADD.L   DDEL1,D4  ;DDEL1 IN D4
        MOVE.L  D4,DDEL1  ;SAVE DDEL1
        ASR.L   #8,D4
        ASR.L   #1,D4
        MULS    #$5B,D4   ;T*DDEL1
        ASR.L   #7,D4
        ADD.L   DEL,D4    ;TORQUE ANGLE IN D4
```

```
        MOVE.L  D4,DEL     ;SAVE THE ANGLE IN DEL
*
*       ************************************************
*       *          INVERSE PARK TRANSFORMATION         *
*       ************************************************
*
INV     MOVE.L  DDEL1,D6
        ASR.L   #8,D6
        ASR.L   #1,D6
        MULS    #$5B,D6   ;DDEL1*T IN D6
        ASR.L   #7,D6
        ADD.L   #$860A91,D6 ;WT*T
*
        MOVE.L  DEL,D5
        ASL.L   #1,D5          ;2*DEL IN D5
        MOVE.L  FI,D4
        ADD.L   D5,D4          ;2*DEL + FI
*                              ; IN D4
        MOVE.L  #VABCNEW,A2
*
        ADD.L   D6,D4    ;2*DEL+FI+W*T
        MOVE.L  D4,D1    ;IN D4 & D1
        MOVE.L  D4,FI
        ASR.L   #8,D1
        ASR.L   #4,D1
*
        CMPI    #0,D1
        BGE     FGE            ;IF FI < 0
        ADDI    #$6488,D1      ;FI+2*PI
        BRA     FGE1
*
FGE     CMPI    #$6488,D1
        BLE     FGE1           ;IF FI > 0
        SUBI    #$6488,D1      ;FI-2*PI
        SUB.L   #$6487ED5,FI   ;SAVE FI
*
FGE1    CMPI    #$1922,D1   ;IF FI > PI
        BGT     FGE2         ;GOTO FGE2
        JSR     SSIN1      ;SIN(FI) IN D3
        BRA     IPARKT     ;COS(FI) IN D5
*
FGE2    CMPI    #$3244,D1   ;IF FI > PI
        BGT     FGE3         ; GOTO FGE3
        SUBI    #$3244,D1;
        NEG     D1           ;PI-FI IN D1
        JSR     SSIN1      ;SIN(FI) IN D3
        NEG     D5         ;COS(FI) IN D5
        BRA     IPARKT
FGE3    CMPI    #$4B66,D1 ;IF FI>1.5*PI
        BGT     FGE4       ; FI-PI
        SUBI    #$3244,D1
        JSR     SSIN1
        NEG     D3         ;SIN(FI) IN D3
        NEG     D5         ;COS(FI) IN D5
        BRA     IPARKT
```

```
*
FGE4     SUBI     #$6488,D1;1.5*PI<FI<2*PI
         NEG      D1
         JSR      SSIN1      ;COS(FI) IN D5
         NEG      D3         ;SIN(FI) IN D3
*
IPARKT   MULS     VT,D3      ;Va=VT*SIN(FI)
         ASR.L    #6,D3
         MOVE.W   D3,(A2)+   ;SAVE VaNEW
         ASR      #2,D3
*
         MULS     #$DDB,D5   ;.866*COS(FI)
         ASR.L    #8,D5
         MULS     VT,D5    ;.866*VT*COS(FI)
         ASR.L    #8,D5    ; IN D5
         ASR.L    #4,D5
         ASR      #1,D3    ;Va/2 IN D3
         ADD      D3,D5
         NEG      D5        ;Vb IN D5
         ASL      #2,D5
         MOVE.W   D5,(A2)+ ;SAVE VbNEW
*
         ASL      #1,D3
         ASR      #2,D5
         ADD      D3,D5
         NEG      D5        ;Vc IN D5
         ASL      #2,D5
         MOVE.W   D5,(A2)+ ;SAVE VcNEW
*
         MOVE.L   DEL,D6
         ASL.L    #1,D6
         SUB.L    D6,FI
*
         MOVE.L   #VABC,A2
         MOVE.L   #VABCOLD,A3
         MOVE     (A2)+,(A3)+ ;SEND VABC
         MOVE     (A2)+,(A3)+ ;TO VABCOLD
         MOVE     (A2)+,(A3)+ ;
*
         MOVE.L   #VABCNEW,A2
         MOVE.L   #VABC,A4
         MOVE     (A2)+,(A4)+ ;SEND VABCNEW
         MOVE     (A2)+,(A4)+ ;TO VABC
         MOVE     (A2)+,(A4)+ ;
*
         BRA      IDLE
*
*        ****************************************
*        ****************************************
*                      SUBROUTINE MEM
*
MEM      EQU      *
         MOVE     D3,D5    ;X IN D5
         MULS     D4,D3    ;A*X IN D3
         ASR.L    #8,D3
```

```
        ASR.L   #8,D3
        ADD     D3,D2     ;A*X+U IN D2
        MULS    D2,D4     ;DXK*A IN D4
        ASR.L   #8,D4     ;DXK=A*X+U
        ASL     #1,D2     ;2*DXK IN D2
        MULS    #$5B,D4   ;DXK*A*T IN D4
        ASR.L   #8,D4
        ASR.L   #8,D4
        ADD     D2,D4     ;2*DXK+DXK*A*T
*                         ;IN D4
        MULS    #$5B,D4   ;(D4) * T IN
        ASR.L   #8,D4     ; D4
        ASR     #1,D4
        ADD     D4,D5     ;XK IN D5
        RTS
*
*       ****************************************
*                   SUBROUTINE MEM1
*
MEM1    EQU     *
        MOVE    D3,D5     ;X IN D5
        MULS    D4,D3     ;A*X IN D3
        ASR.L   #8,D3
        ADD     D3,D2     ;A*X+U IN D2
        MULS    D2,D4     ;DXK*A IN D4
        ASR.L   #8,D4
        ASL     #1,D2     ;2*DXK IN D2
        MULS    #$5B,D4   ;DXK*A*T IN D4
        ASR.L   #8,D4
        ASR.L   #8,D4
        ADD     D2,D4     ;2*DXK+DXK*A*T
*                         ;IN D4
        MULS    #$5B,D4   ;(D4)*T IN D4
        ASR.L   #8,D4
        ASR.L   #1,D4
        ASR.L   #8,D4
        ADD     D4,D5     ;XK IN D5
        RTS
*
*       ****************************************
*                   SUBROUTINE MEM2
*
MEM2    EQU     *
        MOVE    D3,D5     ;X IN D5
        MULS    D4,D3     ;A*X IN D3
        ASR.L   #8,D3
        ASR.L   #7,D3
        ADD     D3,D2     ;DXK=A*X+U IN D2
        MULS    D2,D4     ;DXK*A IN D4
        ASR.L   #8,D4
        ASL     #1,D2     ;2*DXK IN D2
        MULS    #$5B,D4   ;DXK*A*T IN D4
        ASR.L   #8,D4
        ASR.L   #8,D4
        ADD     D2,D4     ;2*DXK+DXK*A*T
```

```
*                              ;IN D4
        MULS    #$5B,D4
        ASR.L   #8,D4
        ASR     #2,D4     ;T/2*(D4)
        ADD     D4,D5     ;XK IN D5
        RTS
*
*       ****************************************
*                   SUBROUTINE MEM3
*
MEM3    EQU     *
        MULS    #$FFA5,D3 ;-X111/2.8
        ASL.L   #1,D3
        ADD.L   D3,D2     ;DXK=A*X+U
        MOVE.L  D2,D5     ; IN D2,D5
        ASR.L   #8,D2
        ASR.L   #4,D2
*
        MOVE    #$FFDF,D4 ;.001389/2.8
        MULS    D2,D4     ;DXK*A*T IN D4
        ASR.L   #4,D4
        ASL.L   #1,D5     ;2*DXK+
        ADD.L   D5,D4     ;DXK*A*T IN D4
        ASR.L   #8,D4
        ASR.L   #2,D4
        MULS    #$5B,D4   ;T*(D4)
        ASR.L   #8,D4
        ASR.L   #8,D4     ;T/2.*(D4)
        RTS
*
*       ****************************************
*                   SUBROUTINE MULM
*
MULM    EQU     *
        MOVE    D0,D4     ;XK1 IN D4
        MOVE    D1,D6     ;XK2 IN D6
        MULS    D2,D4     ;A21*XK1 IN D4
        MULS    D3,D6     ;A22*XK2 IN D6
        ADD.L   D4,D6
        ASR.L   #8,D6
        ASR.L   #4,D6
        SUB     D6,D5     ;DXK2 IN D5
        NEG     D5
        MOVE    D5,D7     ;DXK2 IN D7
        MOVE    D1,D6     ;XK2=DXK1 IN D6
        ASL     #1,D6     ;2*DXK1 IN D6
        MULS    #$5B,D7   ;T*DXK2 IN D7
        ASR.L   #8,D7
        ASR.L   #4,D7
        ADD     D7,D6     ;2*DXK1+T*DXK2
        MULS    #$5B,D6   ; IN D6
        ASR.L   #1,D6     ;(D6)*T/2 IN D6
        ASR.L   #8,D6
        ASR.L   #8,D6
        ADD     D6,D0     ;XK11 IN D0
```

```
        MULS    D7,D3    ;T*DXK2*A22 IN
        ASR.L   #4,D3    ; D3
        ASR.L   #8,D3
        ASL     #1,D5    ;2*DXK2 IN D5
        ADD     D5,D3
        MULS    D1,D2
        ASR.L   #8,D2
        MULS    #$5B,D2
        ASR.L   #8,D2    ;T*A21*DXK1 IN
        ASR.L   #8,D2    ;  D2
        ASR     #4,D2
        ADD     D2,D3    ;T*DXK2*A22+2*
        MULS    #$5B,D3  ;DXK2+T*A21*DXK1
        ASR.L   #8,D3    ;T/2*(...) IN D3
        ASR.L   #4,D3
        ASR     #1,D3
        ADD     D3,D1    ;XK12 IN D1
        RTS
*
*       ****************************************
*                   SUBROUTINE MULM1
*
MULM1   EQU     *
        MOVE    D0,D4    ;XK1 IN D4
        MOVE    D1,D6    ;XK2 IN D6
        MULS    D2,D4    ;A21*XK1 IN D4
        MULS    D3,D6    ;A22*XK2 IN D6
        ADD.L   D4,D6
        ASR.L   #8,D6
        ASR.L   #4,D6
        ASL     #2,D5
        SUB     D6,D5    ;DXK2 IN D5
        NEG     D5
        MOVE    D5,D7    ;DXK2 IN D7
        MOVE    D1,D6    ;XK2=DXK1 IN D6
        ASL     #1,D6    ;2*DXK1 IN D6
        MULS    #$5B,D7  ;T*DXK2 IN D7
        ASR.L   #8,D7
        ASR.L   #4,D7
        ADD     D7,D6    ;2*DXK1+T*DXK2
        MULS    #$5B,D6  ; IN D6
        ASR.L   #1,D6    ;(D6)*T/2 IN D6
        ASR.L   #8,D6
        ASR.L   #8,D6
        ADD     D6,D0    ;XK11 IN D0
        MULS    D7,D3    ;T*DXK2*A22 IN
        ASR.L   #4,D3    ; D3
        ASR.L   #8,D3
        ASL     #1,D5    ;2*DXK2 IN D5
        ADD     D5,D3
        MULS    D1,D2
        ASR.L   #8,D2
        MULS    #$5B,D2
        ASR.L   #8,D2    ;T*A21*DXK1 IN
        ASR.L   #8,D2    ;  D2
```

```
        ASR     #4,D2
        ADD     D2,D3    ;T*DXK2*A22+2*
        MULS    #$5B,D3  ;DXK2+T*A21*DXK1
        ASR.L   #8,D3    ;T/2*(...) IN D3
        ASR.L   #4,D3
        ASR     #1,D3
        ADD     D3,D1    ;XK12 IN D1
        RTS
*
*       ****************************************
*                  SUBROUTINE SSIN
*
SSIN    EQU     *
        MULS    #2000,D3
        ASR.L   #8,D3
        ASR.L   #4,D3      ;D3 IS INTEGER
        AND.L   #$FFFE,D3  ;D3 IS EVEN
        ADD     #$C500,D3
        MOVEA.L D3,A0
        MOVE    (A0),D3 ;SIN(DEL) IN D3
        RTS
*
*       ****************************************
*                   SUBROUTINE SSIN1
*
SSIN1   EQU     *
        MULS    #2000,D1
        ASR.L   #8,D1
        ASR.L   #4,D1     ;D1 IS INTEGER
        AND.L   #$FFFE,D1;D1 IS EVEN
        ADD     #$C500,D1
        MOVEA.L D1,A0
        MOVE    (A0),D3  ;SIN(FI) IN D3
        ADDA    #$1000,A0
        MOVE    (A0),D5  ;COS(FI) IN D5
        RTS
*
*       ****************************************
*                  SUBROUTINE PARK
*
PARK    EQU     *
        MOVE.L  DDEL1,D0
        ASR.L   #8,D0
        ASR.L   #1,D0
        MULS    #$5B,D0
        ASR.L   #7,D0
        ADD.L   #$860A91,D0;(W0+DDEL1)*T
        ADD.L   SITA,D0     ;PI/2+WT*I IN
        MOVE.L  D0,SITA    ;D0, SAVE SITA
        ASR.L   #8,D0
        ASR.L   #4,D0
*
        CMPI    #0,D0
        BGE     SGE         ;IF SITA<0
        ADDI    #$6488,D0  ;SITA+6.28
```

```
        BRA     SGE1
*
SGE     CMPI    #$6488,D0
        BLE     SGE1        ;IF SITA>6.28
        SUBI    #$6488,D0   ;SITA-6.28
        SUB.L   #$6487DE5,SITA
*                           ;SAVE SITA
SGE1    CMPI    #$1922,D0 ;IF SITA>1.57
        BGT     SGE2        ;GOTO SGE2
        MULS    #2000,D0
        ASR.L   #8,D0
        ASR.L   #4,D0
        AND.L   #$FFFE,D0 ;(D0) IS EVEN
        ADD     #$C500,D0
        MOVEA.L D0,A0
        MOVE    (A0),D3     ;SIN(SITA)
        ADDA    #$1000,A0   ;IN D3
        MOVE    (A0),D4     ;COS(SITA)
        BRA     PARKT       ;IN D4
SGE2    CMPI    #$3244,D0 ;IF SITA>3.14
        BGT     SGE3        ;GOTO SGE3
        MOVE    #$3244,D7
        SUB     D0,D7     ;PI-SITA IN D7
        MULS    #2000,D7
        ASR.L   #8,D7
        ASR.L   #4,D7
        AND.L   #$FFFE,D7 ;(D7) IS EVEN
        ADD     #$C500,D7
        MOVEA.L D7,A0
        MOVE    (A0),D3     ;SIN(SITA)
        ADDA    #$1000,A0   ;IN D3
        MOVE    (A0),D4
        NEG     D4        ;COS(SITA) IN D4
        BRA     PARKT
*
SGE3    CMPI    #$4B66,D0
        BGT     SGE4
        SUBI    #$3244,D0 ;SITA-PI IN D5
        MULS    #2000,D0
        ASR.L   #8,D0
        ASR.L   #4,D0
        AND.L   #$FFFE,D0 ;(D0) IS EVEN
        ADD     #$C500,D0
        MOVEA.L D0,A0
        MOVE    (A0),D3
        NEG     D3          ;SIN(SITA)
        ADDA    #$1000,A0   ;IN D3
        MOVE    (A0),D4
        NEG     D4          ;COS(SITA)
        BRA     PARKT       ; IN D4
*
SGE4    MOVE    #$6488,D7
        SUB     D0,D7       ;2*PI-SITA
        MULS    #2000,D7    ;IN D7
```

```
        ASR.L   #8,D7
        ASR.L   #4,D7
        AND.L   #$FFFE,D7   ;(D7) IS EVEN
        ADD     #$C500,D7
        MOVEA.L D7,A0
        MOVE    (A0),D3
        NEG     D3              ;SIN(SITA)
        ADDA    #$1000,A0       ;IN D3
        MOVE    (A0),D4         ;COS(SITA)
*                               ;IN D4
PARKT   MOVE    D3,D6           ;SIN(SITA) IN
*                               ; D3,D6
        MOVE    D4,D7           ;COS(SITA) IN
        MOVE    D7,D0           ; D7,D4,D0
        ASR     #1,D3           ;(D3)/2
        ASR     #1,D4           ;(D4)/2
        MOVE.L  #VABC1,A1
        LEA     SIGNALDATA,A0
        MOVE.L  (A0)+,(A1)+     ;READ INPUT SAMPLES
        MOVE.W  (A0)+,(A1)+     ;VA,VB,VC (DIDN'T USE)
        MOVE.L  (A0)+,IABCNEW   ;READ INPUT SAMPLES
        MOVE.W  (A0)+,IABCNEW+4;IA,IB,IC(PHASE CURRENT)
*
        JSR     EXTRAINPUT
*
        MOVE    IA,D5
        MULS    D5,D0     ;COS(SITA)
        ASR.L   #8,D0     ;*IA IN D0
        MULS    D6,D5     ;SIN(SITA)
        ASR.L   #8,D5     ;*IA IN D5
*
        MULS    #$DDB,D6 ;SIN(SITA)
        ASR.L   #8,D6     ;*.866 IN D6
        ASR.L   #4,D6
        MULS    #$DDB,D7 ;COS(SITA)
        ASR.L   #8,D7     ;*.866 IN D7
        ASR.L   #4,D7
*
        MOVE    D4,D2
        ADD     D6,D4     ;COS(SITA)/2+
*                        ;.866*SIN(SITA)
        MOVE    IC,D1
        MULS    D1,D4    ;IC*(D4) IN D4
        ASR.L   #8,D4
*
        SUB     D2,D6    ;.866*S-.5*C
        MOVE    IB,D2
        MULS    D2,D6    ;IB*(D6) IN D6
        ASR.L   #8,D6
*
        SUB     D4,D6
        ADD     D6,D0
        MULS    #$AAB,D0  ;ID IN D0
        ASR.L   #8,D0
        ASR.L   #8,D0
```

```
*
        MOVE    D3,D6       ;.5*s in d6
        ADD     D7,D3       ;(S/2+.866*C)
        MULS    D2,D3       ;*IB IN D3
        ASR.L   #8,D3
        SUB     D6,D7       ;(.866*C-S/2)
        MULS    D1,D7       ;*IC IN D7
        ASR.L   #8,D7
*
        SUB     D3,D7
        ADD     D7,D5
        MULS    #$AAB,D5    ;IQ IN D5
        ASR.L   #8,D5
        ASR.L   #8,D5
*
        MOVE    D0,AID1     ;SAVE ID
        MOVE    D5,AIQ1     ;SAVE IQ
        RTS
*
*       ****************************************
*               SUBROUTINE EXTRAINPUT
*
EXTRAINPUT:  MOVEM.L D0-D7/A0-A6,SAVEII
*
        MOVE.L  #IABCNEW,A0
        MOVE.L  #IABCOLD,A1
*
        FMOVE.S #-0.3472963,FP0
        FMOVE.S #1.2855752,FP1
        MOVE.L  #IA,A2
*
        MOVE.W  #2,D0
CAL:    FMOVE.W (A0)+,FP2
        FMOVE.W (A1)+,FP3
        FMUL.X  FP0,FP3
        FMUL.X  FP1,FP2
        FADD.X  FP2,FP3
        FMOVE.W FP3,(A2)+
        DBRA    D0,CAL
*
        MOVE.W  IB,D0
        MOVE.W  IC,IB
        MOVE.W  D0,IC
        MOVE.L  #IABCNEW,A0
        MOVE.L  #IABCOLD,A1
        MOVE.L  (A0)+,(A1)+
        MOVE.W  (A0)+,(A1)+
        MOVEM.L SAVEII,D0-D7/A0-A6
*
        RTS
*
*       ***********************************************
*       ***********************************************
DEL     DC.L    $71CA47
TIME    DC.L    $5B
```

```
SITA     DC.L    $1921FB5
FI       DC.L    $1921FB5
*
FIRST    DS.L    1
EFD      DS.W    1
VT       DS.W    1
AID1     DS.W    1
AID      DS.W    1
AIQ1     DS.W    1
AIQ      DS.W    1
IA       DS.W    1
IB       DS.W    1
IC       DS.W    1
PSID1    DS.W    1
PSID     DS.W    1
PSIQ1    DS.W    1
PSIQ     DS.W    1
PSI      DS.W    1
PSIL     DS.W    1
TE1      DS.L    1
TM1      DS.L    1
DDEL     DS.L    1
DDEL1    DS.L    1
DDEL3    DS.W    1
*
X1       DS.L    1
X2       DS.W    1
X3       DS.W    1
X4       DS.W    1
X61      DS.W    1
X62      DS.W    1
X71      DS.W    1
X72      DS.W    1
X81      DS.L    1
X82      DS.L    1
X83      DS.W    1
X9       DS.W    1
X101     DS.W    1
X102     DS.W    1
X111     DS.W    1
X112     DS.W    1
X113     DS.L    1
X12      DS.W    1
*
VABCNEW  DS.W    3  ;VA,VB,VC STORAGE SPACE
VABC     DS.W    3  ;
VABCOLD  DS.W    3  ;
VEXTRA   DS.W    9  ;STORAGE FOR Vextra
VABC1    DS.W    3
IABCOLD  DS.W    3;OLD IA,IB,IC STORAGE SPACE
IABCNEW  DS.W    3;REAL IA,IB,IC STORAGE SPACE
SAVEII   DS.L    20
LAST     DS.L    1
*
         END
```

```
*************************************************************
*                                                           *
*  THIS SUBROUTINE IS UESD TO CALCULATE THE SYNCHRONOUS     *
*  MACHINE'S SATURATION FUNCTION.                           *
*                                                           *
*************************************************************
*
        XREF    PSID1,PSIQ1,PSI,PSIL
*
        ORG     $9000
        MOVE    PSID1,D1
        MULS    PSID1,D1        ;PSID*PSID
        MULS    PSIQ1,D2        ;PSIQ*PSIQ
        ADD.L   D2,D1           ;PSIL*PSIL
        ASR.L   #8,D1           ;IN D1
*
        CMP     #$1000,D1
        BLE     LIMI
        MOVE    #$1000,D1
*
LIMI    MULS    #500,D1
        ASR.L   #8,D1           ;(D1)=INTEGER
        AND.L   #$FFFE,D1       ;(D1)=EVEN NO
        ADD     #$A500,D1
        MOVEA.L D1,A2
        MOVE    (A2),D1         ;PSIL IN D1
        MOVE    D1,D5           ;PSIL IN D5
        MOVE    D1,PSIL
*
        CMPI    #$3B,D1         ;IF PSIL>=.7
        BGT     SATU            ;GOTO SATU
*
        MOVE    D1,PSI
        MOVE    PSID1,D1
        MOVE    PSIQ1,D2
        BRA     SATU1
*
SATU    MOVE    PSI,D3          ;(D3)=PSI
        MOVE    D3,D4           ;(D4)=PSI
        SUB     #$3B,D3         ;(D3)=PSI-.7
        MULS    #$7FF,D3        ;(D3)=(D3)*BG
        ASR.L   #8,D3           ;(BG=7.995)
        MULS    #1000,D3
        ASR.L   #8,D3           ;(D3)=INTEGER
        AND.L   #$FFFE,D3       ;(D3)=EVEN NO
        ADD     #$9100,D3
        MOVEA.L D3,A2
        MOVE    (A2),D3     ;EXP(BG*(PSI-.7))
        MULS    #$9,D3      ;SG=AG*EXP(..) IN D3
        ASR.L   #4,D3       ;(AG=0.035)
        MULS    PSI,D3      ;PSI*SG IN D3
        ASR.L   #8,D3
        ASR.L   #4,D3
        SUB     D3,D1       ;PSIL-PSI*SG IN D1
```

```
*
        CMP     D4,D1           ;PSI<PSI1
        BLT     SATU2
*
        BRA     SATU3
*
SATU2   MOVE    D4,D1
*
SATU3   MOVE    D1,PSI          ;SAVE PSI
*
        MOVE    D1,D2
        MULS    PSID1,D1        ;PSID1*PSI
        DIVS    D5,D1           ;PSID1*PSI/PSIL
*                               ;=PSID1 IN D1
        MULS    PSIQ1,D2        ;PSIQ1*PSI
        DIVS    D5,D2           ;PSIQ1*PSI/PSIL
*                               ;=PSIQ1 IN D2
        MOVE    D1,PSID1        ;SAVE PSID1
        MOVE    D2,PSIQ1        ;SAVE PSIQ1
*
SATU1   RTS
```

```
*************************************************************
*                                                           *
*              USER INTERFACE INITIALIZE SYSTEM             *
*            TIMER HANDLING AND INTERRUP ROUTINE            *
*         A/C CONVERSION AND COLLECTION OF RAW DATA         *
*                                                           *
*************************************************************
*
            XDEF    VREAL,VIMAG,PHASORFLG,sdata1
            XDEF    SIGNALDATA,EXTRAPOLATE
            XREF    IDLE,VEXTRA,VABCNEW,VABC,VABCOLD
            XREF    FIRST,LAST


*       TTL     'ERROR MODEL GENERATOR'
*
*   ADDRESS SPACE OF MULTIFUNCTION PERIPHERAL 68901
*
GPIP        EQU     $FFF80000
AER         EQU     $FFF80002
DDR         EQU     $FFF80004
IERA        EQU     $FFF80006
IERB        EQU     $FFF80008
IPRA        EQU     $FFF8000A
IPRB        EQU     $FFF8000C
ISRA        EQU     $FFF8000E
ISRB        EQU     $FFF80010
IMRA        EQU     $FFF80012
IMRB        EQU     $FFF80014
VR          EQU     $FFF80016
TACR        EQU     $FFF80018
TBCR        EQU     $FFF8001A
TCDCR       EQU     $FFF8001C
TADR        EQU     $FFF8001E
TBDR        EQU     $FFF80020
TCDR        EQU     $FFF80022
TDDR        EQU     $FFF80024
SCR         EQU     $FFF80026
UCR         EQU     $FFF80028
RSR         EQU     $FFF8002A
TSR         EQU     $FFF8002C
UDR         EQU     $FFF8002E
*
*   ADDRESS SPACE OF A/D BOARD DT 1401 BOARD 1
*
BASE        EQU     $FFFFFF00
CSR         EQU     BASE
ADCHAN      EQU     BASE+2
PACLK       EQU     BASE+4
ADDATA      EQU     BASE+6
DAC0        EQU     BASE+8
DAC1        EQU     BASE+10
DIOREG      EQU     BASE+12
*   ADDRESS SPACE OF A/D BOARD DT 1401 BOARD 2
*
```

```
BASE2       EQU     $FFFFFE00
CSR2        EQU     BASE2
ADCHAN2     EQU     BASE2+2
PACLK2      EQU     BASE2+4
ADDATA2     EQU     BASE2+6
DAC02       EQU     BASE2+8
DAC12       EQU     BASE2+10
DIOREG2     EQU     BASE2+12
*
CORLEN      EQU     8
*
*************************************************************
*                                                           *
*        * * *   MAIN PROGRAM    * * *                      *
*                                                           *
*************************************************************
*
            ORG     $5000
            MOVE.L  #$5000,A7
*
            MOVE.L  #TIMER,($03F4)      ;Define address of the
*                                       ;interrupt routine
            MOVE.L  #DUMMY1,A0
            MOVE.L  #DUMMY2,A1
INI:        MOVE.L  #0,(A0)+
            CMPA.L  A0,A1
            BGE     INI
*
            MOVE.L  #FIRST,A0
            MOVE.L  #LAST,A1
INI1:       MOVE.L  #0,(A0)+
            CMPA.L  A0,A1
            BGE     INI1
*
            FMOVE.S #-0.2610523,FP6     ;Initialization for
            FMOVE.S #1.2175229,FP7      ;EXTRAPOLATE
*
            MOVE.L  #0,OFFSET_FLG
            MOVE.W  #0,LEDCOUNT
            MOVE.L  #MASTAB,A3    ;MOVE VALUE OF TABLE BEGINNING
            MOVE.L  A3,SAVADR     ;SAVE FOR NEXT STEP
*
            MOVE.W  #$1000,CSR    ;Initialising the AD board
            MOVE.W  #$0300,CSR    ;Set the DIOREG as 16 bit
*                                 ;output bort
            ANDI    #$00CF,DDR
            ORI     #$00B0,AER
            ANDI    #$00CF,GPIP
            ORI     #$0030,DDR
            MOVE.W  #$2400,SR
*
            MOVE.W  #$00F0,VR  ;Set the vector intterupt address
            MOVE.W  #$0010,TACR ;STARTING TIMER #A
            MOVE.W  #$0069,TADR
            MOVE.W  #$0001,TACR
```

```
            MOVE.B  #$20,IERA+1
            MOVE.B  #$20,IMRA+1
*
            BRA     IDLE
*
*************************************************************
*                                                           *
*     A/D  CONVERSION  AND  COLLECTION  OF RAW  DATA        *
*                                                           *
*************************************************************
*
AD:         MOVEM.L D0-D7/A0-A6,SAVE ;SAVING THE REGISTERS
            MOVE.L  #0,D1
            MOVE.L  #SDATA1,A1      ;SET POINTERS TO TEMPORARY
            MOVE.B  #0,ADCHAN+1
            MOVE.B  #6,ADCHAN
            MOVE.W  #6,D4
*
            MOVE.W  #$1000,CSR    ;Initialize AD board
            MOVE.W  #$0300,CSR
*
START:      CMP.W   #0,D4
            BEQ     DONAD
            MOVE.W  #$0700,CSR
*
CK_TO:      MOVE.W  CSR,D0
            ANDI.W  #$0002,D0
            BNE     CK_TO
*
            MOVE.W  #$0305,CSR
*
CK_DN:      MOVE.W  CSR,D0
            ANDI.W  #$0080,D0
            BEQ     CK_DN
*
*           MOVE.B  ADCHAN+1,D0
*           MOVE.B  ADCHAN,D1
*           CMP.B   D0,D1
*           BNE     READ1
*           MOVE.W  CSR,D0
*           ANDI.W  #$0020,D0
*           BEQ     READ2
*
*           JMP     DONAD
*
READ1:      MOVE.W  ADDATA,(A1)+ ;Moving data in the
            SUB.W   #1,D4        ;temperory buffer
            JMP     START        ;before it can be processed
*READ2:      MOVE.W  ADDATA,(A1) ;Last channel read
*
DONAD:  MOVE.L  #SDATA1,A1          ;SET POINTERS TO TEMPORARY &
        MOVE.L  #SDATA,A2                 ;FINAL DESTINATION ARRAYS
        move.l  #signaldata,a3
*
*************************************************************
```

```
*
        MOVE.W  #5,D4    ; COUNTER FOR 6 PHASES
XY:     MOVE.W  (A1)+,(A2)    ;SCALING LOOP WHICH MOVES DATA
        MOVE.W  (A2)+,D0        ;FROM ARRAY SDATA1 INTO SDATA
        MOVE.W  D0,D1           ;& CALCULATES FACTORS
        MOVE.W  D1,D2           ;1/2*CONV(i) & SQRT(3)/2*CONV(i)
        move.w  d2,d3
        asl.w   #2,d3
        asr.w   #1,d1
        sub.w   d1,d3
        asr.w   #2,d1
        SUB.W   D1,D0
        asr.w   #1,d1
        add.w   d1,d3
        asr.w   #1,d1
        sub.w   d1,d3
        asr.w   #2,d1
        SUB.W   D1,D0
        asr.w   #1,d1
        add.w   d1,d3
        asr.w   #2,d1
        SUB.W   D1,D0
        ASR.W   #1,D2
        MOVE.W  D0,(A2)+
        MOVE.W  D2,(A2)+
        move.w  d3,(a3)+
        DBRA    D4,XY             ;END OF SCALING LOOP
        MOVEM.L SAVE,D0-D7/A0-A6 ;RESTORING THE REGISTERS
        RTS
*
**********************************************************
*
*          PHASOR CALCULATION ROUTINE
*
PHASOR: MOVEM.L  D0-D7/A0-A6,SAVE    ;SAVING REGISTERS
        MOVE.W   #1,VIFLG            ;INITIALIZE COUNT
        MOVEA.L  SAVADR,A3           ;RETRIEVE MASTER ADDRESS
        ADD      #6,A3               ;UPDATE THE MASTER ADDRESS
        CMPA     #MASTEN,A3          ;RESET, IF NECCESSARY
        BLE      SKIP                ;
        MOVE.L   #MASTAB,A3         ;MOVE VALUE OF TABLE BEGINNING
SKIP:   MOVE.L   A3,SAVADR           ;SAVE FOR NEXT STEP
        MOVEM    (A3)+,A1            ;GET SUBROUTINE  ADDRESS
        MOVE.L   (A3),A2             ;GET CORRECTION  ADDRESS
        MOVEA.L  #SDATA,A3           ;GET INPUT DATA  ADDRESS
        MOVEM    (A3)+,D0-D2         ;MOVE THE DATA FROM THE
        MOVE     (A3)+,D6            ;INPUT  BUFFER INTO THE
        MOVE     (A3)+,A0            ;REGISTERS FOR USE HERE
        MOVE     (A3)+,D7            ;
        MOVEM    (A3),D3-D5          ;
        MOVEA.L  #PHTABL,A4          ;GET OLD PHASOR ADDRESS
        JMP      (A1)                ;JUMP TO SUBROUTINE
STARTI: ADD.W    #1,VIFLG            ;CHECK VOLTAGE/CURRENT FLAG
        MOVEM    (A3)+,D0-D2         ;MOVE THE DATA FROM THE
        MOVE     (A3)+,D6            ;INPUT  BUFFER INTO THE
```

```
        MOVE    (A3)+,A0            ;REGISTERS FOR USE HERE
        MOVE    (A3)+,D7            ;
        MOVEM   (A3)+,D3-D5         ;
        JMP     (A1)                ;JUMP TO SUBROUTINE
*
*   12 SUBROUTINES TO HANDLE EACH VALUE OF K START HERE
*
;                           DATA ORGANIZATION
;
;                            D0 -----   Va
;                            D1 -----   Va'
;                            D2 -----   Va"
;                            D3 -----   Vc
;                            D4 -----   Vc'
;                            D5 -----   Vc"
;                            D6 -----   Vb
;                            A0 -----   Vb'
;                            D7 -----   Vb"
;                            Va' = Va * 0.866
;                            Va" = Va * 0.500
;
K0:     SUB     A0,D1           ;SUBROUTINE FOR K = 0
        MOVE    D1,D0           ;
        SUB     (A2),D1         ;
        ADD     D1,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;  SAVE REAL VALUE CORRECTION
        ADD     D7,D2           ;
        SUB     D2,D3               ;
        MOVE    D3,D0               ;
        SUB     (A2),D3             ;
        SUB     D3,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K1:     ADD     D7,D5           ;SUBROUTINE FOR K = 1
        SUB     D5,D0           ;
        MOVE    D0,D1           ;
        SUB     (A2),D0         ;
        ADD     D0,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D1,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     A0,D4           ;
        MOVE    D4,D1           ;
        SUB     (A2),D4         ;
        SUB     D4,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D1,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K2:     SUB     D4,D1           ;SUBROUTINE FOR K = 2
        MOVE    D1,D0           ;
        SUB     (A2),D1         ;
        ADD     D1,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        ADD     D2,D5           ;
        SUB     D6,D5           ;
        MOVE    D5,D0           ;
        SUB     (A2),D5         ;
        SUB     D5,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
```

```
K3:     ADD     D7,D2           ;SUBROUTINE FOR K = 3
        SUB     D3,D2           ;
        MOVE    D2,D0           ;
        SUB     (A2),D2         ;
        ADD     D2,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     A0,D1           ;
        MOVE    D1,D0           ;
        SUB     (A2),D1         ;
        SUB     D1,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K4:     SUB     D4,A0           ;SUBROUTINE FOR K = 4
        MOVE    A0,D3           ;
        SUB     (A2),A0         ;
        MOVE    A0,D1           ;
        ADD     D1,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D3,(A2)+        ;SAVE REAL VALUE CORRECTION
        ADD     D7,D5           ;
        SUB     D5,D0           ;
        MOVE    D0,D3           ;
        SUB     (A2),D0         ;
        SUB     D0,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D3,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K5:     ADD     D5,D2           ;SUBROUTINE FOR K = 5
        SUB     D2,D6           ;
        MOVE    D6,D0           ;
        SUB     (A2),D6         ;
        ADD     D6,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     D4,D1           ;
        MOVE    D1,D0           ;
        SUB     (A2),D1       ;
        SUB     D1,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K6:     SUB     D1,A0           ;SUBROUTINE FOR K = 6
        MOVE    A0,D0           ;
        SUB     (A2),A0         ;
        MOVE    A0,D4           ;
        ADD     D4,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        ADD     D7,D2           ;
        SUB     D3,D2           ;
        MOVE    D2,D0           ;
        SUB     (A2),D2         ;
        SUB     D2,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K7:     ADD     D7,D5           ;SUBROUTINE FOR K = 7
        SUB     D0,D5           ;
        MOVE    D5,D3           ;
        SUB     (A2),D5         ;
        ADD     D5,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
```

```
        MOVE    D3,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     D4,A0           ;
        MOVE    A0,D3           ;
        SUB     (A2),A0         ;
        MOVE    A0,D6           ;
        SUB     D6,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D3,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K8:     SUB     D1,D4           ;SUBROUTINE FOR K = 8
        MOVE    D4,D0           ;
        SUB     (A2),D4         ;
        ADD     D4,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        ADD     D5,D2           ;
        SUB     D2,D6           ;
        MOVE    D6,D0           ;
        SUB     (A2),D6         ;
        SUB     D6,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K9:     ADD     D7,D2           ;SUBROUTINE FOR K = 9
        SUB     D2,D3           ;
        MOVE    D3,D0           ;
        SUB     (A2),D3
        ADD     D3,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     D1,A0           ;
        MOVE    A0,D0           ;
        SUB     (A2),A0         ;
        MOVE    A0,D6           ;
        SUB     D6,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K10:    SUB     A0,D4           ;SUBROUTINE FOR K = 10
        MOVE    D4,D3           ;
        SUB     (A2),D4         ;
        ADD     D4,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D3,(A2)+        ;SAVE REAL VALUE CORRECTION
        ADD     D7,D5           ;
        SUB     D0,D5           ;
        MOVE    D5,D3           ;
        SUB     (A2),D5         ;
        SUB     D5,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
        MOVE    D3,(A2)+        ;SAVE IMAG VALUE CORRECTION
        BRA     CHECKVI         ;
K11:    ADD     D5,D2           ;SUBROUTINE FOR K = 11
        SUB     D6,D2           ;
        MOVE    D2,D0           ;
        SUB     (A2),D2         ;
        ADD     D2,(A4)+        ;MOVE PHASOR-REAL TO OUTPUT
        MOVE    D0,(A2)+        ;SAVE REAL VALUE CORRECTION
        SUB     D1,D4           ;
        MOVE    D4,D0           ;
        SUB     (A2),D4         ;
        SUB     D4,(A4)+        ;MOVE PHASOR-IMAG TO OUTPUT
```

```
        MOVE      D0,(A2)+          ;SAVE IMAG VALUE CORRECTION
CHECKVI CMPI.W    #1,VIFLG          ;CHECK FOR END
        BNE       STARTI            ;THE RESULTS PROCESSING
        MOVE.L    #PHTABL,A4
        FMOVE.S   #81.458701,FP1
        FMOVE.W   (A4)+,FP0
        FDIV      FP1,FP0
        FMOVE.W   FP0,VREAL
        FMOVE.W   (A4)+,FP0
        FDIV      FP1,FP0
        FMOVE.W   FP0,VIMAG
        MOVEM.L   SAVE,D0-D7/A0-A6 ;RESTORE THE REGISTERS FOR
        RTS                        ;RETURN TO CALLING PROGRAM
*
**********************************************************
*                                                        *
*       * * *   TIMER HANDLING ROUTINE   * * *           *
*                  GENERATS 2880 HZ,                     *
*                  CALLS 'AD' & 'PHASOR' AT 720 HZ       *
*                                                        *
**********************************************************
*                                                        *
TIMER:      MOVEM.L SAVE2,D0/A0  ;SAVE THE REGISTERS
            MOVE.W  #$0010,TACR
            MOVE.W  #0,D0
DELAY:      DBRA    D0,DELAY
            NOP
            NOP
            MOVE.W  #$0001,TACR   ;The three intermidiate values
            CMP.L   #18,OFFSET_FLG;for voltages are extrapolated
            BNE     SKIP_P        ;and outputed at 2880 Hz
            AND.W   #$FFF0,DIOREG ;Set hold mode
            MOVE.L  #VABCNEW,A0
            MOVE.W  (A0)+,DAC02   ;Output VA,VB,VC
            MOVE.W  (A0)+,DAC0
            MOVE.W  (A0)+,DAC1
            BSR     AD            ;Perform A/D conversion
            OR.W    #$000F,DIOREG ;Set sample mode
            BSR     PHASOR        ;Calculate voltage phasor
            MOVE.W  #1,PHASORFLG
            MOVE.L  #0,OFFSET_FLG ;OFFSET_FLG vary from 0 to 18
            ADD.W   #1,LEDCOUNT   ;in steps of 6
            CMP.W   #720,LEDCOUNT
            BGE     RUNLIGHT
            MOVEM.L SAVE2,D0/A0   ;RESTORE THE REGISTERS
            RTE
SKIP_P:     MOVE.L  #VEXTRA,A0    ;Calculate the address for
            MOVE.L  A0,D0         ;the extrapolated
            ADD.L   OFFSET_FLG,D0 ;values of the voltages
            MOVE.L  D0,A0
            MOVE.W  (A0)+,DAC02   ;Output VA,VB,VC
            MOVE.W  (A0)+,DAC0
            MOVE.W  (A0)+,DAC1
            ADD.L   #6,OFFSET_FLG ;Update the offset
            MOVEM.L SAVE2,D0/A0   ;RESTORE THE REGISTERS
```

```
            RTE
RUNLIGHT:   MOVE.W  DIOREG,D0
            BCHG    #4,D0
            MOVE.W  D0,DIOREG
            MOVE.W  #0,LEDCOUNT
            MOVEM.L SAVE2,D0/A0  ;RESTORE THE REGISTERS
            RTE
*
*********************************************************
*
*              MASTER TABLE STARTS HERE
*
MASTAB: DC      K0          ;     SUBROUTINE K0 ADDRESS
        DC.L  CORTAB+(0*CORLEN) ;CORRECTION TERMS ADDRESS
        DC      K1                ;
        DC.L    CORTAB+(1*CORLEN);
        DC      K2                ;
        DC.L    CORTAB+(2*CORLEN);
        DC      K3                ;
        DC.L    CORTAB+(3*CORLEN);
        DC      K4                ;
        DC.L    CORTAB+(4*CORLEN);
        DC      K5                ;
        DC.L    CORTAB+(5*CORLEN);
        DC      K6                ;
        DC.L    CORTAB+(6*CORLEN);
        DC      K7                ;
        DC.L    CORTAB+(7*CORLEN);
        DC      K8                ;
        DC.L    CORTAB+(8*CORLEN);
        DC      K9                ;
        DC.L    CORTAB+(9*CORLEN);
        DC      K10               ;
        DC.L    CORTAB+(10*CORLEN);
MASTEN: DC      K11               ;
        DC.L    CORTAB+(11*CORLEN);
*********************************************************
*                                                       *
* The subroutine extrapolates three points in a 60 hz   *
* sine wave.  The input is two points measured at rate  *
* of 720 Hz and output is three points calculated at    *
* 2880 Hz.                                              *
*********************************************************
EXTRAPOLATE MOVEM.L D0-D7/A0-A6,SAVEI
*                  ;SAVING THE REGISTERS
*
        MOVE.L  #VEXTRA,A4
        BSR     CAL
*
        FMOVE.S #-0.517638,FP6
        FMOVE.S #1.4142136,FP7
        BSR     CAL
*
        FMOVE.S #-0.7653668,FP6
        FMOVE.S #1.5867067,FP7
```

```
        BSR     CAL
*
        FMOVE.S #-0.2610523,FP6
        FMOVE.S #1.2175229,FP7
*
        MOVEM.L SAVEI,D0-D7/A0-A6
        RTS
*****************************************************
*
CAL:    MOVE.L  #VABCOLD,A3
        MOVE.L  #VABC,A2
*
        FMOVE.W  (A2)+,FP0
        FMOVE.W  (A2)+,FP1
        FMOVE.W  (A2)+,FP2
        FMOVE.W  (A3)+,FP3
        FMOVE.W  (A3)+,FP4
        FMOVE.W  (A3)+,FP5
*
        FMUL.X   FP6,FP3
        FMUL.X   FP6,FP4
        FMUL.X   FP6,FP5
        FMUL.X   FP7,FP0
        FMUL.X   FP7,FP1
        FMUL.X   FP7,FP2
        FADD.X   FP3,FP0
        FADD.X   FP4,FP1
        FADD.X   FP5,FP2
*
        FMOVE.W  FP0,(A4)+
        FMOVE.W  FP1,(A4)+
        FMOVE.W  FP2,(A4)+
*
        RTS
*************************************************************
DUMMY1      DS.L    1
OFFSET_FLG  DS.L    1
LEDCOUNT    DS.W    1
SDATA1      DS.W    6
SDATA       DS.W    18
SIGNALDATA  DS.W    6
SAVE        DS.L    20
SAVEI       DS.L    20
SAVE2       DS.L    20
SAVADR      DS.L    1
VIFLG       DS.W    1
VREAL       DS.W    1
VIMAG       DS.W    1
PHASORFLG   DS.W    1
PHTABL      DS.W    4
CORTAB      DS.L    24
DUMMY2      DS.L    1
TABEL       EQU     $F000
            END
```

```
          TTL     'EXPONENTIAL FUNCTION LOOK-UP TABLE'
*
          ORG     $9100
EXPTBL    DC.W    $100
          DC.W    $0101,$0101,$0102,$0102,$0103
          DC.W    $0103,$0104,$0104,$0105,$0105
          DC.W    $0106,$0106,$0107,$0107,$0108
          DC.W    $0108,$0109,$0109,$010A,$010A
          DC.W    $010B,$010C,$010C,$010D,$010D
          DC.W    $010E,$010E,$010F,$010F,$0110
          DC.W    $0110,$0111,$0111,$0112,$0113
          DC.W    $0113,$0114,$0114,$0115,$0115
          DC.W    $0116,$0116,$0117,$0118,$0118
          DC.W    $0119,$0119,$011A,$011A,$011B
          DC.W    $011B,$011C,$011D,$011D,$011E
          DC.W    $011E,$011F,$011F,$0120,$0121
          DC.W    $0121,$0122,$0122,$0123,$0124
          DC.W    $0124,$0125,$0125,$0126,$0126

                        .   .   .
                        .   .   .
                        .   .   .

          DC.W    $8148,$818B,$81CD,$8210,$8252
          DC.W    $8295,$82D8,$831B,$835E,$83A1
          DC.W    $83E5,$8429,$846C,$84B0,$84F4
          DC.W    $8538,$857D,$85C1,$8606,$864A
          DC.W    $868F,$86D4,$8719,$875E,$87A4
          DC.W    $87E9,$882F,$8875,$88BB,$8901
          DC.W    $8947,$898D,$89D4,$8A1A,$8A61
          DC.W    $8AA8,$8AEF,$8B36,$8B7E,$8BC5
          DC.W    $8C0D,$8C55,$8C9D,$8CE5,$8D2D
          DC.W    $8D75,$8DBE,$8E06,$8E4F,$8E98
          DC.W    $8EE1,$8F2A,$8F74,$8FBD,$9007
          DC.W    $9051,$909B,$90E5,$912F,$9179
          DC.W    $91C4,$920F,$925A,$92A5,$92F0
          DC.W    $933B,$9387,$93D2,$941E,$946A
          END
```

```
          TTL     'SQUARE ROOT FUNCTION LOOK-UP TABLE'
*
          ORG     $A500
SQRTBL    DC.W    $0
          DC.W    $10,$17,$1C,$20,$24,$28,$2B,$2E,$31,$33
          DC.W    $36,$38,$3A,$3D,$3F,$41,$43,$45,$47,$48
          DC.W    $4A,$4C,$4E,$4F,$51,$53,$54,$56,$57,$59
          DC.W    $5A,$5C,$5D,$5E,$60,$61,$62,$64,$65,$66
          DC.W    $68,$69,$6A,$6B,$6D,$6E,$6F,$70,$71,$72
          DC.W    $74,$75,$76,$77,$78,$79,$7A,$7B,$7C,$7D
          DC.W    $7E,$7F,$81,$82,$83,$84,$85,$86,$86,$87
          DC.W    $88,$89,$8A,$8B,$8C,$8D,$8E,$8F,$90,$91
          DC.W    $92,$93,$94,$94,$95,$96,$97,$98,$99,$9A
          DC.W    $9A,$9B,$9C,$9D,$9E,$9F,$9F,$A0,$A1,$A2
          DC.W    $A3,$A4,$A4,$A5,$A6,$A7,$A7,$A8,$A9,$AA
          DC.W    $AB,$AB,$AC,$AD,$AE,$AE,$AF,$B0,$B1,$B1
          DC.W    $B2,$B3,$B4,$B4,$B5,$B6,$B6,$B7,$B8,$B9
          DC.W    $B9,$BA,$BB,$BB,$BC,$BD,$BE,$BE,$BF,$C0

                                  .   .   .
                                  .   .   .
                                  .   .   .

          DC.W    $3EE,$3EE,$3EE,$3EE,$3EF,$3EF,$3EF,$3EF,$3EF,$3EF
          DC.W    $3EF,$3EF,$3F0,$3F0,$3F0,$3F0,$3F0,$3F0,$3F0,$3F1
          DC.W    $3F1,$3F1,$3F1,$3F1,$3F1,$3F1,$3F1,$3F2,$3F2,$3F2
          DC.W    $3F2,$3F2,$3F2,$3F2,$3F2,$3F3,$3F3,$3F3,$3F3,$3F3
          DC.W    $3F3,$3F3,$3F4,$3F4,$3F4,$3F4,$3F4,$3F4,$3F4,$3F4
          DC.W    $3F5,$3F5,$3F5,$3F5,$3F5,$3F5,$3F5,$3F5,$3F6,$3F6
          DC.W    $3F6,$3F6,$3F6,$3F6,$3F6,$3F6,$3F7,$3F7,$3F7,$3F7
          DC.W    $3F7,$3F7,$3F7,$3F8,$3F8,$3F8,$3F8,$3F8,$3F8,$3F8
          DC.W    $3F8,$3F9,$3F9,$3F9,$3F9,$3F9,$3F9,$3F9,$3F9,$3FA
          DC.W    $3FA,$3FA,$3FA,$3FA,$3FA,$3FA,$3FA,$3FB,$3FB,$3FB
          DC.W    $3FB,$3FB,$3FB,$3FB,$3FC,$3FC,$3FC,$3FC,$3FC,$3FC
          DC.W    $3FC,$3FC,$3FD,$3FD,$3FD,$3FD,$3FD,$3FD,$3FD,$3FD
          DC.W    $3FE,$3FE,$3FE,$3FE,$3FE,$3FE,$3FE,$3FE,$3FF,$3FF
          DC.W    $3FF,$3FF,$3FF,$3FF,$3FF,$3FF,$400,$400,$400,$400
          END
```

```
        TTL     'SINE FUNCTION LOOK-UP TABLE'
        ORG      $c500
SINTBL   DC.W     $0
        DC.W    $00,$01,$01,$01,$01,$02,$02,$02,$02,$03
        DC.W    $03,$03,$03,$04,$04,$04,$04,$05,$05,$05
        DC.W    $05,$06,$06,$06,$06,$07,$07,$07,$07,$08
        DC.W    $08,$08,$08,$09,$09,$09,$09,$0A,$0A,$0A
        DC.W    $0A,$0B,$0B,$0B,$0C,$0C,$0C,$0C,$0D,$0D
        DC.W    $0D,$0D,$0E,$0E,$0E,$0E,$0F,$0F,$0F,$0F
        DC.W    $10,$10,$10,$10,$11,$11,$11,$11,$12,$12
        DC.W    $12,$12,$13,$13,$13,$13,$14,$14,$14,$14
        DC.W    $15,$15,$15,$15,$16,$16,$16,$16,$17,$17
        DC.W    $17,$18,$18,$18,$18,$19,$19,$19,$19,$1A
        DC.W    $1A,$1A,$1A,$1B,$1B,$1B,$1B,$1C,$1C,$1C
        DC.W    $1C,$1D,$1D,$1D,$1D,$1E,$1E,$1E,$1E,$1F
        DC.W    $1F,$1F,$1F,$20,$20,$20,$20,$21,$21,$21
        DC.W    $21,$22,$22,$22,$22,$23,$23,$23,$23,$24

                                .   .   .
                                .   .   .
                                .   .   .

        DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100,$100,$100
        DC.W    $100,$100,$100
        END
```

```
          TTL     'COSINE FUNCTION LOOK-UP TABLE'
*
          ORG     $D500
COSTBL    DC.W    $100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100,$100,$100,$100
          DC.W    $100,$100
          DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
          DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
          DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
          DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
          DC.W    $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FE,$FE

                                  .   .   .
                                  .   .   .
                                  .   .   .

          DC.W    $21,$21,$21,$20,$20,$20,$20,$1F,$1F,$1F
          DC.W    $1F,$1E,$1E,$1E,$1E,$1D,$1D,$1D,$1D,$1C
          DC.W    $1C,$1C,$1C,$1B,$1B,$1B,$1B,$1A,$1A,$1A
          DC.W    $1A,$19,$19,$19,$18,$18,$18,$18,$17,$17
          DC.W    $17,$17,$16,$16,$16,$16,$15,$15,$15,$15
          DC.W    $14,$14,$14,$14,$13,$13,$13,$13,$12,$12
          DC.W    $12,$12,$11,$11,$11,$11,$10,$10,$10,$10
          DC.W    $0F,$0F,$0F,$0F,$0E,$0E,$0E,$0E,$0D,$0D
          DC.W    $0D,$0C,$0C,$0C,$0C,$0B,$0B,$0B,$0B,$0A
          DC.W    $0A,$0A,$0A,$09,$09,$09,$09,$08,$08,$08
          DC.W    $08,$07,$07,$07,$07,$06,$06,$06,$06,$05
          DC.W    $05,$05,$05,$04,$04,$04,$04,$03,$03,$03
          DC.W    $03,$02,$02,$02,$01,$01,$01,$01,$00,$00
          DC.W    $0
          END
```

```
C     ************************************************************************
C     *                                                                      *
C     *       THIS FORTRAN PROGRAM IS USED TO SIMULATE THE GENERATOR         *
C     *       MODEL (OFF-LINE).                                              *
C     *       TRAPEZOIDAL INTEGRATION ALGORITHM (I.E.ORDER 2 RUNGE-          *
C     *       KUTTA OR MODIFIED EULER NUMERICAL METHOD) IS USED IN            *
C     *       SOLVING THE DIFFERENCE EQUATIONS OF THE SIMULATION             *
C     *       SYSTEM.                                                        *
C     *                                                                      *
C     ************************************************************************
C
      DIMENSION X6K(2,1),X6K1(2,1),X7K(2,1),X7K1(2,1),X8K(3,1),
     1          X8K1(3,1),X10K(2,1),X10K1(2,1),X11K(2,1),X11K1(2,1),
     2          DXK1(2,1),DXK3(3,1),ADXK1(2,1),ADXK3(3,1),
     3          B1(2,1),B2(2,1),B3(3,1),A1(2,2),A2(2,2),A3(3,3)
C
      T = .001389
      PI = 3.1415926
      P = .0983
      Q = .553
      VRMAX = 7.3
      VRMIN =-7.3
      SEM = .86
      SEQ = .5
      W0 = 377.0
      DELTA0 = 0.4445
      P0 = 1.09
      RA = .005
      XDD = .43
      XLL = .0
      TA = .02
      TAA = -1./TA
      TE = .8
      TEE = -1./TE
      TF = 1.0
      TFF = -1./TF
      KA = 400.0
      KE = 1.
      AKF = .03
      K = 1.0
      AJ = 10.0
      D = .08
      AJJ = -377*D/AJ
      T1 = 2.8
      T11 = -1./T1
      T3 = .15
      T33 = -1./T3
      TCH = .3
      TCHH = -1./TCH
      AI4 = -6.67
      DATA B1,B3,B2/0,1,0,0,1,0,1/
      DATA A1,A3,A2/0,-8.36,1,-7.8,0,0,-.77,1,0,-7.93,0,1,-8.12,
     1      0,-4.19,1,-8.91/
      DATA X1K,X2K,X3K,X4K,X5K,X6K,X7K,X8K,X9K,X10K,X11K,X12K/0,
```

```
     1       0,0,0,0, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0/
C
        READ(*,*) XL
        READ(*,*) RL
        READ(*,*) VREF
        APF = ATAN(XL/RL)
        ARL=-RL*377/(XL+XLL)
C
        DO 100 I=1,20000
C
C       ************************************************************************
C       *                          EXCITATION SUBSYSTEM                        *
C       ************************************************************************

        X1K = X1K1
        X2K = X2K1
        X3K = X3K1
        FB = AKF/TE/TF*X2K-X3K
        UE = VREF - VT - FB
C
        CALL MEM(DXK,X1K1,X1K,T,TAA,UE)
C
        X4K = KA/TA*X1K
        IF(X4K.GT.7.3) GOTO 10
        IF(X4K.LT.-7.3) GOTO 20
        GOTO 30
10      X4K = VRMAX
        GOTO 30
20      X4K = VRMIN
30      CALL MEM(DXK,X2K1,X2K,T,TEE,X4K)
        X5K1 = P*EXP(Q*X2K1/TE)
        US = X4K-X5K1
        CALL MEM(DXK,X2K1,X2K,T,TEE,US)
        EFD = 1/TE*X2K1
        UF = EFD*AKF
        CALL MEM(DXK,X3K1,X3K,T,TFF,UF)
C
C       ************************************************************************
C       *                     GENERATOR D-AXIS SUBSYSTEM                       *
C       ************************************************************************

        WK = (377.0+DDEL)/377.0
        X6K(1,1) = X6K1(1,1)
        X6K(2,1) = X6K1(2,1)
        X7K(1,1) = X7K1(1,1)
        X7K(2,1) = X7K1(2,1)
        X8K(1,1) = X8K1(1,1)
        X8K(2,1) = X8K1(2,1)
        X8K(3,1) = X8K1(3,1)
C
C       Efd IS THE INPUT ELEMENT

        CALL MULM(2,1,2,DXK1,X6K,X6K1,ADXK1,A1,B1,T,EFD)
C
        Y1 = .539*X6K(1,1)+.443*X6K(2,1)+.028*EFD
C
C       Id(T*k1) IS THE INPUT ELEMENT (t = T*K1)
```

```
      CALL MULM(2,1,2,DXK1,X7K,X7K1,ADXK1,A1,B1,T,AIDK1)
C
      Y2 = -(4.32*X7K(1,1)+6.68*X7K(2,1))+.74*AIDK1+.044*(AIDK1-AIDK)/T
      Y3 = Y1+Y2
C
C     Y3 IS THE INPUT ELEMENT
C
      CALL MULM(3,1,3,DXK3,X8K,X8K1,ADXK3,A3,B3,T,Y3)
C
      PSIDK1 = 8.36*X8K(1,1)+7.8*X8K(2,1)+X8K(3,1)-.225*AIDK1
C
C     ******************************************************************
C     *                    GENERATOR Q-AXIS SUBSYSTEM                  *
C     ******************************************************************
      X9K = X9K1
      X10K(1,1) = X10K1(1,1)
      X10K(2,1) = X10K1(2,1)
C
C     Iqk1 IS THE INPUT ELEMENT
C
      CALL MEM(DXK,X9K1,X9K,T,AI4,AIQK1)
C
      Y4 = -20*X9K1+4.16*AIQK1+.063*(AIQK1-AIQK)/T
C
C     Y4 IS THE INPUT ELEMENT
C
      CALL MULM(2,1,2,DXK1,X10K,X10K1,ADXK,A2,B2,T,Y4)
C
      PSIQK1 = .97*X10K1(2,1)+6.54*X10K1(1,1)-.225*AIQK1
C
C     TREATING THE MACHINE SATURATION
C
      PSIL1 = PSIL
      IF (I.LE. 4000) THEN
        PSIL = SQRT(PSIDK1**2 + PSIQK1**2)
      ELSEIF (I .GT. 4000) THEN
        PSIL = PSIL + ABS(PSIL-PSIL1)
      ENDIF
C
      IF (PSIL .LT. .7) THEN
        PSI = PSIL
        GOTO 1000
      ELSEIF (PSIL .GE. .7) THEN
        PSI1 = PSI
        AG = 0.035
        BG = 7.955
        SG = AG*EXP(BG*(PSI-.7))
        PSI = PSIL - PSI*SG
        IF (PSI .LT. PSI1) PSI = PSI1
      ENDIF
C
      PSIDK1 = PSI*PSIDK1/PSIL
      PSIQK1 = PSI*PSIQK1/PSIL
C
1000  W = W0 + DDEL
      VQK1 = -RA*AIQK1+WK*PSIDK1-(PSIQK1-PSIQK)/T/W
```

```
      VDK1 = -RA*AIDK1-WK*PSIQK1-(PSIDK1-PSIDK)/T/W
      PSIQK = PSIQK1
      PSIDK = PSIDK1
      AIDK = AIDK1
      AIQK = AIQK1
      VT = SQRT(VDK1**2+VQK1**2)
C     ********************************************************************
C     *                      GOVERNOR-TURBINE SUBSYSTEM                  *
C     ********************************************************************

      PMK = X12K*3.333
      TMK = PMK/Wk
      X11K(1,1) = X11K1(1,1)
      X11K(2,1) = X11K1(2,1)
      X12K = X12K1

C     DDEL1 IS THE INPUT ELEMENT

      CALL MEM(DXK,X11K1(1,1),X11K(1,1),T,T11,DDEL1)
C
      Y7 = X11k(1,1)*100./t1+(DDEL1-x11k(1,1)/t1)*100./t1
      UG = (P0-Y7)/T3

C     UG IS THE INPUT ELEMENT

      CALL MEM(DXK,X11K1(2,1),X11K(2,1),T,TCHH,UG)
C
      Y5 = 1./T3*(P0-Y7-X11K1(2,1))
      IF(Y5.GT..1) GOTO 145
      IF(Y5.LT.-.1) GOTO 150
      GOTO 155
145   Y5 = .1
      X11K1(2,1) = (p0*1./t3-y7*1./t3-.1)*t3
      X11K(2,1) = X11K1(2,1)
      GOTO 155
150   Y5 = -.1
      X11K1(2,1) = (p0*1./t3-y7*1./t3+.1)*t3
      X11K(2,1) = X11K1(2,1)
155   Y6 = X11K1(2,1)
      IF(Y6.GT.1.2) GOTO 160
      IF(Y6.LT..9) GOTO 165
      GOTO 170
160   Y6 = 1.2
      X11K(2,1) = Y6
      GOTO 170
165   Y6 = 0.9
      X11K(2,1) = Y6
170   CALL MEM(DXK,X12K1,X12K,T,TCHH,X11K(2,1))
      PMK1 = X12K1/TCH
      Wk1 = (377+DDEL1)/377
      TMK1 = PMK1/WK1

C     ********************************************************************
C     *                        MECHANICAL SUBSYSTEM                      *
C     ********************************************************************

      DDEL = DDEL1
```

```
      PEK1 = VT*1.0/XDD*SIN(DELTA)
      TEK1 = PEK1/WK1
      UKK = (PMK1-PEK1)*377/AJ
C
C     UKK IS THE INPUT ELEMENT
C
      CALL MEM(DXK,DDEL1,DDEL,T,AJJ,UKK)
C
      DEL=DEL+(DDEL1+DDEL)*T/2
      DELTA=DEL+DELTA0
      TEK = TEK1
C
C     ************************************************************************
C     *                      INVERSE PARK TRANSFORMATION                     *
C     ************************************************************************
C
      TI = T/4
      DO 200 j = 0,3
        FIK1 = (W0+DDEL1)*T*(I-1) + 2*DELTA + (W0+DDEL1)*TI*J
205     IF (FIK1) 201,202,202
201     FIK1 = FIK1 + 2*PI
        GOTO 205
202     IF (FIK1 - 2*PI) 203,204,204
204     FIK1 = FIK1 - 2*PI
        GOTO 202
203     IF (FIK1 .GT. PI/2) GOTO 206
        SS = SIN(FIK1)
        CC = COS(FIK1)
        GOTO 210
206     IF (FIK1 .GT. PI) GOTO 207
        SS = SIN (PI - FIK1)
        CC = COS (PI - FIK1)
        GOTO 210
207     IF (FIK1 .GT. 3*PI/2) GOTO 208
        SS = -SIN(FIK1 - PI)
        CC = -COS(FIK1 - PI)
        GOTO 210
208     SS = - SIN(2*PI - FIK1)
        CC = COS(2*PI - FIK1)
210     CALL IPT(SS,CC,VT,VAG,VBG,VCG)
200   CONTINUE
C
C     ************************************************************************
C     *                              LOAD MODEL                              *
C     ************************************************************************
C
      WT = (377+DDEL1)*T*I
      ZL = SQRT((XL+XLL)**2+RL**2)
      AIA = VT/ZL*SIN(WT-APF)-VT/ZL*SIN(-APF)*EXP(ARL*T*I)
      AIB = VT/ZL*SIN(WT-APF-2./3.*PI)-VT/ZL*SIN(-APF-2./3.*PI)
     1       *EXP(ARL*T*I)
      AIC = VT/ZL*SIN(WT-APF+2./3.*PI)-VT/ZL*SIN(-APF+2./3.*PI)
     1       *EXP(ARL*T*I)
C
C     ************************************************************************
C     *                           PARK TRANSFORMATION                        *
C     ************************************************************************
C
```

```
        SITA = (377+DDEL1)*T*I+DELTA
2       IF(SITA.GT.0) GOTO 5
        SITA = SITA+2*PI
        GOTO 2
5       IF(SITA.LT.2*PI) GOTO 15
        SITA = SITA-2*PI
        GOTO 5
15      IF(SITA.GT.PI/2) GOTO 25
        S = SIN(SITA)
        C = COS(SITA)
        GOTO 105
25      IF (SITA.GT.PI) GOTO 35
        S = SIN(PI-SITA)
        C = -COS(PI-SITA)
        GOTO 105
35      IF (SITA.GT.3*PI/2) GOTO 45
        S = -SIN(SITA-PI)
        C = -COS(SITA-PI)
        GOTO 105
45      S = -SIN(2*PI-SITA)
        C = COS(2*PI-SITA)
105     AIDK1 = 2./3.*(AIA*C+AIB*(-.5*C+.866*S)+AIC*(-.5*C-.866*S))
        AIQK1 = 2./3.*(AIA*S+AIB*(-.5*S-.866*C)+AIC*(-.5*S+.866*C))
        AIGK1 = SQRT(AIDK1**2+AIQK1**2)
C       ************************************************************************

        IF(25*K.EQ.I) GOTO 40
        GOTO 100
40      TK=T*K*25
50      FORMAT(2X,5F14.6)
        PRINT 50,TK,VT,EFD,PSIDK1,PSIQK1
        WRITE(1,50)TK,VT,EFD,PSIDK1,PSIQK1
        K=K+1
C
100     CONTINUE

        END
        ************************************************************************
        *       SUBROUTINE OF MODIFIED EULER METHOD,JUST FOR SCALAR            *
        ************************************************************************

        SUBROUTINE MEM(DXK,XK1,XK,T,A,U)
        DXK=A*XK+U
        XK1=XK+T/2*(2*DXK+T*A*DXK)
        RETURN
        END

        ************************************************************************
        *          SUBROUTINE OF MULTIPLICATION OF MATRIX (MEM)                *
        ************************************************************************

        SUBROUTINE MULM(L,M,N,DXK,XK,XK1,ADXK,A,B,T,UU)
        DIMENSION DXK(L,M),ADXK(L,M),XK(L,M),XK1(L,M),A(L,N),B(L,M)
        DO 10 I=1,L
        DO 10 J=1,M
        DXK(I,J)=0
```

```
      DO 20 K=1,N
20    DXK(I,J)=DXK(I,J)+A(I,K)*XK(K,J)
      DXK(I,J)=DXK(I,J)+B(I,J)*UU
      ADXK(I,J)=0
      DO 25 K=1,N
25    ADXK(I,J)=ADXK(I,J)+A(I,K)*DXK(K,J)*T
      XK1(I,J)=XK(I,J)+T/2*(2*DXK(I,J)+ADXK(I,J))
10    CONTINUE
      RETURN
      END
C
C     ************************************************************************
C     *               SUBROUTINE OF INVERSE PARK TRANSFORMATION              *
C     ************************************************************************
C
      SUBROUTINE IPT(SS,CC,V,Va,Vb,Vc)
      Va = V * SS
      Vb = -Va/2 - 1.732/2*V*CC
      Vc = -Va -Vb
      RETURN
      END
```

```
C     ************************************************************
C     *                                                          *
C     *   THIS PROGRAM IS USED TO BUILD EXPONENTIAL FUNCTION     *
C     *                      LOOK-UP TABLE                       *
C     *              (X RANGES FROM 0.0 TO 5.0)                  *
C     *                                                          *
C     ************************************************************

      CHARACTER*1 HEX(16),A(2),B(20),C(2)
      DATA HEX/'0','1','2','3','4','5','6','7','8','9',
     1         'A','B','C','D','E','F'/

      OPEN(1,FILE='EXPTBL.SRC',STATUS='UNKNOWN')
5     FORMAT(///9X,'ORG',5X,6H$10000)
      WRITE(1,5)
      PRINT 5
      PRINT 10
      WRITE(1,10)
10    FORMAT(1X,'EXPTBL',2X,'DC.W',4X,4H$100)
C
      K=1
      DO 20 I=1,2500
      X=I*.002
      Y=EXP(X)
C
      CALL TABLE(Y,A,C,HEX)
C
      IK1=4*(I-(K-1)*5)-3
      IK2=IK1+1
      IK3=IK2+1
      IK4=IK3+1
      B(IK1)=C(1)
      B(IK2)=C(2)
      B(IK3)=A(1)
      B(IK4)=A(2)
C
      IF(I.EQ.5*K) GOTO 40
      GOTO 20
40    WRITE(1,30) (B(J),J=1,20)
      PRINT 30,(B(J),J=1,20)
30    FORMAT(9X,'DC.W',4X,4('$'4A1',),'$'4A1)
      K=K+1
C
20    CONTINUE
      CLOSE(1)
      END
C     ************************************************************
      SUBROUTINE TABLE (Y,A,C,HEX)
      CHARACTER*1 HEX(16),A(2),C(2)
      JY=Y
      IY=JY/16
      ID0=JY-IY*16
      ID1=IY
C
      R=Y-JY
```

```
        IR=R*16
        Z=R*16-IR
        IZ=Z*16
        T=Z*16-IZ
        IT=T*16
C
        IF(IT.LT.8) GOTO 25
        IF(IZ.EQ.15) GOTO 45
        IZ=IZ+1
        GOTO 25
C
45      IZ=0
        IF(IR.EQ.15) GOTO 55
        IR=IR+1
        GOTO 25
C
55      IR=0
        IF(ID0.EQ.15) GOTO 65
        ID0=ID0+1
        GOTO 25
C
65      ID0=0
        ID1=ID1+1
C
25      A(2)=HEX(IZ+1)
        A(1)=HEX(IR+1)
        C(2)=HEX(ID0+1)
        C(1)=HEX(ID1+1)
C
        RETURN
        END
```

```
C       ********************************************************
C       *                                                      *
C       *      THIS PROGRAM IS USED TO BUILD SQUARE ROOT       *
C       *              FUNCTION LOOK-UP TABLE                  *
C       *            (X RANGES FROM 0.0 TO 16)                 *
C       *                                                      *
C       ********************************************************
C
        INTEGER U
        CHARACTER*1 HEX(16),A(2),B(20)
        DATA HEX/'0','1','2','3','4','5','6','7','8','9',
     1          'A','B','C','D','E','F'/
        OPEN(1,FILE='SQRTBL.SRC',STATUS='UNKNOWN')
5       FORMAT(///9X,'ORG',5X,5H$B500)
        WRITE(1,5)
        PRINT 5
        U=0
        PRINT 10,U
        WRITE(1,10)U
10      FORMAT(1X,'SQRTBL',2X,'DC.W',4X,'$',I1)
        K=1
        DO 20 I=1,4000
        X=I*.004
        Y=SQRT(X)
        IF(I.GT.250.AND.I.LE.1000) GOTO 60
        IF(I.GT.1000.AND.I.LE.2250) GOTO 110
        IF(I.GT.2250) GOTO 140
C
        CALL TABLE(Y,A,HEX)
        IK=2*(I-(K-1)*10)-1
        IKK=IK+1
        B(IK)=A(1)
        B(IKK)=A(2)
        IF(I.EQ.10*K) GOTO 40
        GOTO 20
40      WRITE(1,30) (B(J),J=1,20)
        PRINT 30,(B(J),J=1,20)
30      FORMAT(9X,'DC.W',4X,9('$'2A1',')','$'2A1)
C
        GOTO 35
C
60      Y=Y-1
        CALL TABLE(Y,A,HEX)
        IK=2*(I-(K-1)*10)-1
        IKK=IK+1
        B(IK)=A(1)
        B(IKK)=A(2)
        IF(I.EQ.10*K) GOTO 70
        GOTO 20
70      WRITE(1,80)(B(J),J=1,20)
        PRINT 80,(B(J),J=1,20)
80      FORMAT(9X,'DC.W',4X,9('$1'2A1',')','$1'2A1)
        GOTO 35
110     Y=Y-2
        CALL TABLE(Y,A,HEX)
```

```
        IK=2*(I-(K-1)*10)-1
        IKK=IK+1
        B(IK)=A(1)
        B(IKK)=A(2)
        IF(I.EQ.10*K) GOTO 120
        GOTO 20
120     WRITE(1,130)(B(J),J=1,20)
        PRINT 130,(B(J),J=1,20)
130     FORMAT(9X,'DC.W',4X,9('$2'2A1',),'$2'2A1)
        GOTO 35
C
140     Y=Y-3
        CALL TABLE(Y,A,HEX)
        IK=2*(I-(K-1)*10)-1
        IKK=IK+1
        B(IK)=A(1)
        B(IKK)=A(2)
        IF(I.EQ.10*K) GOTO 150
        GOTO 20
150     WRITE(1,160)(B(J),J=1,20)
        PRINT 160,(B(J),J=1,20)
160     FORMAT(9X,'DC.W',4X,9('$3'2A1',),'$3'2A1)
C
35      K=K+1
        GOTO 20
95      PRINT 96
        WRITE(1,96)
96      FORMAT(9X,'DC.W',4X,4H$100)
        K=K+1
20      CONTINUE
21      FORMAT(9X,3HEND)
        WRITE(1,21)
        PRINT 21
        END
C       ************************************************************
        SUBROUTINE TABLE(Y,A,HEX)
        CHARACTER*1 HEX(16),A(2)
        IY=Y*16
        Z=Y*16-IY
        IZ=Z*16
        T=Z*16-IZ
        IT=T*16
        IF(IT.LT.8) GOTO 25
        IF(IZ.EQ.15) GOTO 45
        IZ=IZ+1
        GOTO 25
45      IZ=0
        IY=IY+1
25      A(1)=HEX(IY+1)
        A(2)=HEX(IZ+1)
        RETURN
        END
```

```
C     ********************************************************
C     *                                                      *
C     *    THIS PROGRAM IS USED TO BUILD SINE FUNCTION       *
C     *                  (LOOK-UP TABLE)                     *
C     *         (X RANGES FROM 0.0 TO 1.57 RADIAN)           *
C     *                                                      *
C     ********************************************************

      INTEGER U
      CHARACTER*1 HEX(16),A(2),B(20)
      DATA HEX/'0','1','2','3','4','5','6','7','8','9',
     1        'A','B','C','D','E','F'/
      OPEN(1,FILE='SINTBL.SRC',STATUS='UNKNOWN')
      U=00
5     FORMAT(///9X,'ORG',5X,6H$11500)
      WRITE(1,5)
      PRINT 5
      PRINT 10,U
      WRITE(1,10)U
10    FORMAT(1X,'SINTBL',2X,'DC.W',4X,'$',I1)
      K=1
      DO 20 I=1,1500
      X=I*.001
      Y=SIN(X)
      CALL TABLE(Y,A,HEX)
      IK=2*(I-(K-1)*10)-1
      IKK=IK+1
      B(IK)=A(1)
      B(IKK)=A(2)
      IF(I.EQ.10*K) GOTO 40
      GOTO 20
40    WRITE(1,30) (B(J),J=1,20)
      PRINT 30,(B(J),J=1,20)
30    FORMAT(9X,'DC.W',4X,9('$'2A1',')'$'2A1)
C
      K=K+1
20    CONTINUE
      PRINT 31
      WRITE(1,31)
31    FORMAT(9X,'DC.W',4X,31H$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF)
      DO 33 L=1,12
      PRINT 32
      WRITE(1,32)
32    FORMAT(9X,'DC.W',4X,24H$100,$100,$100,$100,$100)
C
33    CONTINUE
      PRINT 34
      WRITE(1,34)
34    FORMAT(9X,'DC.W',4X,14H$100,$100,$100)
      WRITE(1,35)
35    FORMAT(9X,3HEND)
      CLOSE(1)
      END
C     ********************************************************
```

```
      SUBROUTINE TABLE(Y,A,HEX)
      CHARACTER*1 HEX(16),A(2)
      IY=Y*16
      Z=Y*16-IY
      IZ=Z*16
      T=Z*16-IZ
      IT=T*16
      IF(IT.LT.8) GOTO 25
      IF(IZ.EQ.15) GOTO 45
      IZ=IZ+1
      GOTO 25
45    IZ=0
      IY=IY+1
25    A(1)=HEX(IY+1)
      A(2)=HEX(IZ+1)
      RETURN
      END
```

```
C     ************************************************************
C     *                                                          *
C     *     THIS PROGRAM IS USED TO BUILD COSINE FUNCTION        *
C     *                    LOOK-UP TABLE                         *
C     *           (X RANGES FROM 0.0 TO 1.57 RADIAN)             *
C     *                                                          *
C     ************************************************************
C
      CHARACTER*1 HEX(16),A(2),B(20)
      DATA HEX/'0','1','2','3','4','5','6','7','8','9',
     1         'A','B','C','D','E','F'/
      OPEN(1,FILE='COSTBL.SRC',STATUS='UNKNOWN')
5     FORMAT(///9X,'ORG',5X,6H$12500)
      WRITE(1,5)
      PRINT 5
      U=100
      PRINT 10,U
      WRITE(1,10) U
10    FORMAT(1X,'COSTBL'2X,'DC.W',4X,'$',I3)
      DO 33 L=1,12
      PRINT 32
      WRITE(1,32)
32    FORMAT(9X,'DC.W',4X,24H$100,$100,$100,$100,$100)
33    CONTINUE
      PRINT 34
      WRITE(1,34)
34    FORMAT(9X,'DC.W',4X,9H$100,$100)
      PRINT 31
      WRITE(1,31)
31    FORMAT(9X,'DC.W',4X,31H$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF)
      K=8
      DO 20 I=71,1571
      X=I*.001
      Y=COS(X)
      CALL TABLE(Y,A,HEX)
      IK=2*(I-(K-1)*10)-1
      IKK=IK+1
      B(IK)=A(1)
      B(IKK)=A(2)
      IF(I.EQ.10*K) GOTO 40
      GOTO 20
40    WRITE(1,30) (B(J),J=1,20)
      PRINT 30,(B(J),J=1,20)
30    FORMAT(9X,'DC.W',4X,9('$'2A1',')'$'2A1)
C
      K=K+1
20    CONTINUE
      M=00
      PRINT 50,M
      WRITE(1,50)M
50    FORMAT(9X,'DC.W',4X,'$',I1)
51    FORMAT(9X,3HEND)
```

```
      PRINT 51
      WRITE(1,51)
      CLOSE(1)
      END
C
C     ************************************************************
      SUBROUTINE TABLE(Y,A,HEX)
      CHARACTER*1 HEX(16),A(2)
      IY=Y*16
      Z=Y*16-IY
      IZ=Z*16
      T=Z*16-IZ
      IT=T*16
      IF(IT.LT.8) GOTO 25
      IF(IZ.EQ.15) GOTO 45
      IZ=IZ+1
      GOTO 25
45    IZ=0
      IY=IY+1
25    A(1)=HEX(IY+1)
      A(2)=HEX(IZ+1)
      RETURN
      END
```

Note: The software 'Error Model Generator' is programmed by Nitin Vichare, research assistant of Power System Research Center, Dept. of Elec. Eng. of VPI.

# Appendix C

# Nomenclature

All quantities are in per unit unless noted.

| | |
|---|---|
| $V_T$ | Generator terminal voltage |
| $V_{ref}$ | Regulator reference voltage setting |
| $E_{fd}$ | Exciter output voltage |
| $E_{fdmax}$ | Exciter output voltage ceiling |
| $E_{fdq}$ | $0.75 \times E_{fdmax}$ |
| $E_{fdl}$ | $E_{fd}$ at full load |
| $T_r$, $T_F$ | Exciter model time constants |
| $T_e$, $T_a$ | See Figure 3 and Table 1. |
| $K_F$, $K_a$ | Exciter model gains |
| $K_e$ | See Figure 3 and Table 1. |
| $V_R$ | Regulator output voltage |
| $V_{Rmax}$ | Maximum value of $V_R$ |

| | |
|---|---|
| $V_{Rmin}$ | Minimum value of $V_R$ |
| $S_E$ | Exciter saturation function |
| $S_{Em}$ | $S_E$ at $E_{fdmax}$ |
| $S_{Eq}$ | $S_E$ at $E_{fdq}$ |
| $S_{Efl}$ | $S_E$ at $E_{fdl}$ |
| $P$, $Q$ | Coefficients of exciter saturation function |
| $s$ | Laplace operator |
| $i_a$, $i_b$, $i_c$ | Armature currents, phase components |
| $V_a$, $V_b$, $V_c$ | Armature voltages, phase components |
| $i_d$, $i_q$, $i_0$ | Armature currents, d, q axis and zero sequence components |
| $V_d$, $V_q$, $V_0$ | Armature voltages, d, q axis and zero sequence components |
| $\omega_{ref}$ | Base angular speed of rotor |
| $\omega_s$ | Synchronous speed $\omega_s = \omega_{ref}$ |
| $\Delta\omega$ | Rotor speed oscillation about $\omega_{ref}$ |
| $\omega$ | Rotor speed $(\omega_{ref} + \Delta\omega)$ |
| $\delta_{ref}$ | Reference $\delta$ |
| $\Delta\delta$ | Machine torque angle oscillation about $\delta_{ref}$ |
| $\delta$ | Machine torque angle |
| $\theta$ | Cumulative rotor angle $(\omega t + \delta)$ |
| $\theta_0$ | Rotor angle at time t = 0 |
| $\phi$ | $(\theta + \delta)$ |
| $P(\theta)$ | Park's transformation matrix |
| $\overline{E}$ | Machine EMF |
| $\overline{V}$ | Machine constant terminal voltage |
| $e_{fd}$ | Generator field voltage on MVA base |
| $\psi$ | Armature flux linkage |
| $\psi_l$ | Armature flux linkage when saturation effect is neglected |
| $\psi_d$, $\psi_q$ | Armature flux linkages, d and q axis components |

| | |
|---|---|
| $\psi_a,\ \psi_b,\ \psi_c$ | Armature flux linkages, phase components |
| $\psi_F,\ \psi_D,\ \psi_Q$ | Rotor flux linkages |
| $L_{aa},\ L_{bb},\ L_{cc}$ | Phase-winding self inductances |
| $L_{ab},\ L_{ba},\ L_{bc}$ | Phase-winding mutual inductances |
| $L_{cb},\ L_{ca},\ L_{ac}$ | |
| $L_{FF},\ L_{DD},\ L_{QQ}$ | Rotor-winding self inductances |
| $L_{FD},\ L_{DF},\ L_{FQ}$ | Rotor-winding mutual inductances |
| $L_{QF},\ L_{DQ},\ L_{QD}$ | |
| $L_{aF},\ L_{Fa},\ L_{bF}$ | Mutual inductances between the phase |
| $L_{Fb},\ L_{cF},\ L_{Fc}$ | windings and the field winding |
| $L_{aD},\ L_{Da},\ L_{bD}$ | Mutual inductances between the phase |
| $L_{Db},\ L_{cD},\ L_{Dc}$ | and the damper windings |
| $L_{aQ},\ L_{Qa},\ L_{bQ}$ | |
| $L_{Qb},\ L_{cQ},\ L_{Qc}$ | |
| $L_d,\ L_q,\ L_0$ | Stator inductances, d, q axis and zero sequence components |
| $L_{ad},\ L_{aq}$ | Stator-rotor mutual inductances |
| $\psi_{ad},\ \psi_{aq}$ | Stator-rotor mutual flux linkages |
| $L_l$ | Armature leakage inductances |
| $L_{fd}$ | Field leakage inductances |
| $R_{nd},\ L_{nd},\ R_{nq},\ L_{nq}$ | Damper resistances and leakage inductances |
| $R_{fd}$ | Field resistance |
| $r_a,\ r_b,\ r_c$ | Phase-winding resistances |
| $r_F$ | Field-winding resistance |
| $r_D,\ r_Q$ | Damper-winding resistances |
| $V_F$ | Field-winding voltage |
| $V_D,\ V_Q$ | Damper-winding voltages |
| $i_F$ | Field-winding current |
| $i_D,\ i_Q$ | Damper-winding currents |

| | |
|---|---|
| $Z_{fkd}(s)$, $Z_{fd}(s)$ | Operational impedances of the solid |
| $Z_{kd}(s)$, $Z_{kq}(s)$ | rotor machine detailed equivalent circuit |
| $A(s)$, $B(s)$ | Transfer functions in the time domain model |
| $Z_{eq}(s)$ | Impedances and inductances obtained by algebraic reduction of |
| $L_{eq}$, $L_{kq}$ | $Z_{fkd}(s)$, $Z_{kd}(s)$, $Z_{fd}(s)$ and $Z_{kq}(s)$ . |
| $R_{eq0}$, $R_{kq0}$ | Resistive components of $Z_{eq}(s)$ and $Z_{kq}(s)$, respectively, at extreme low slip |
| $S_G$ | Synchronous machine saturation function |
| $S_{G1.0}$ | $S_G$ at $V_T = 1.0$ p.u. |
| $S_{G1.2}$ | $S_G$ at $V_T = 1.2$ p.u. |
| $A_G$, $B_G$ | Constants of machine saturation function |
| $V_\Delta$ | $(V_T - 0.7)$ |
| $P_0$ | Initial (time = $0^-$) mechanical power |
| $P_{GV}$ | Power at gate or valve outlet |
| $T_1$, $T_2$, $T_3$ | Governor model time constants<br>See Figure 16 and Table 4. |
| $T_{ch}$ | Steam chest time constant |
| $K$ | Total effective speed-governing system gain |
| $\dot{P}_{up}$, $\dot{P}_{down}$ | Limits on Rate of change of power imposed by control valve rate limits |
| $P_{max}$, $P_{min}$ | Power limits imposed by valve or gate travel |
| $P_m$ | Mechanical power |
| $P_e$ | Electrical or Electromagnetic power |
| $T_m$ | Mechanical input torque to the synchronous machine in N-m |
| $T_e$ | Electrical or electromagnetic torque in N-m |
| $J$ | Lumped moment of inertia in $kg - m^2$ |
| $D$ | Differential damping coefficient in Joule-sec. |
| $\delta_{eq}$ | $\delta$ at equilibrium |
| $x_l$ | Transmission line reactance |

| | |
|---|---|
| $x'_d$ | Machine transient reactance |
| $T$ | Sampling period |

# Vita

Yujie Lu was born on January 1st, 1958, in Beijing, China. She attended Beijing 26th High School, and graduated in January 1976. After two and half years of employment at Beijing Power Supply Bureau, she entered the Department of Electrical Engineering, Huazhong University of Science and Technology, where she received a Bachelor of Engineering degree in Electrical Engineering in July 1982. She worked at Scientific and Technical Information Institute of Electric Power, Ministry of Electric Power of China, Beijing, China, from August 1982 to May 1987. She joined the Department of Electrical Engineering, Virginia Polytechnic Institute and State University as a graduate student in December 1987. She is a member of the IEEE.