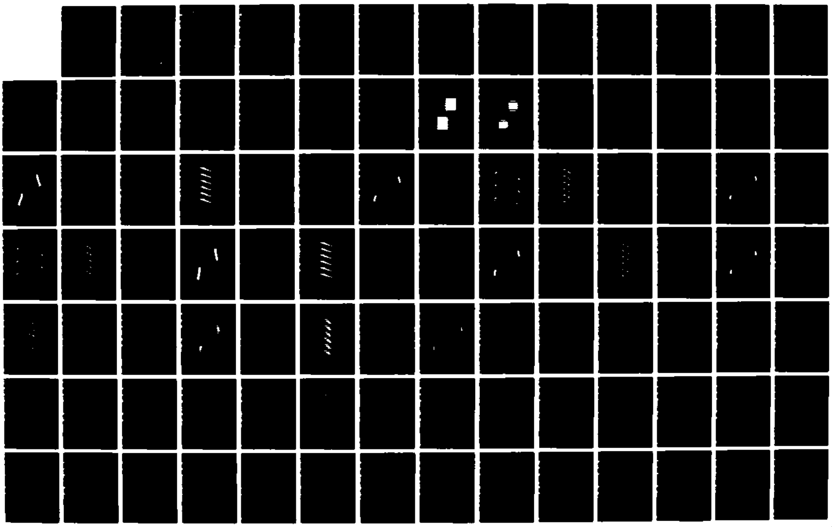
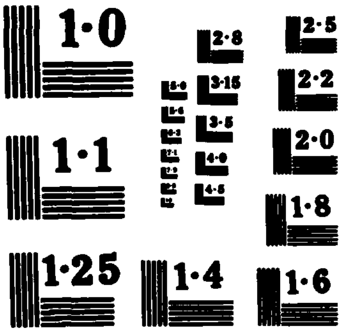
AD-A144 216 HARMONIC CONTROL TO REDUCE TORQUE PULSATIONS IN BRUSHLESS DC MOTOR DRIVES(U) KENTUCKY UNIV RESEARCH FOUNDATION LEXINGTON J J CATHEY MAR 84 1/2

UNCLASSIFIED AFOSR-TR-84-0607 AFOSR-83-0189 F/G 20/3 NL




1·0
2·8
2·5
3·15
2·2
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

AFOSR - [redacted] - 83 - 0189

HARMONIC CONTROL TO REDUCE TORQUE PULSATIONS IN BRUSHLESS DC MOTOR DRIVES

Jimmie J. Cathey
University of Kentucky Research Foundation
Lexington, KY 40506

March 1984

Final report for period May 1983 - January 1984

Approved for public release; Distribution unlimited

Prepared for
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
BOLLING AIR FORCE BASE, D.C. 20332

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 84-0607 ~~AFOSR-TR-83-0189~~	2. GOVT ACCESSION NO. AD-A144 216	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) HARMONIC CONTROL TO REDUCE TORQUE PULSATIONS IN BRUSHLESS DC MOTOR DRIVES		5. TYPE OF REPORT & PERIOD COVERED Final Report May 1983 - January 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jimmie J. Cathey		8. CONTRACT OR GRANT NUMBER(s) AFOSR - 83 - 0189
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Kentucky Research Foundation Kinkead Hall Lexington, KY 40506		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2305/D9 61102F
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research (NE) Building 410 Bolling AFB, D.C. 20332		12. REPORT DATE March 1984
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Brushless DC Motors
Cycloconverter Drives
Torque Pulsations
Direct Current (DC)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The brushless ~~DC~~ machine theoretically offers wide speed range torque characteristics like unto the commutator DC machine. However, in brushless DC motor drive systems there exists a performance deficiency in that at near zero speeds driven mechanical loads can respond to the pulsating component of developed torque when simple rotor position-activated switching is utilized. This report analytically develops a pulse width modulation control philosophy that reduces torque pulsations to an acceptable level.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONTINUED)

LIST OF TABLES

SECTION I

INTRODUCTION

1.0 BACKGROUND

1.1 Orientation

Because of their inherent flexibility of control, reliability, and efficiency, uses of electromechanical actuators and electric drives are finding increased interests in aircraft applications. Three areas of technology advancement over the past decade are largely responsible for placing electromechanical energy converters into a favorable position when compared to hydraulic motors or actuators:

(a). Development of rare earth permanent magnet (PM) motors with inherent high efficiency and high power to weight ratios.

(b). Advancements in power level solid-state devices with high switching speeds.

(c). Emergence of microprocessors allowing control capability and sophistication far surpassing that of analog systems while reducing the volume occupied.

The conventional or commutator DC machine performance characteristics in the areas of speed control and position control are highly desirable. The brushless DC machine has wide range torque-speed characteristics like unto the commutator DC machine without the commutator-brush maintenance problems. In addition, the brushless DC machine with a permanent magnet rotor has certain superior features to the conventional DC machine:

(a). Field excitation is eliminated, which removes the complexity of supplying power to a rotating member. Also, machine efficiency is increased due to absence of field excitation losses.

(b). Higher speed design is possible for PM rotors than is feasible with wound rotors permitting increased gear ratios, which leads to substantial reduction in electric machine power to weight ratios.

(c). Thermal transfer characteristics are improved since the bulk of the losses (ohmic and core losses) are generated within the stationary member, allowing efficient implementation of fluid cooling.

1.2 Identification of Problem

Brushless DC motor performance is reported in the literature, but it concentrates on the nature of instantaneous voltage and current along with average values of developed torque [1-9]. These works typically describe systems that do not operate continuously at near zero speed, and thus, only average value of torque is of concern. However, the brushless DC machine inherently has oscillatory components of instantaneous torque at all frequencies that are integer multiples of six times the electrical radian of the stator impressed voltage. Since stator frequency is directly related to rotor position, these oscillatory components are in the frequency range of mechanical system response at low speed values.

Published works relating to performance of brushless DC machines have not analyzed cases of position or sustained near zero speed operation; thus, pulsating torque components have been considered of no significant consequence. However, Williamson et al [10] have mentioned that torque capability deteriorates at low speeds, and Demerdash and Nehl have shown some instantaneous developed torque and power wave forms [11] without comment on the oscillatory components.

Widespread usages of brushless DC motor drive systems in low-speed and position-control actuator applications are contingent upon development of control philosophies and hardware realizations of power conditioning arrangements that allow bidirectional power flow while resulting in instantaneous motor developed torques that are free of harmonics in the range of response for coupled mechanical loads.

1.3 Consideration of Alternative Technologies

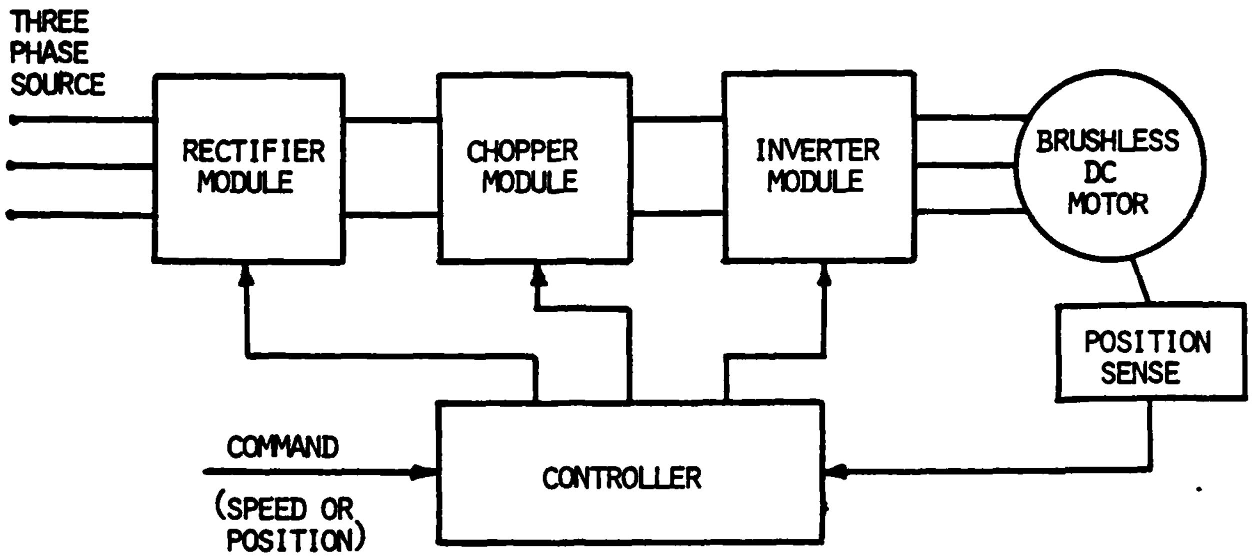
There are two basic power electronic circuits that are used as power conditioning links to couple brushless dc motor drives to a high frequency, three-phase AC source:

(a) DC link inverter

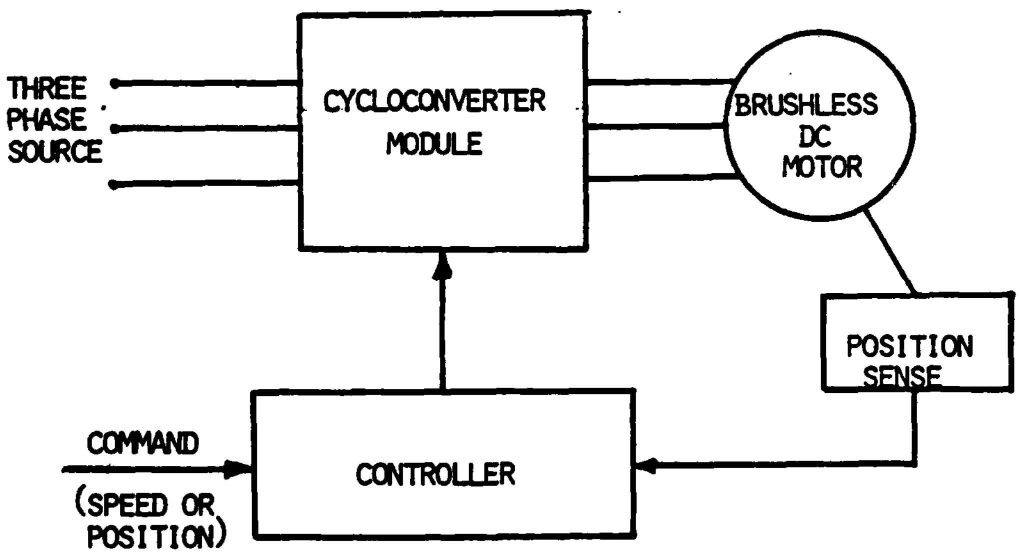
(b) Cycloconverter link

Power circuits of these two basic approaches are illustrated in functional block form by Figure 1. Variations of each arrangement are made according to the control philosophy adopted to satisfy required motor performance. Further, for low voltage (500 V or less), low current (200 A or less) applications, power circuits can be synthesized with transistors for controlled switching elements, while for high power level applications silicon controlled rectifiers (SCRs) must be used for controlled switching elements. For this study, SCR switching elements are assumed.

1.3.1 <u>Operation for Speeds Not Near Zero</u>. For the case of a DC link inverter driving a brushless DC motor that does not operate at near zero



(a)



(b)

Figure 1. Alternative Power Circuits
(a) DC Link Inverter
(b) Cycloconverter Link

speed, the rectifier may be a diode bridge while the chopper is used to vary magnitude of DC voltage applied to the inverter input terminals; or, the chopper may be eliminated and a phase-controlled converter used as the rectifier to vary inverter input voltage.

For the case of a cycloconverter, the necessity of a DC link does not exist, and thus, there is only one power conditioning module interconnecting the three-phase AC source with the brushless DC motor. Generally, the cycloconverter may be realized as either a midpoint or a full-bridge arrangement and either phase-control or synchronous envelope operation implemented.

1.3.2 Operation for Speeds Near Zero. For near zero speed operation of a DC link inverter for high power level systems, the counter emf of each motor phase is small enough so that natural commutation of the inverter SCRs is prohibited. In such a case, the chopper is used to reduce inverter terminal voltage to zero sufficiently long to accomplish commutation of the inverter SCRs [3]. Since the chopper must be controlled so as to enhance inverter commutation, the rectifier must be a phase-controlled converter to permit necessary inverter input voltage magnitude control.

For near zero speed operation of a phase-controlled cycloconverter using circulating current free mode, discontinuous load current tends to occur, leading to increase in load current harmonics. Continuous current can be restored by changing to circulating current mode of control at the expense of increased losses due to circulating reactive current [12]. Control is generally simplified at fractional hertz frequency operation by use of synchronous envelope control with circulating current free mode.

1.3.3 Operation for Regenerative Power Flow. For the case of a DC link inverter, regenerative power flow requires further power circuit modification. Gating signals to inverter SCRs are suppressed and the associated shunting diodes function to form a full-bridge diode rectifier. A switching circuit is introduced in the DC link to reverse polarity of the voltage appearing at the DC terminals of the rectifier module. The rectifier must be a phase-controlled converter operated in the synchronous inversion mode. Reversal of motor developed torque requires three coordinated control actions: phase forward and suppression of inverter SCRs; polarity reversal switch activation; and, phase forward of phase-controlled converter SCRs.

Regenerative power flow using the cycloconverter link is introduced by simply delaying SCR firing angles beyond 90°. Rate of change of motor developed torque from a positive to a negative value is controlled by the rate at which the SCR firing angles are changed. Thus, the nature of transition from motoring to regeneration is determined by a single control action leading to smooth change with minimum time delay.

2.0 OBJECTIVE

Of the presently available power conditioning technologies for linking the brushless DC motor to a high frequency polyphase AC source in a near zero speed or actuator application, the cycloconverter using synchronous envelope control with circulating free current mode of operation appears most promising in that it requires no external commutation circuitry and it can be smoothly and quickly changed from motoring to regeneration by a single control action. This arrangement using a mid-point, three-pulse cycloconverter is chosen for study with pulse width modulation (PWM) control to eliminate those harmonics from the output waveform that lead to harmonics in the developed torque in the range to which coupled mechanical loads can respond.

3.0 SCOPE

The following tasks were carried out to accomplish the objective:

(a). Determine an appropriate mathematical model of the system for near zero speed operation.

(b). Determine the current harmonics to be eliminated in order to remove undesireable pulsations in motor developed torque.

(c). Develop a modulation function to use in control of phase voltage for eliminating the undesired current harmonics.

(d). Calculate motor performance over a wide speed range with and without the PWM phase voltage.

(e). Compare ohmic losses with and without PWM to assess the effect of harmonic elimination on efficiency.

This page left blank

SECTION II

PROCEDURE

1.0 SYSTEM DESCRIPTION

Typical values for motor parameters were obtained by ratio of known values reported in the literature [11] where the results are given in Table 1. The motor is wye connected with a maximum design speed of 45000 rpm.

TABLE 1. MOTOR PARAMETERS

Parameter	Symbol	Value
No. poles	p	4
EMF constant	K	0.0225 V·s/rad
Stator resistance	R_a	0.4 Ω
Stator leakage inductance	L_a	25 μH

Characteristics of the balanced three-phase source are listed in Table 2.

TABLE 2. SOURCE CHARACTERISTICS

Quantity	Value
Phases	3
Line voltage	138 V
Frequency	7950 Hz

The power circuit schematic of a midpoint, three-pulse cycloconverter linking the high frequency, three-phase AC source to the brushless DC motor is depicted by Figure 2. Use of the switch in the midpoint or neutral line will be discussed later.

2.0 ASSUMPTIONS

When analyzing rare earth permanent magnet machines with non-magnetic retaining rings, it has been found that position dependence and interphase

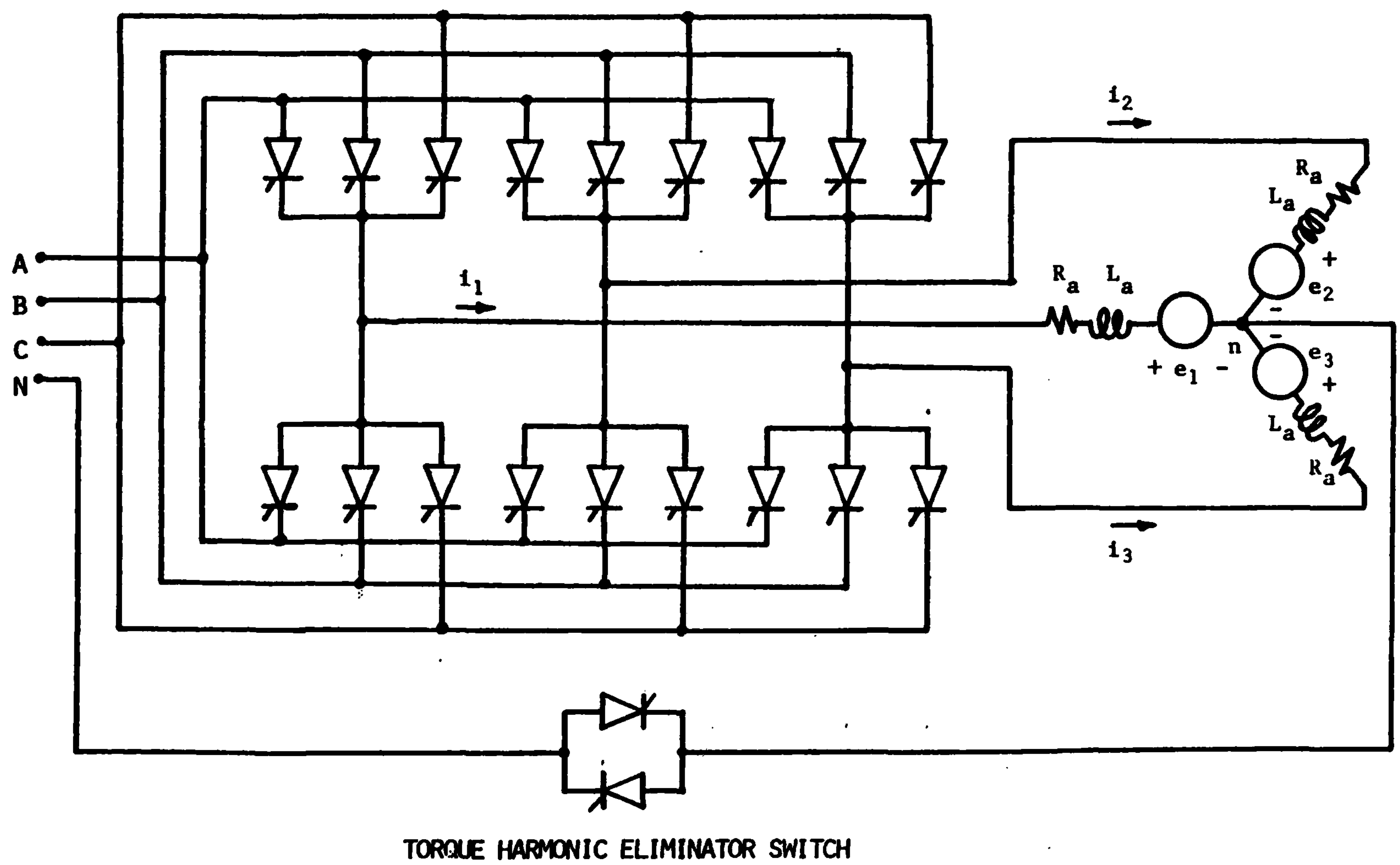


Figure 2. Schematic of Power Circuit

coupling of machine inductances can be neglected [11]. Such an approximation leads to decoupled equations that can be used in networks formed by addition of power conditioning circuitry with minimum difficulty. Motor counter emfs are taken to be sinusoidal. Further, since the study of this report is concerned with low speed operation, the commutation overlap is very small compared with a period of the motor current wave form and can be neglected.

The SCRs are modelled by a 0.02 ohm forward resistance (2 V at 100 A) and a blocking resistance described by 1000 i^2 ohms.

3.0 MATHEMATICAL MODEL

Referring to Figure 2, using the above assumptions, and applying Kirchhoff's voltage law results in the following set of simultaneous differential equations to describe the motor electrical performance:

$$\frac{di_1}{dt} = \frac{1}{L_a}(-i_1R + v_{an} - e_1) \qquad (1)$$

$$\frac{di_2}{dt} = \frac{1}{L_a}(-i_2R + v_{bn} - e_2) \qquad (2)$$

$$\frac{di_3}{dt} = \frac{1}{L_a}(-i_3R + v_{cn} - e_3) \qquad (3)$$

R_1, R_2, and R_3 are the sum of the motor phase resistance and the resistance of the particular SCR between the source and phase of the motor at the particular instant of time under analysis. The form of phase voltages (v_{an}, v_{bn}, v_{cn}) will be discussed later.

Neutral connection current is given by Kirchhoff's current law as

$$i_{nN} = i_1 + i_2 + i_3 \qquad (4)$$

As long as motor speed (ω_m) remains non zero, the electromechanical developed torque is given by

$$T_d = (e_1i_1 + e_2i_2 + e_3i_3)/\omega_m \qquad (5)$$

Equations (1) - (5) completely describe the instantaneous performance of the brushless DC motor.

4.0 CURRENT HARMONICS TO BE ELIMINATED

For both the case of a DC link inverter drive and the case of a synchronous envelope cycloconverter drive, the motor phase currents approach the pulse width controlled square wave of Figure 3. A Fourier series representation of the wave form shows that all odd, nontriplen harmonics exist in the phase current.

$$i_1 = \frac{4I}{\pi} \sum_n \frac{1}{n} \cos\left(\frac{n\pi}{6}\right) \sin n\omega t, \qquad n = 1,5,7,11,13,\cdots \tag{6}$$

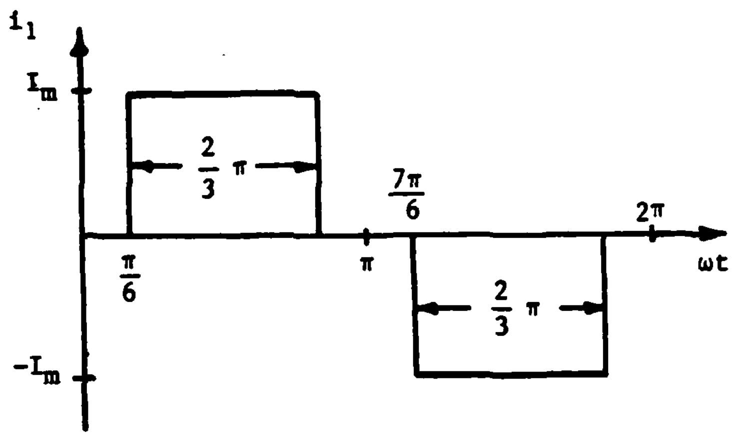


Figure 3. Approximate Form of Motor Phase Current

Similarly, the remaining balanced phase currents are given by

$$i_2 = \frac{4I}{\pi} \sum_n \frac{1}{n} \cos\left(\frac{n\pi}{6}\right) \sin n\left(\omega t - \frac{2\pi}{3}\right) \tag{7}$$

$$i_3 = \frac{4I}{\pi} \sum_n \frac{1}{n} \cos\left(\frac{n\pi}{6}\right) \sin n\left(\omega t + \frac{2\pi}{3}\right) \tag{8}$$

$$n = 1,5,7,11,13,\cdots$$

Since the motor counter emfs (e_1, e_2, e_3) are sinusoids of fundamental frequency and form a balanced three-phase set, each can be expressed as

$$e_1 = E_m \sin(\omega t + \phi) \tag{9}$$

$$e_2 = E_m \sin(\omega t + \phi - \frac{2\pi}{3}) \tag{10}$$

$$e_3 = E_m \sin(\omega t + \phi + \frac{2\pi}{3}) \tag{11}$$

If equations (6) - (11) are substituted into (5), the simplified result for instantaneous developed motor torque is found.

$$T_d = \frac{4}{\pi} E_m I \sum_n \frac{1}{n} \cos(\frac{n\pi}{6}) \{\cos[(n-1)\frac{\pi}{6}\omega t - \phi] - \cos[(n+1)\frac{\pi}{6}\omega t + \phi]\} \tag{12}$$

$$n = 1,5,7,11,13,\cdots$$

From (12), it is seen that the instantaneous developed motor torque is made up of a constant term plus all multiples of the sixth harmonic of motor phase current frequency. Further, the multiples of sixth harmonic components of torque are a direct result of the current harmonics that lie immediately on either side of multiples of six. Specifically, the sixth harmonic of torque results from the fifth and seventh harmonic of current; and, the twelfth harmonic of torque results from the eleventh and thirteenth harmonic of current; etc. Clearly, the motor developed torque will be constant if the harmonics of motor phase current described by $6n \pm 1$, $n = 1,2,3,\cdots$ are eliminated.

The above discussed harmonics only need to be eliminated if their existence results in torque harmonics in the range of response of a coupled mechanical load. Since typical mechanical loads exhibit negligible response to frequencies above 20 hertz, it should only be necessary to eliminate those nontriplen, odd current harmonics beyond the first for which

$$M \leq \frac{80\pi}{6 p\omega_m} \tag{13}$$

where p = number of poles and ω_m is the motor speed in rad/s.

5.0 MODULATION OF PHASE VOLTAGE

The cycloconverter system is voltage excited; thus, if the $(n \pm 1)$ harmonics are to be eliminated from motor phase currents, they must be eliminated from the phase voltage set.

5.1 Development of Modulation Function

Modulation function $h(\omega t)$ that contains no selected harmonics up through M is derived in Appendix B. The balanced three-phase voltage set to be applied for equations (1) - (3) is given by

$$v_{an} = v_d \, h(\omega t) \tag{14}$$

$$v_{bn} = v_d \, h(\omega t - \frac{2\pi}{3}) \tag{15}$$

$$v_{cn} = v_d \, h(\omega t + \frac{2\pi}{3}) \tag{16}$$

If ω_L is the radian frequency of the three-phase AC source, then v_d is a periodic wave form described by

$$v_d(t) = V_m \sin(\omega_L t + \frac{\pi}{6} + \alpha_s) , \quad 0 \leq \omega_L t \leq \frac{2\pi}{3} \tag{17}$$

and, $$v_d(t + T) = v_d(t) \tag{18}$$

where $$T = \frac{2\pi}{3\omega_L}$$

It is shown in Appendix A that if the modulated phase voltages (v_{an}, v_{bn}, v_{cn}) are to contain none of the selected harmonics through M, it is necessary that the frequency of v_d be much greater than the frequency of h(t); however, this condition is easily satisfied at near zero speed.

Figure 4 illustrates the unmodulated phase voltage. The modulated phase voltage is shown by Figure 5.

5.2 Minimization of Slack Variable

Development of the modulation function for use with the synchronous envelope cycloconverter in Appendix A required an extension of the well-known pulse width modulation techniques for elimination of harmonics in inverter drives [14] to include a slack variable. The modulation function contained no selected harmonics through M as determined by (13). For the reported work, the slack variable is always chosen so as to minimize the sum of the squares of the amplitude of the next higher (6n ± 1) current harmonic pair beyond harmonics selectively eliminated. Listings of the FORTRAN programs used to compute the modulation function are presented in Appendix B.

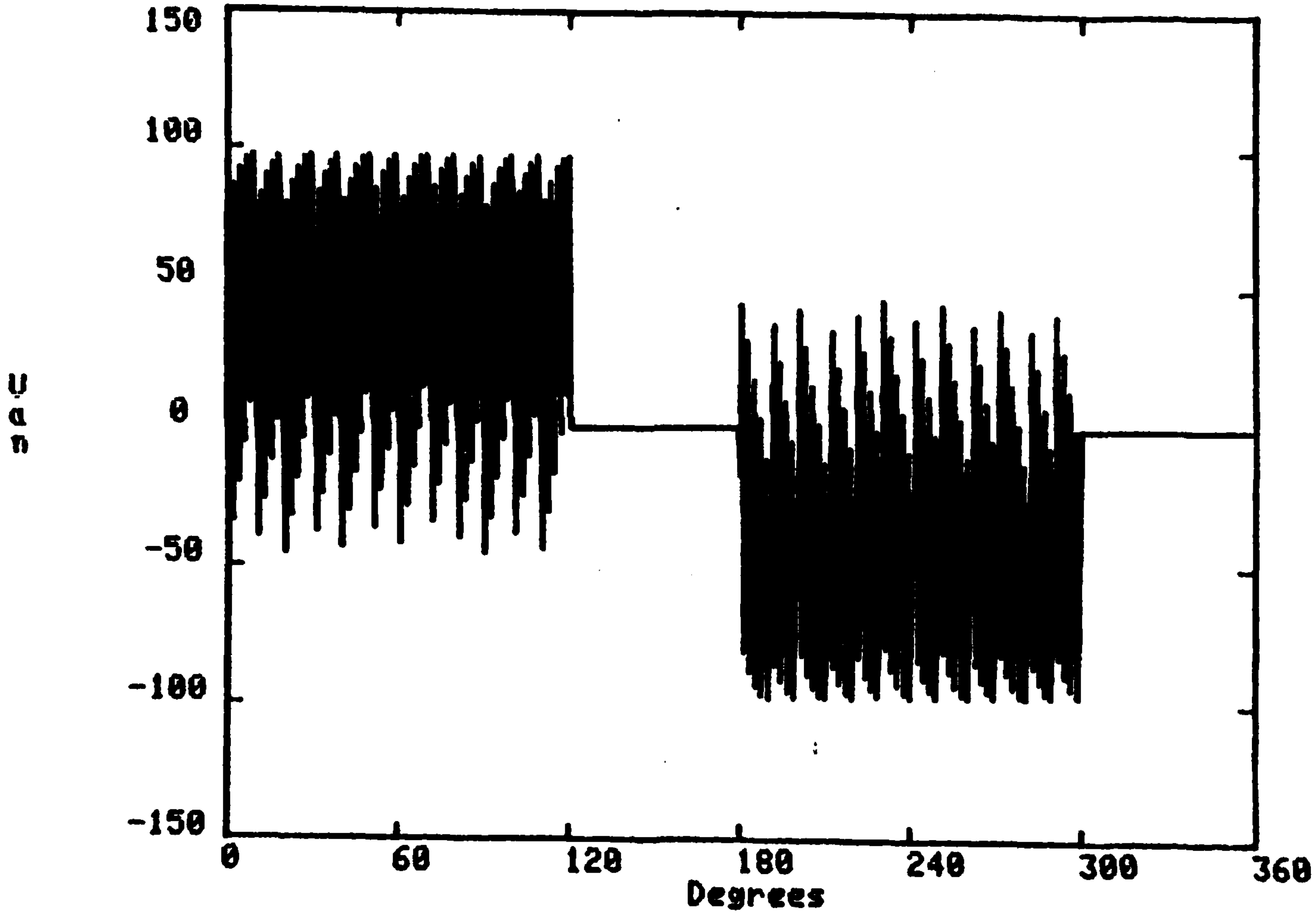


Figure 4. Unmodulated Phase Voltage

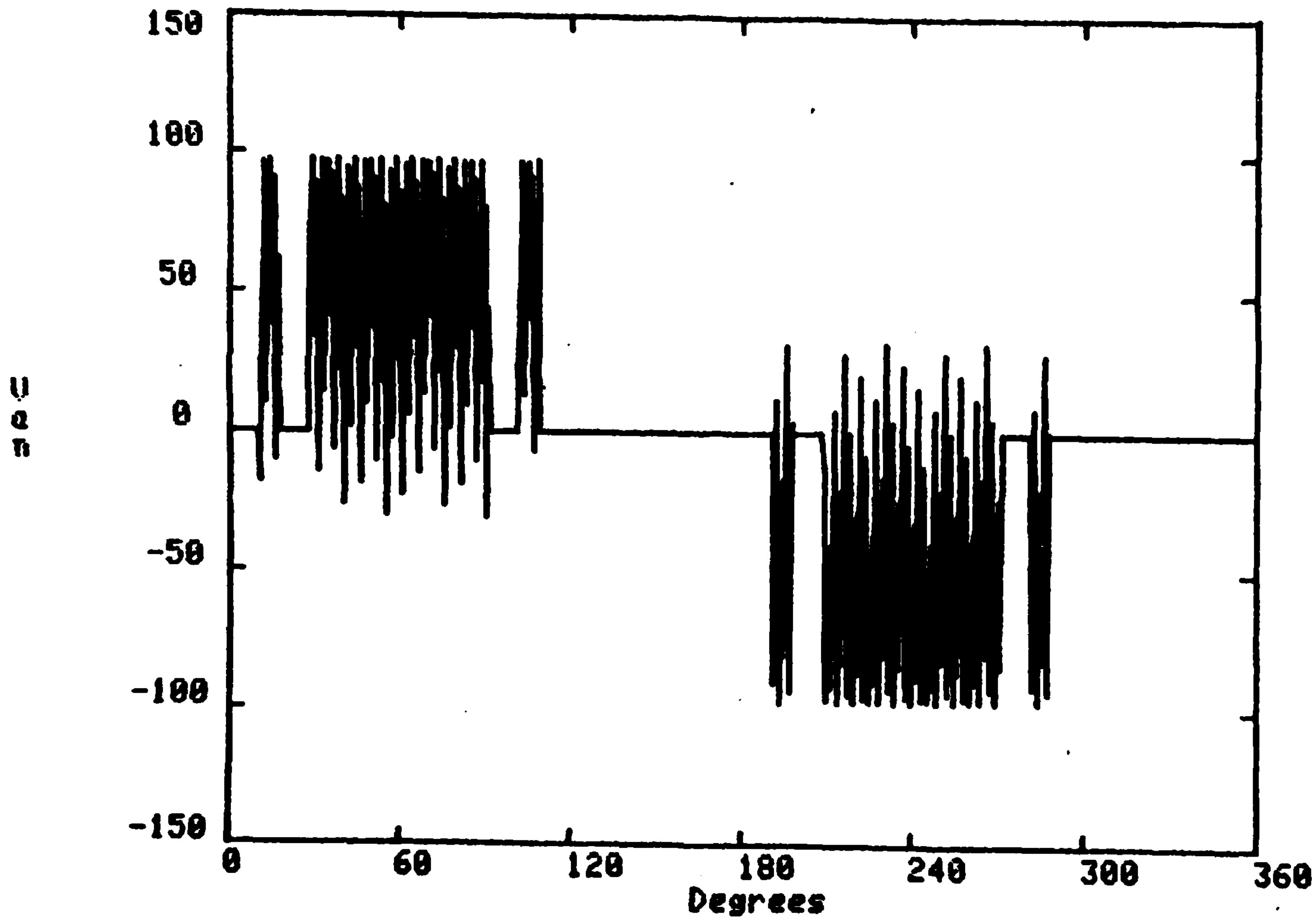


Figure 5. Modulated Phase Voltage

5.3 Cycloconverter Connection Method

Since the modulation function $h(\omega t)$ does not repeat itself every $\pi/6$ radians on the interval $[\pi/6, 5\pi/6]$, attention must be directed to the cycloconverter connection scheme utilized so that the eliminated harmonics are not inadvertently reintroduced. Specifically, a conduction path must be provided so that all harmonics remaining in the phase voltage can flow if the phase current is to be of the same harmonic content. Two connection realizations are possible wherein no restriction on phase current harmonics are imposed by Kirchhoff's current law. The first of these acceptable connections is that of the midpoint cycloconverter with the neutral lead in place. A second realization would be the full-bridge cycloconverter with total phase isolation. Since this latter connection requires 36 SCRs, it was not used in the reported study due to a factor of three on component requirement over the midpoint cycloconverter case. However, it should be pointed out that the full-bridge cycloconverter with its six-pulse output has a smaller ripple magnitude and may offer a slight efficiency advantage.

5.4 Fourier Spectrum Analysis

Nature of the waveforms for current and instantaneous developed torque are such that visual inspection to determine harmonic content is not practical or accurate. The listing of a FORTRAN program to find the Fourier coefficients is given in Appendix D.

This program was used to generate the data for the normalized Fourier spectrum plots of Section III for harmonics up through forty-eight in order to verify that attempt to selectively eliminate harmonics was successful.

This page left blank

SECTION III

RESULTS

1.0 SUMMARY

Three speed conditions were selected for numerical study to evaluate the effectiveness of the harmonic elimination technique in removing undesired torque pulsations from the midpoint cycloconverter driven brushless DC motor system when using synchronous envelope control:

(a). 4500 rpm (10% speed case)

(b). 2250 rpm (5% speed case)

(c). 450 rpm (1% speed case)

Although these speeds are great enough that elimination of harmonics as determined by equation (13) would not be necessary, there are two reasons for their selection. The first is conservation of computer time. Computation of a set of data for the 450 rpm (1% speed case) required approximately 8 hours of CPU time on a Control Data PDP 11/44 computer. Since many trial-and-error runs were necessary for each data set, runs at lower motor speed become prohibitive timewise. Secondly, if the harmonics can be successfully eliminated at higher speeds, then there is not difficulty to be encountered at reduced speed. Justification lies in the fact that the undesired harmonics are totally eliminated from the phase voltage set. Any re-introduction of the undesired harmonics into the phase current wave forms would be attributable to commutation overlap which is dependent on the L_a/R time constant that determines current decay at the end of each phase current conduction sequence. Since this time constant is not speed dependent, the commutation overlap interval is a larger portion of the current wave form period at high speed than it is at low speed. Thus, the departure of phase current from the form of the phase voltage is greater at high speed than at low speed. It is concluded that if undesireable harmonics are not re-introduced at high speed, then success at low speed is guaranteed.

For each speed case, the nature of current and torque were examined for application of a balanced set of phase voltages of the following nature:

(a). No harmonic eliminated.

(b). Fifth and seventh harmonics eliminated.

(c). Fifth, seventh, eleventh, and thirteenth harmonics eliminated.

A brushless DC machine system has an extra input or degree of freedom over either a conventional DC machine or a synchronous machine drive system in that the angular displacement between the stator and rotor magnetic fields can be specifically controlled. For this study, this extra degree of control freedom is implemented by specification of the angle (α) measured from the onset of positive phase current conduction to the positive going zero crossing of the associated motor phase counter emf. A convenient value ($\alpha = 30^{o}$) is used for most of the study; however, since selection of α shifts the position of motor phase counter emf with respect to phase current, the nature of commutation overlap can be altered by the value of α. The effect of changing α over the range from 15^{o} to 40^{o} is studied for one speed case.

2.0 PERFORMANCE STUDIES

Equations (1) - (3) can be written in compact matrix form as

$$\frac{d}{dt}\underline{i} = \frac{1}{L_a}[R]\,\underline{i} + \frac{1}{L_a}\underline{v_\phi} - \frac{1}{L_a}\underline{e} \qquad (20)$$

where:

$$\underline{i} = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} \quad (21) \; ; \quad \underline{e} = \frac{2\omega K}{p}\begin{bmatrix} \sin(\omega t-\alpha) \\ \sin(\omega t-\alpha-\frac{2\pi}{3}) \\ \sin(\omega t-\alpha+\frac{2\pi}{3}) \end{bmatrix} \quad (22)$$

$$[R] = \begin{bmatrix} R_1 & & \\ & R_2 & \\ & & R_3 \end{bmatrix} \quad (23) \; ; \quad \underline{v_\phi} = v_d(t)\begin{bmatrix} h(\omega t) \\ h(\omega t-\frac{2\pi}{3}) \\ h(\omega t+\frac{2\pi}{3}) \end{bmatrix} \quad (24)$$

Equation (20) is a nonlinear set of differential equations due to the elements of [R] being functions of current. Any attempt at linearization for a wide range of $\underline{i}$ would unacceptably alter the physical nature of the problem by removing the rectifying characteristic of the SCRs. Consequently, a closed form solution of (20) is not possible and a numerical integration must be implemented. A fixed increment (1.0×10^{-6}s) fourth-order Runge-Kutta numerical integration method is utilized for this work.

After forming a modulation function through use of the programs of Appendix B, selecting a particular motor speed ($\omega_m = 2\omega/p$), and arbitrarily choosing a value of SCR firing angle α_g(see equation (17)), numerical solution to (20) is implemented subject to initial conditions $\underline{i}(0) = \underline{0}$. Integration is continued until steady-state is reached. Instantaneous values of developed torque are calculated according to (5) and numerically averaged over the last steady-state cycle of integration to give the average value of developed torque. This value of average torque is compared with a specified value for the operating point. If the average value calculated is different from the specified value, then α_s is appropriately adjusted and the numerical integration is repeated until the difference magnitude between calculated and specified torque are sufficiently close in an epsilon sense. For all cases studied, the specified torque was taken as 1.5 N·m.

The listing of a FORTRAN program to compute the performance as described above is presented in Appendix D. In addition to the discussed computations, the program also calculates the RMS value of phase current over the last steady-state cycle as $(\frac{1}{T} \sum i^2 \Delta t)^{\frac{1}{2}}$. Also, values of phase current and instantaneous torque are stored over the last steady-state cycle for use by the Fourier Spectrum Program of Appendix C.

Results of the various speed cases and harmonic elimination studies are summarized in the paragraphs that follow.

2.1 4500 RPM (10% Speed Case)

2.1.1 Unmodulated Phase Voltage.

Input Data

Motor speed, n_m = 4500 rpm
Motor frequency, f = 150 Hz
SCR firing angle, α_s = 58.68°
Specified average torque, T_{dav} = 1.5 N·m

Output Data

Average torque, T_{dav} = 1.501 N·m
RMS phase current, I_{RMS} = 72.77 A

A plot of phase current i_1 is given by Figure 6. The associated Fourier spectrum of i_1 is shown by Figure 7. Predominantly triplen harmonic current that flows through the neutral connection is depicted by Figure 8.

A plot of the instantaneous developed torque where the sixth harmonic is obvious is given by Figure 9. A Fourier spectrum of the instantaneous torque is shown in Figure 10 where existence of all multiples of six as theoretically predicted are noted.

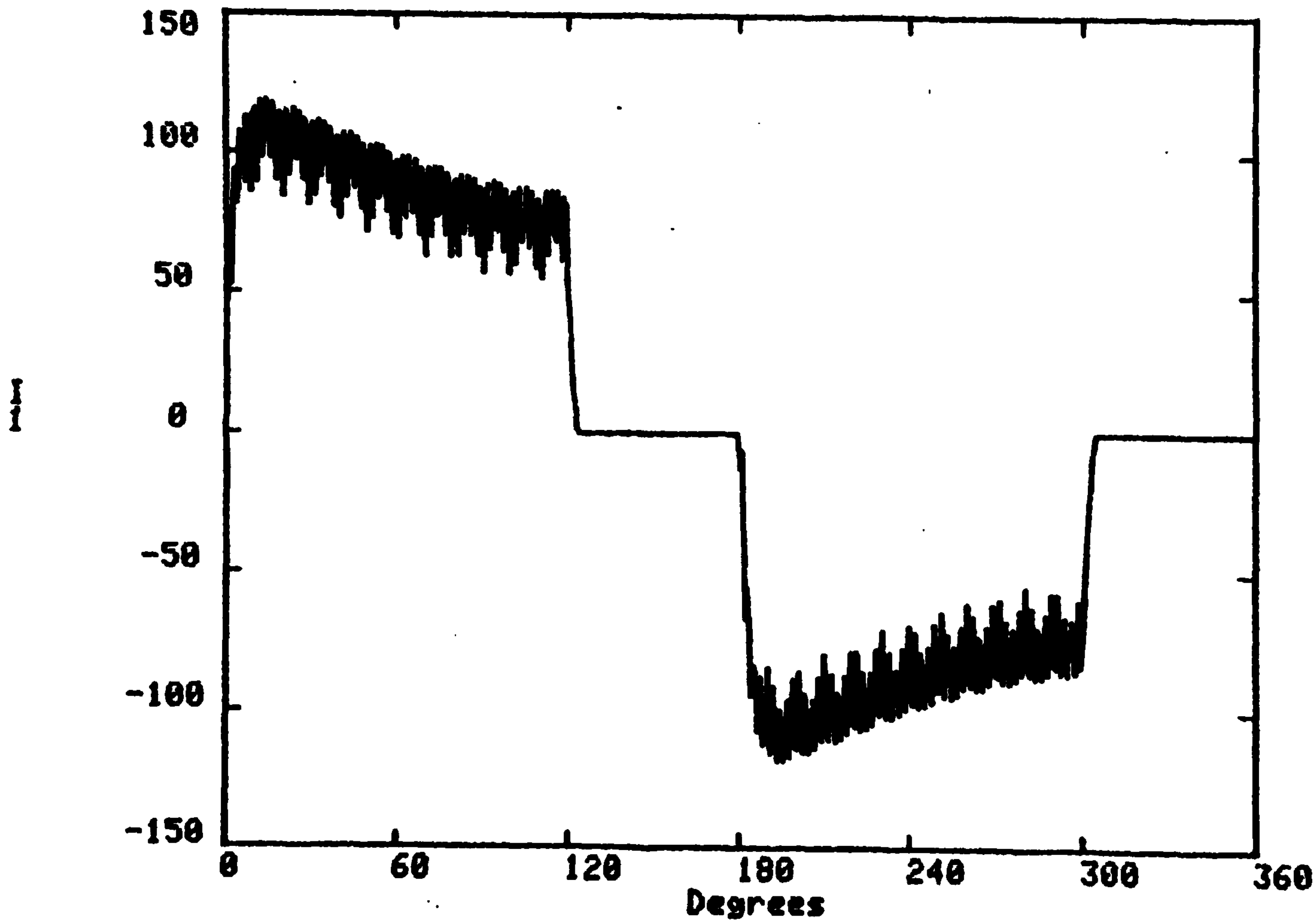


Figure 6. Phase Current - 4500 RPM, No Modulation

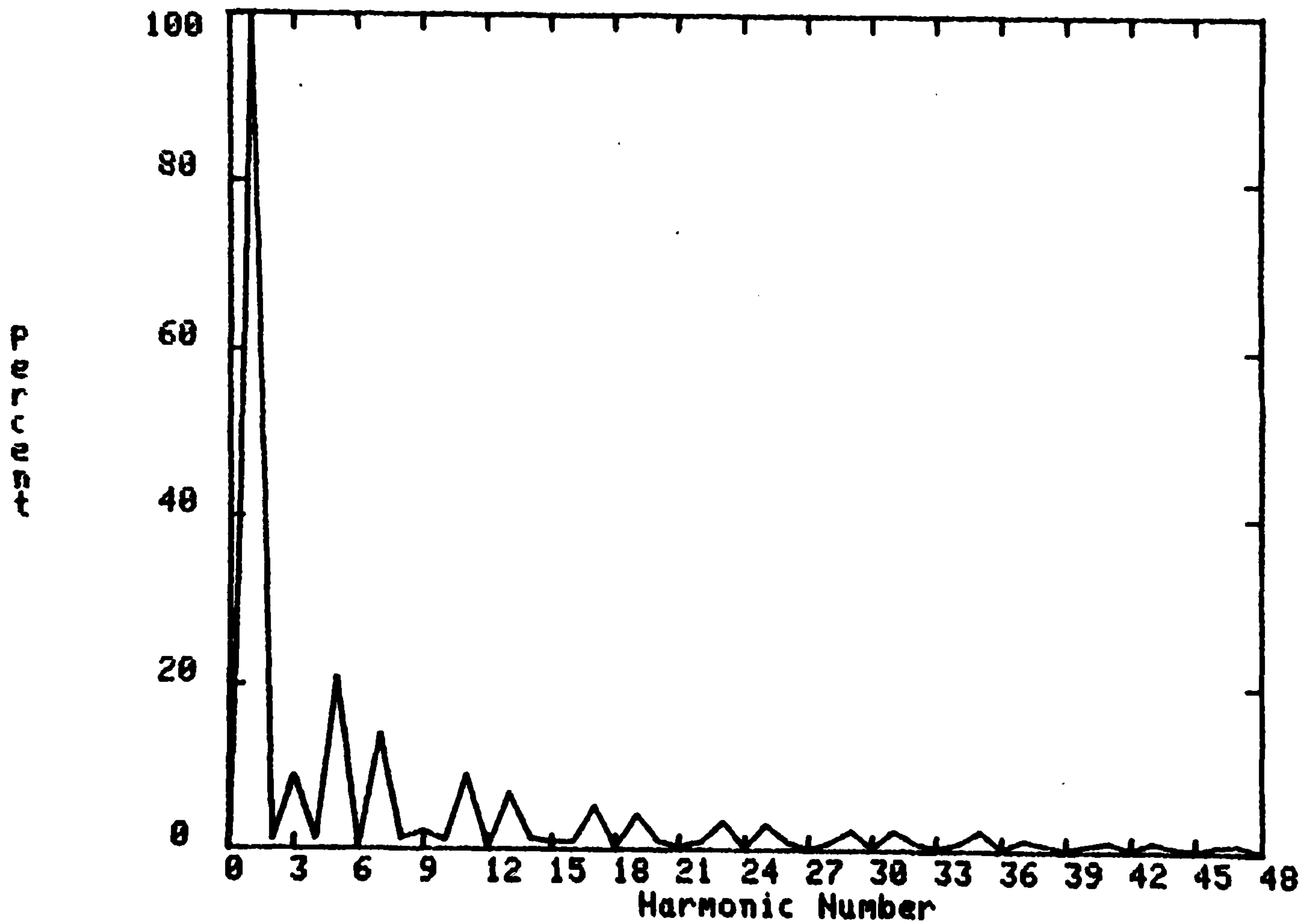


Figure 7. Fourier Spectrum of Phase Current - 4500 RPM, No Modulation

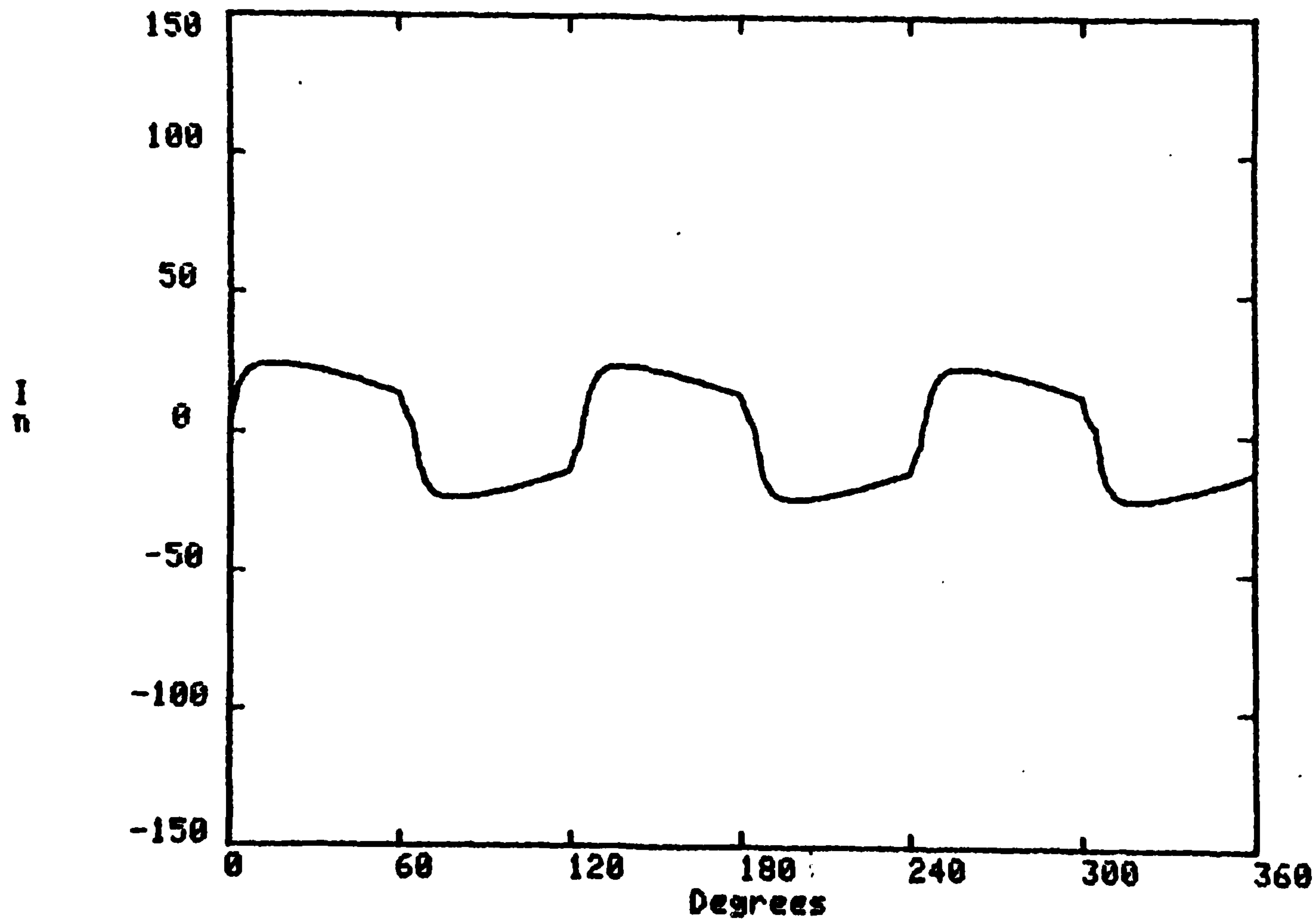


Figure 8. Neutral Current - 4500 RPM, No Modulation

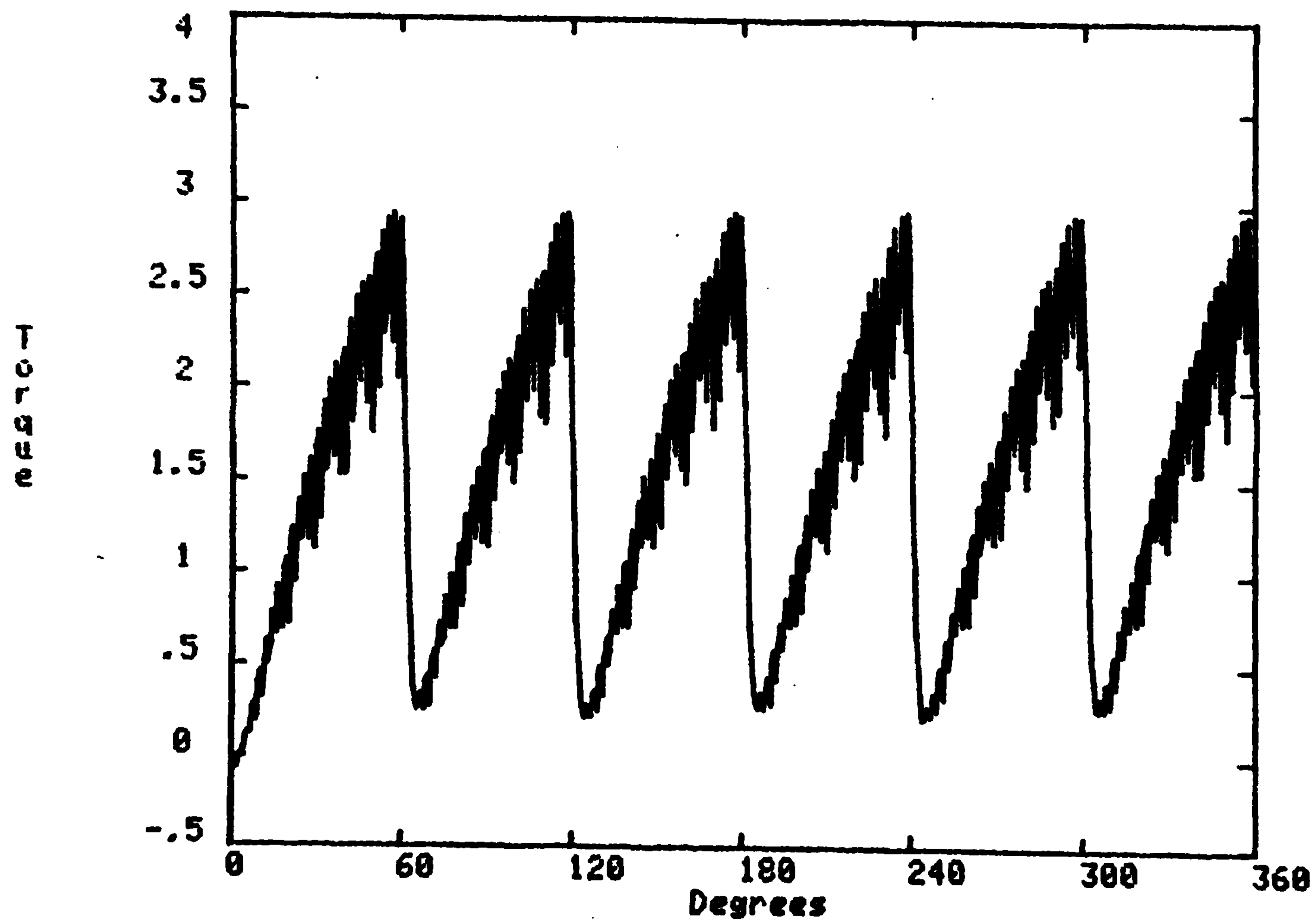


Figure 9. Developed Torque - 4500 RPM, No Modulation

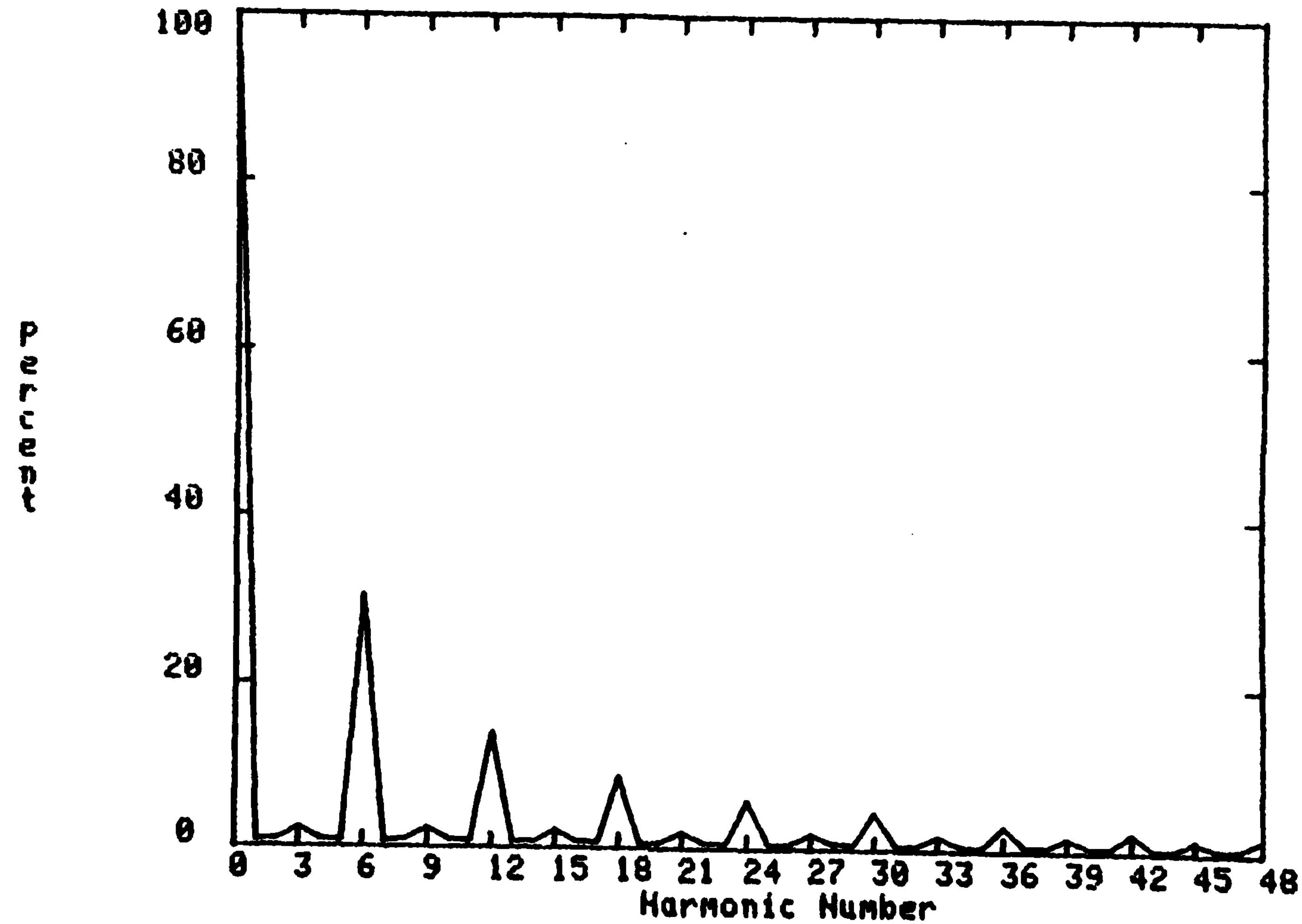


Figure 10. Fourier Spectrum of Developed Torque - 4500 RPM, No Modulation

2.1.2 Elimination of Sixth Harmonic of Torque. The sixth harmonic of torque is eliminated by removing the fifth and seventh harmonics of phase current.

Input Data

Motor speed, n_m = 4500 rpm
Motor frequency, f = 150 Hz
SCR firing angle, α_s = 46.45°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 40.76°
α_2 = 47.73°
α_3 = 58.65°

Output Data

Average torque, T_{dav} = 1.5001 N·m
RMS phase current, I_{RMS} = 74.24 A

A plot of phase current i_1 is given by Figure 11. The associated Fourier spectrum of i_1 is shown by Figure 12, where negligibly small fifth and seventh harmonics are noted. Current flow through the neutral line is depicted by Figure 13.

A plot of the instantaneous developed torque is given by Figure 14. The Fourier spectrum of the developed torque is shown in Figure 15 where it is seen that the sixth harmonic is near zero.

2.1.3 Elimination of Sixth and Twelfth Harmonics of Torque. Both the sixth and twelfth harmonics of torque are eliminated by removing the fifth, seventh, eleventh, and thirteenth harmonics of phase current.

Input Data

Motor speed, n_m = 4500 rpm
Motor frequency, f = 150 Hz
SCR firing angle, α_s = 42.63°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 38.73°
α_2 = 42.13°
α_3 = 57.25°
α_4 = 61.93°
α_5 = 66.86°

Output Data

Average torque, T_{dav} = 1.505 N·m
RMS phase current, I_{RMS} = 74.59 A

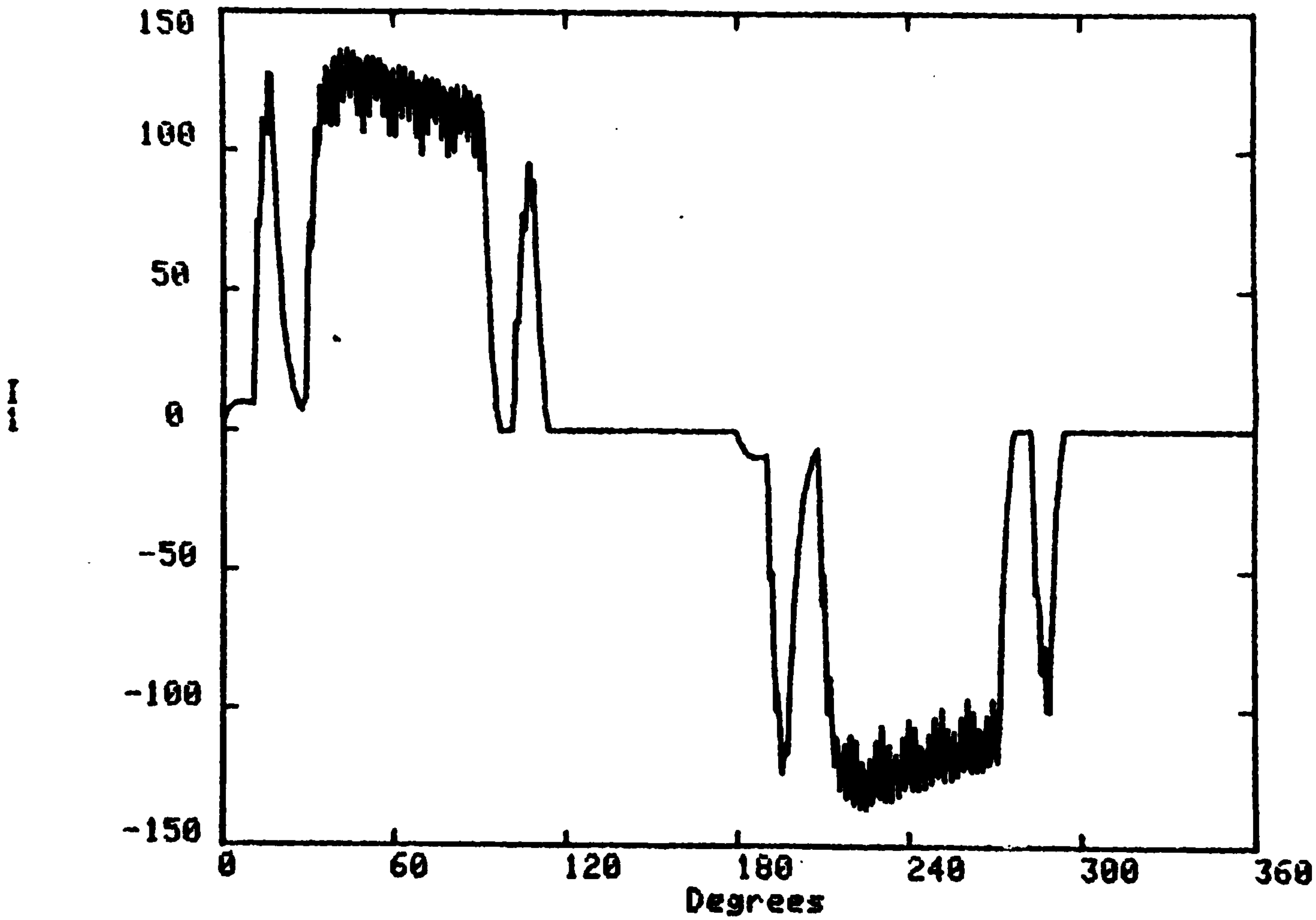


Figure 11. Phase Current - 4500 RPM, Fifth and Seventh Harmonic Eliminated

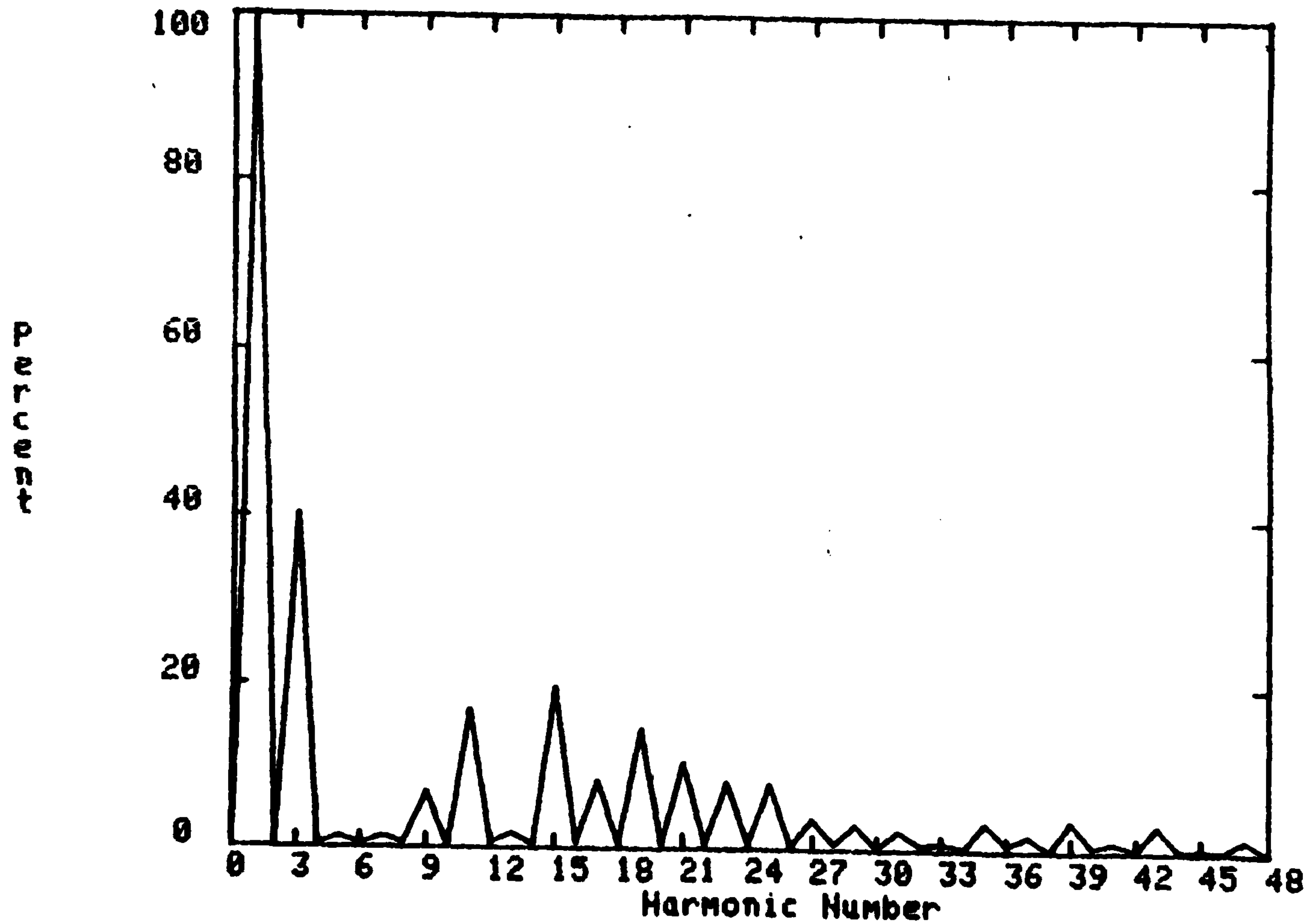


Figure 12. Fourier Spectrum of Phase Current - 4500 RPM, Fifth and Seventh Harmonics Eliminated

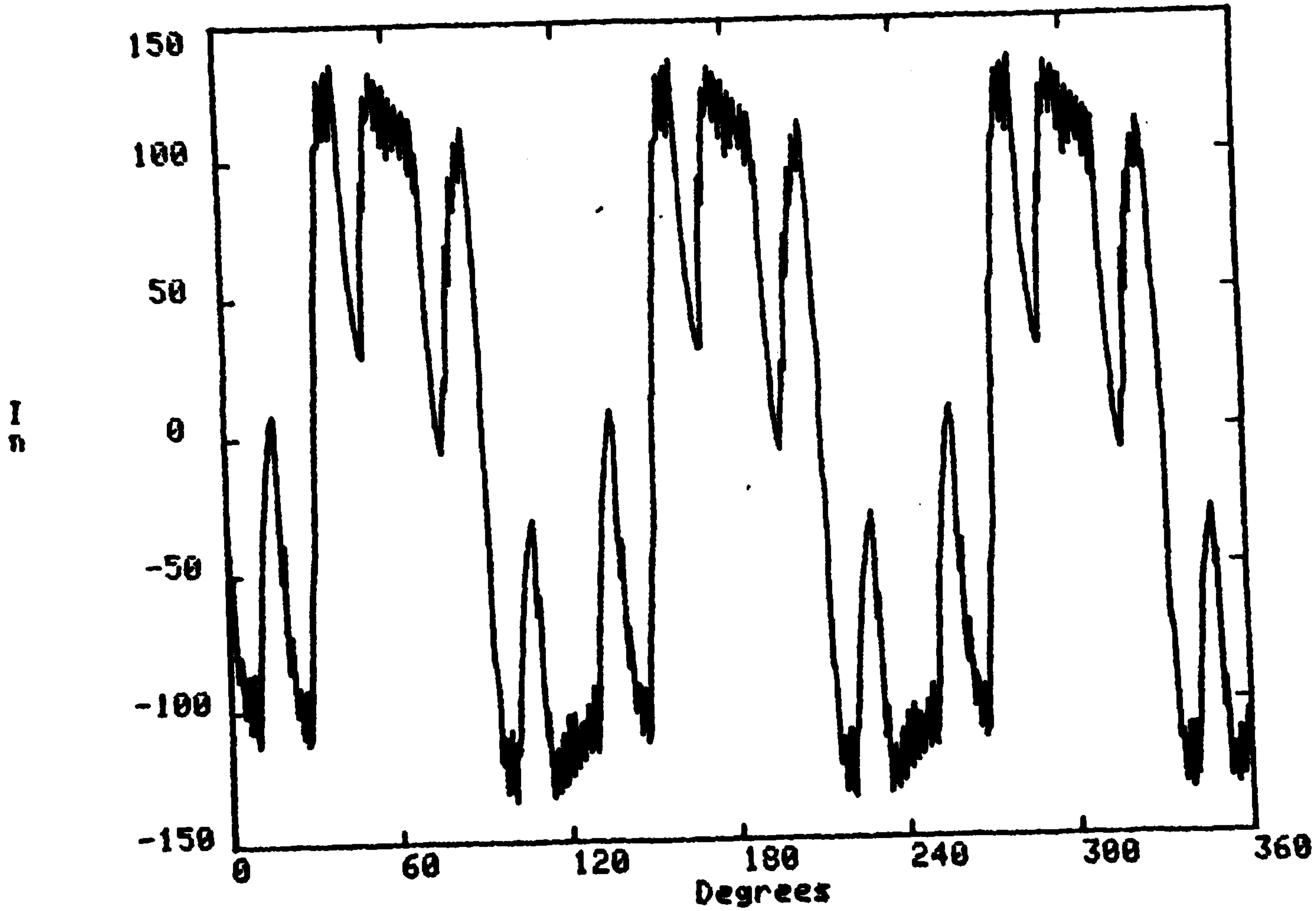


Figure 13. Neutral Current - 4500 RPM, Fifth and Seventh Harmonics Eliminated

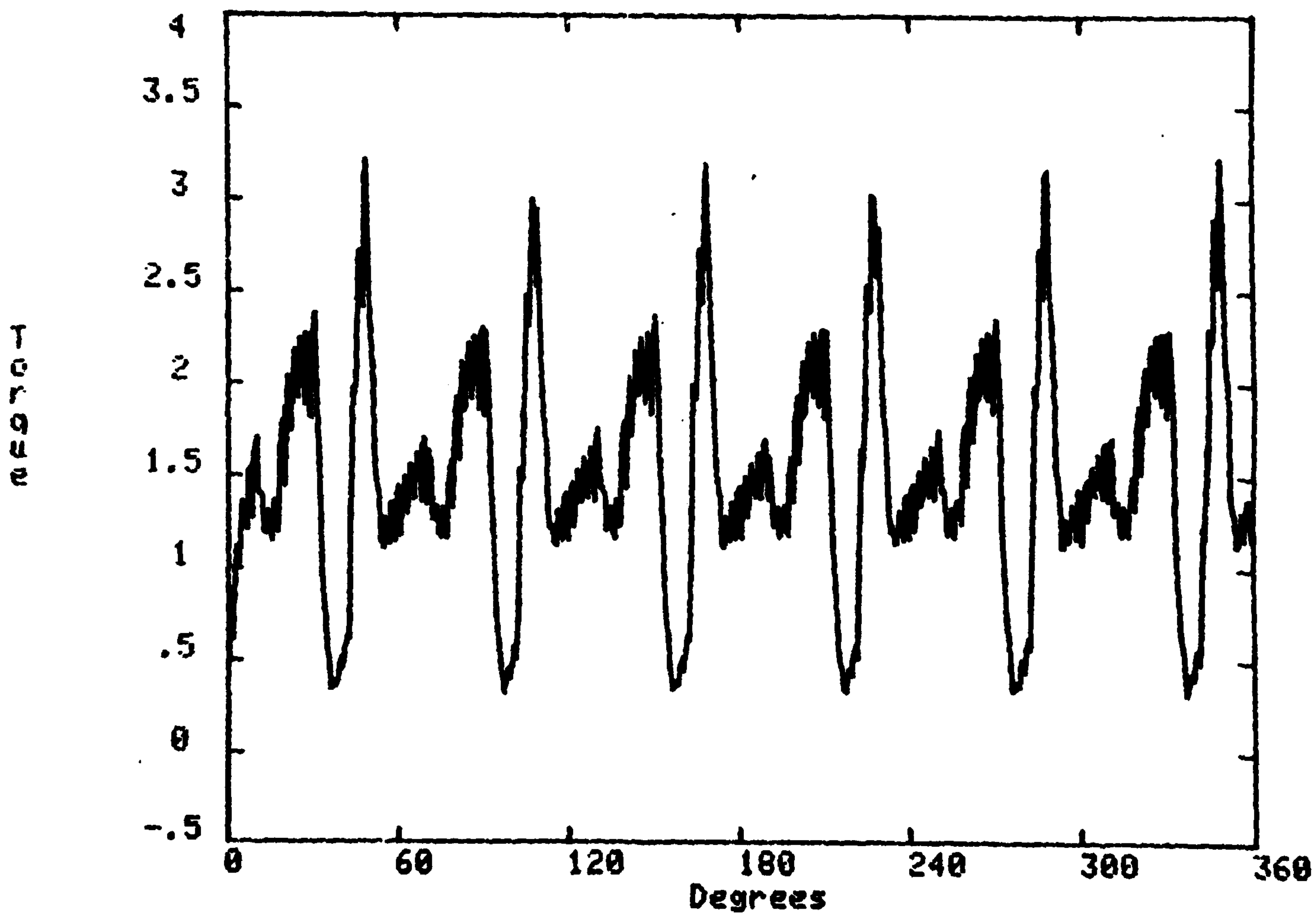


Figure 14. Developed Torque - 4500 RPM, Sixth Harmonic Eliminated

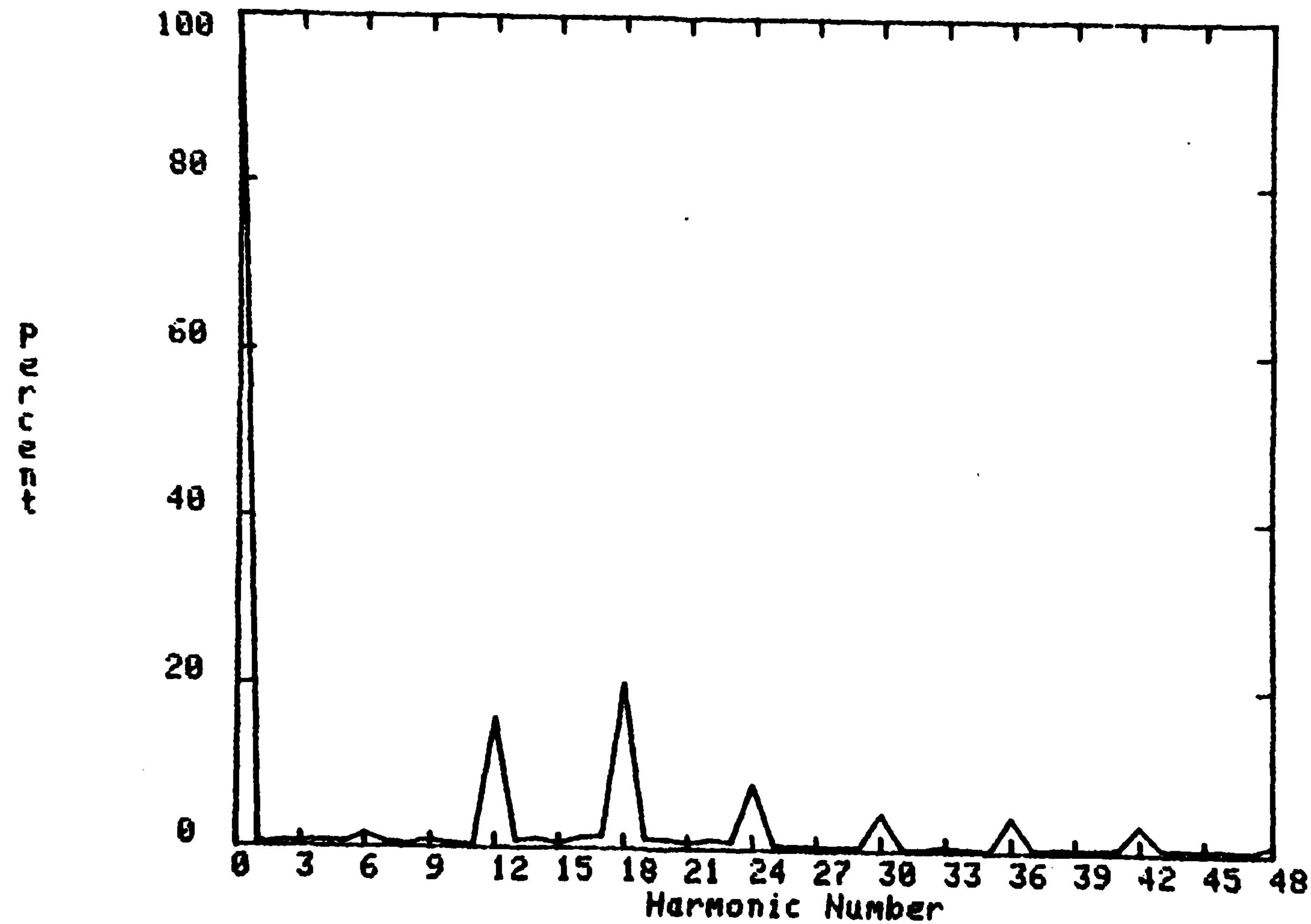


Figure 15. Fourier Spectrum of Developed Torque - 4500 RPM, Sixth Harmonic Eliminated

A plot of phase current i_1 is given by Figure 16. The associated Fourier spectrum of i_1 is shown by Figure 17 where negligibly small fifth, seventh, eleventh, and thirteenth harmonics are noted. Current flow through the neutral line is depicted by Figure 18.

A plot of the instantaneous developed torque is given by Figure 19. The Fourier spectrum of the developed torque is shown in Figure 20 where it is seen that the sixth and twelfth harmonics are near zero.

2.2 2250 RPM (5% Speed Case)

2.2.1 Unmodulated Phase Voltage.

Input Data

Motor speed, n_m = 2250 rpm
Motor frequency, f = 75 Hz
SCR firing angle, $_s$ = 61.93°
Specified average torque, T_{dav} = 1.5 N·m

Output Data

Average torque, T_{dav} = 1.499 N·m
RMS phase current, I_{RMS} = 69.53 A

A plot of phase current i_1 is given by Figure 21. The associated Fourier spectrum of i_1 is shown by Figure 22.

A plot of the instantaneous developed torque is given by Figure 23. A Fourier spectrum of the developed torque is shown by Figure 24.

2.2.2 Elimination of Sixth Harmonic of Torque. The sixth harmonic of torque is eliminated by removing the fifth and seventh harmonics of phase current.

Input Data

Motor speed, n_m = 2250 rpm
Motor frequency, f = 75 Hz
SCR firing angle, α_s = 50.21°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 40.76°
α_2 = 47.73°
α_3 = 58.65°

Output Data

Average torque, T_{dav} = 1.5001 N·m
RMS phase current, I_{RMS} = 73.96 A

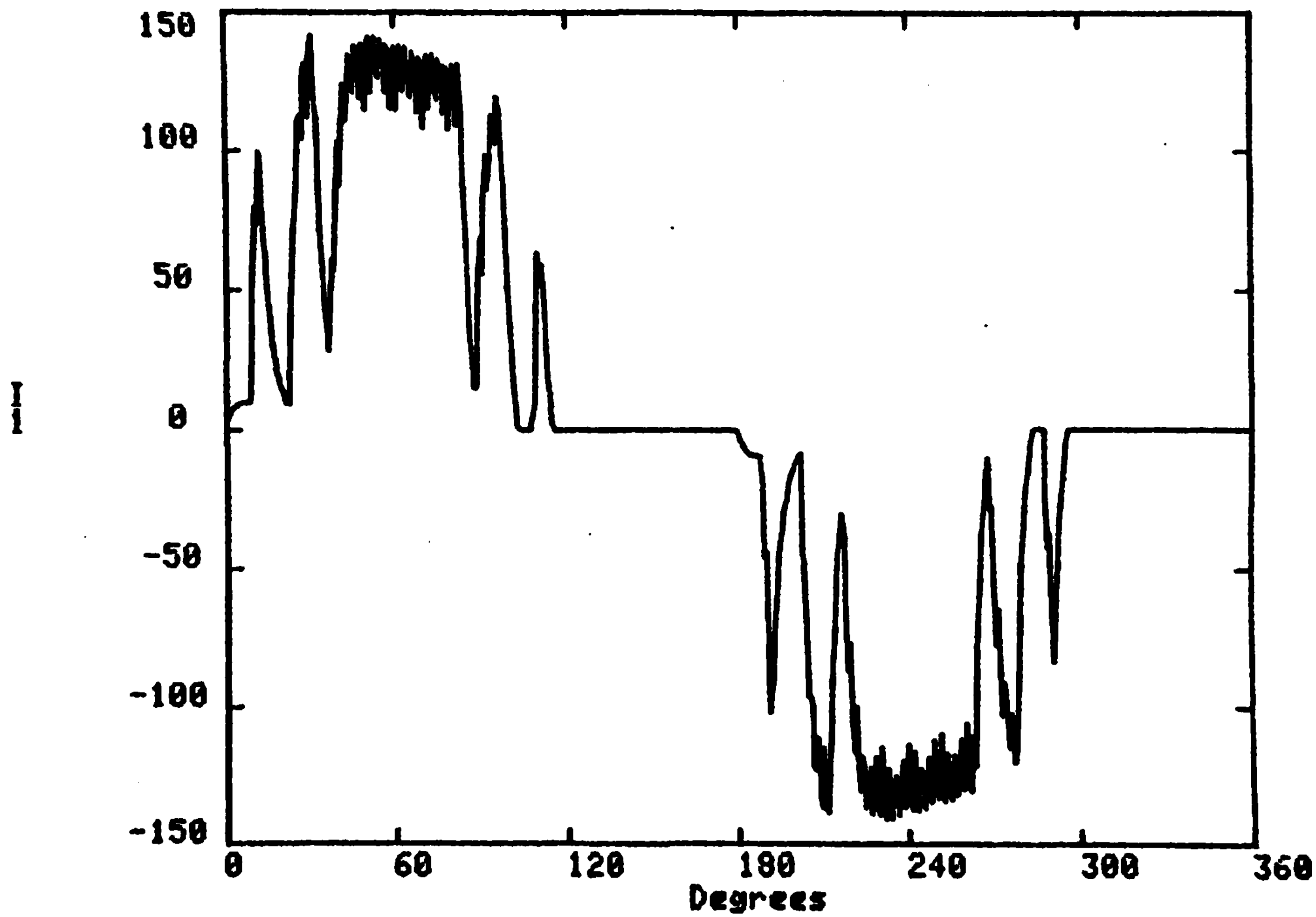


Figure 16. Phase Current - 4500 RPM, Fifth, Seventh, Eleventh and Thirteenth Harmonics Eliminated

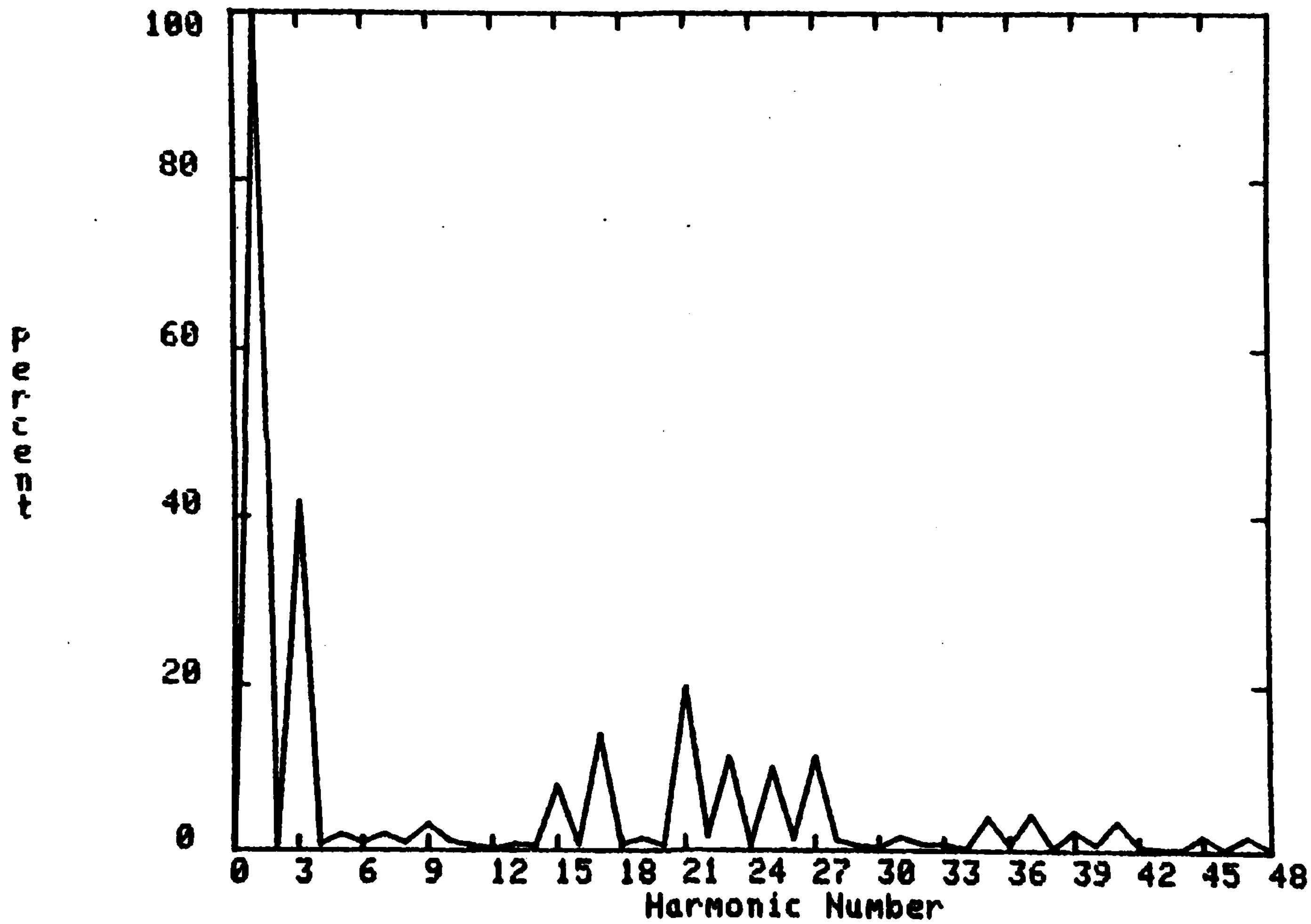


Figure 17. Fourier Spectrum of Phase Current - 4500 RPM, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

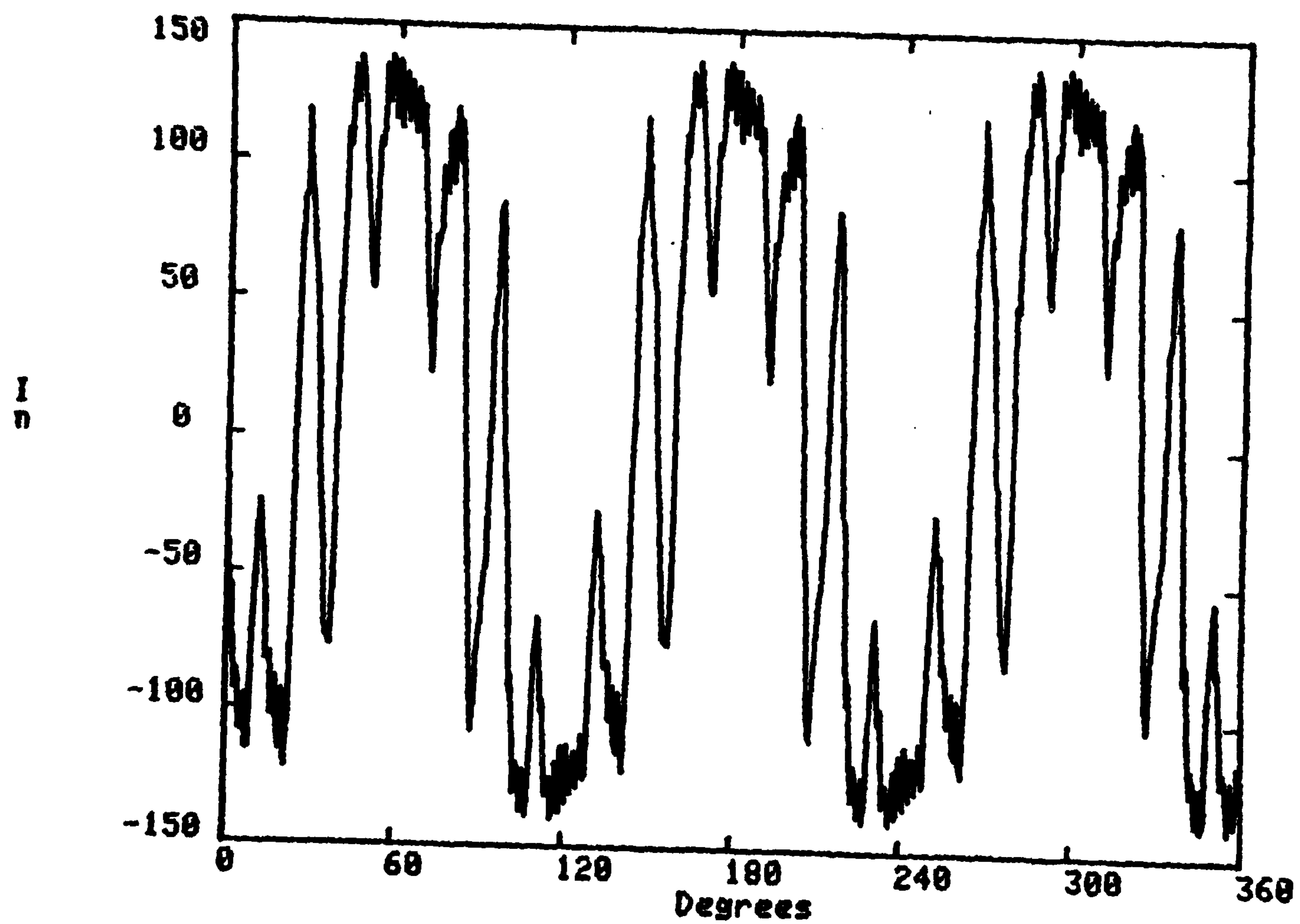


Figure 18. Neutral Current - 4500 RPM, Fifth, Seventh, Eleventh and Thirteenth Harmonics Eliminated

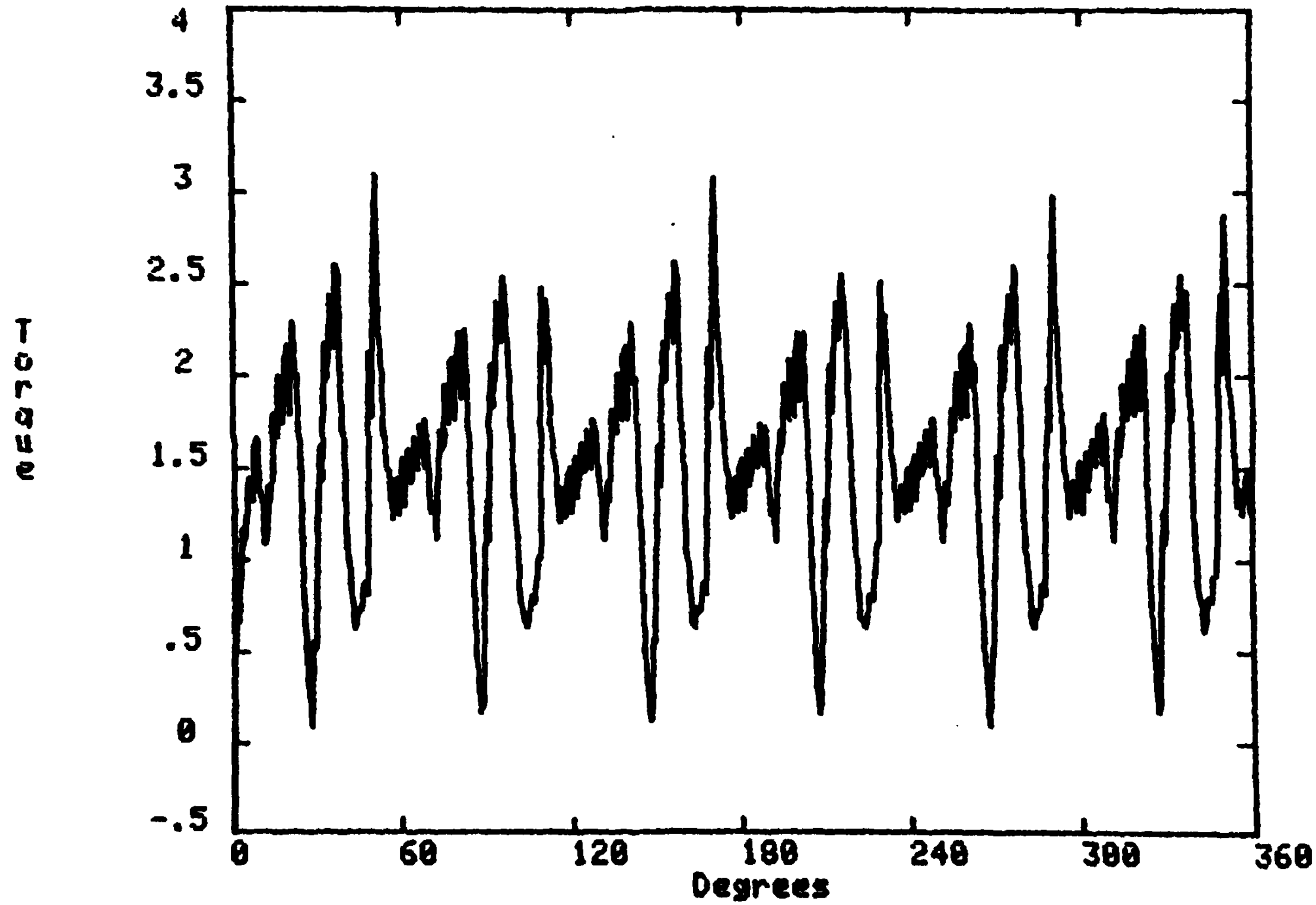


Figure 19. Developed Torque - 4500 RPM, Sixth and Twelfth Harmonics Eliminated

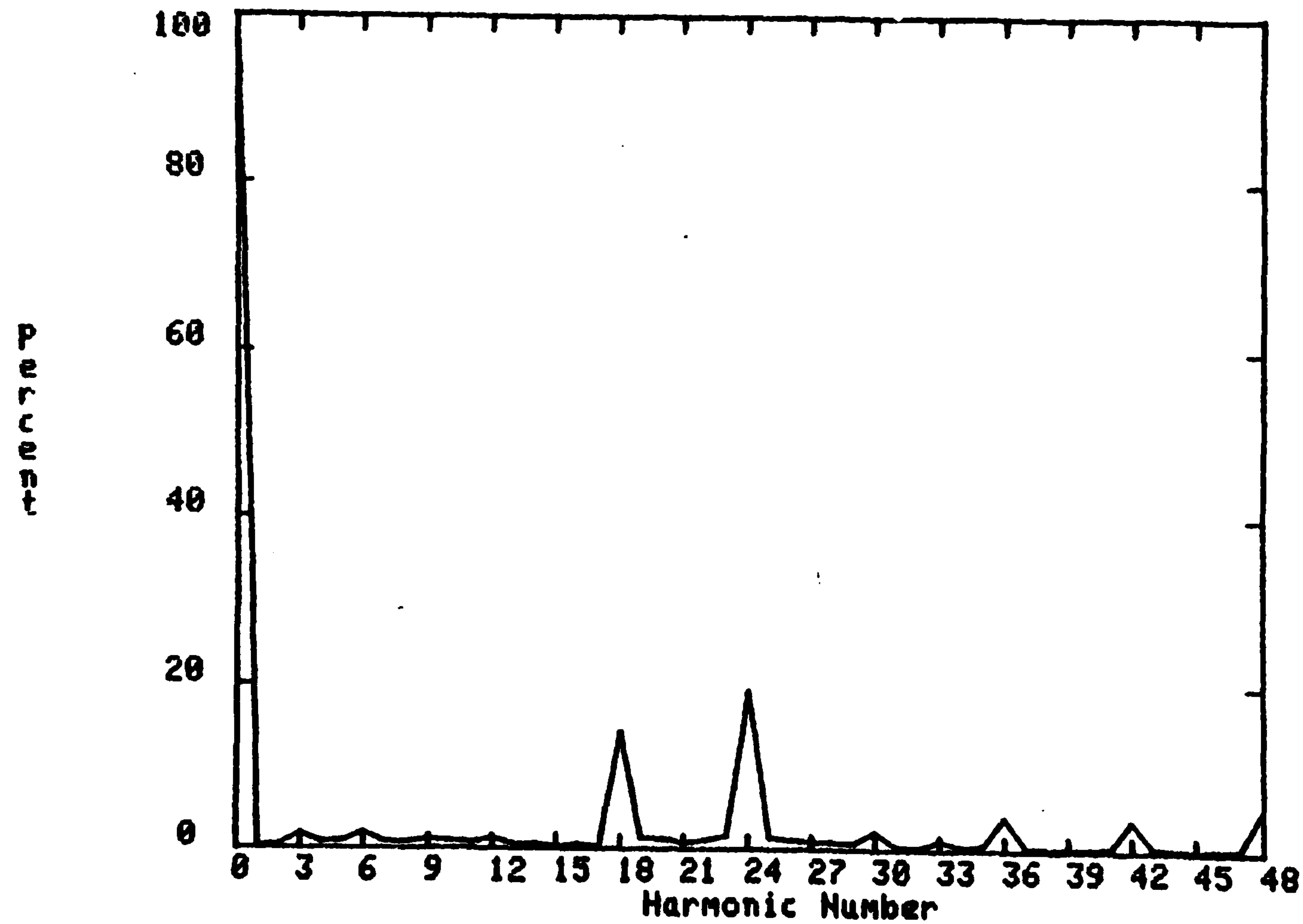


Figure 20. Fourier Spectrum of Developed Torque - 4500 RPM, Sixth and Twelfth Harmonics Eliminated

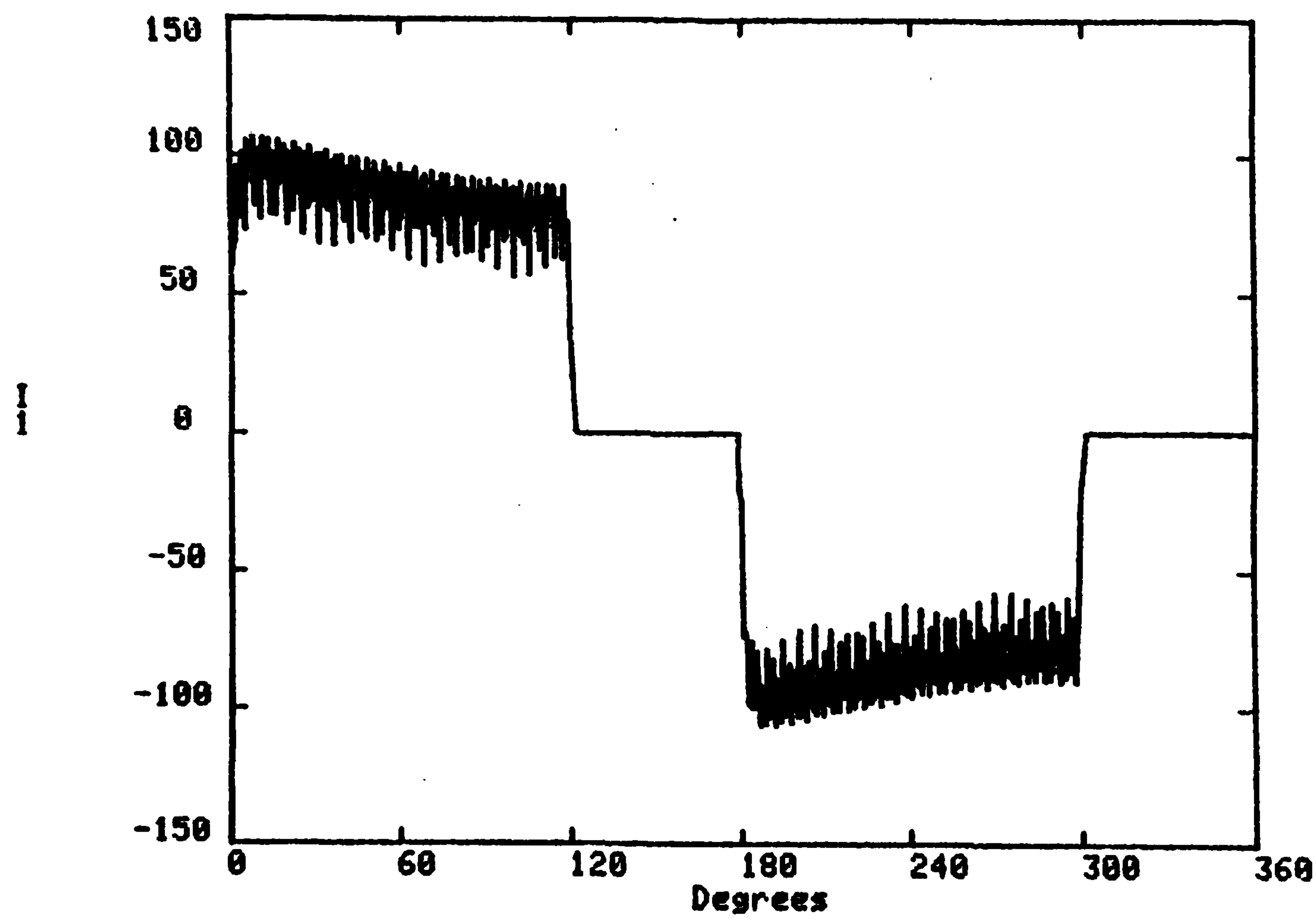


Figure 21. Phase Current - 2250 RPM, No Modulation

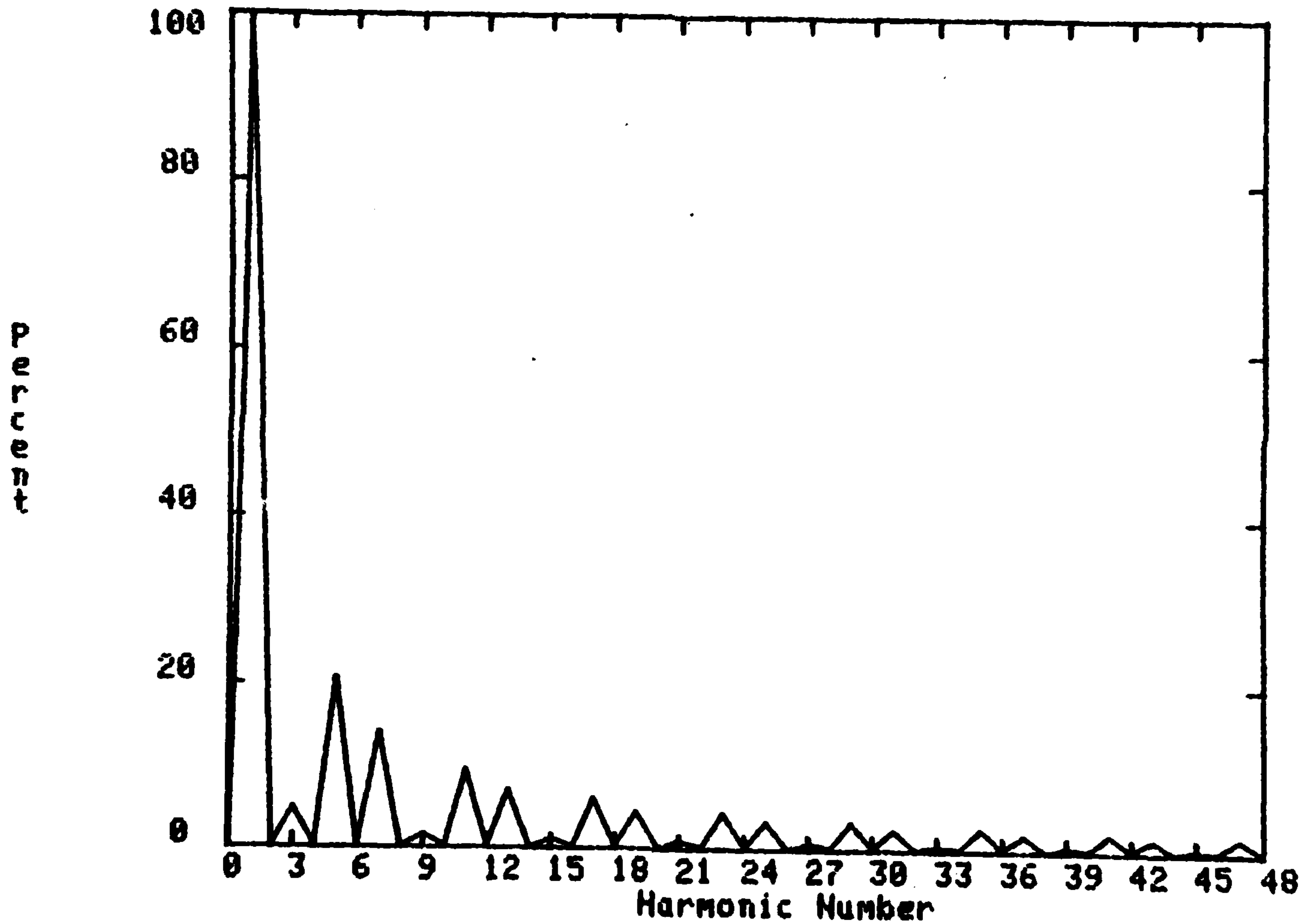


Figure 22. Fourier Spectrum of Phase Current - 2250 RPM, No Modulation

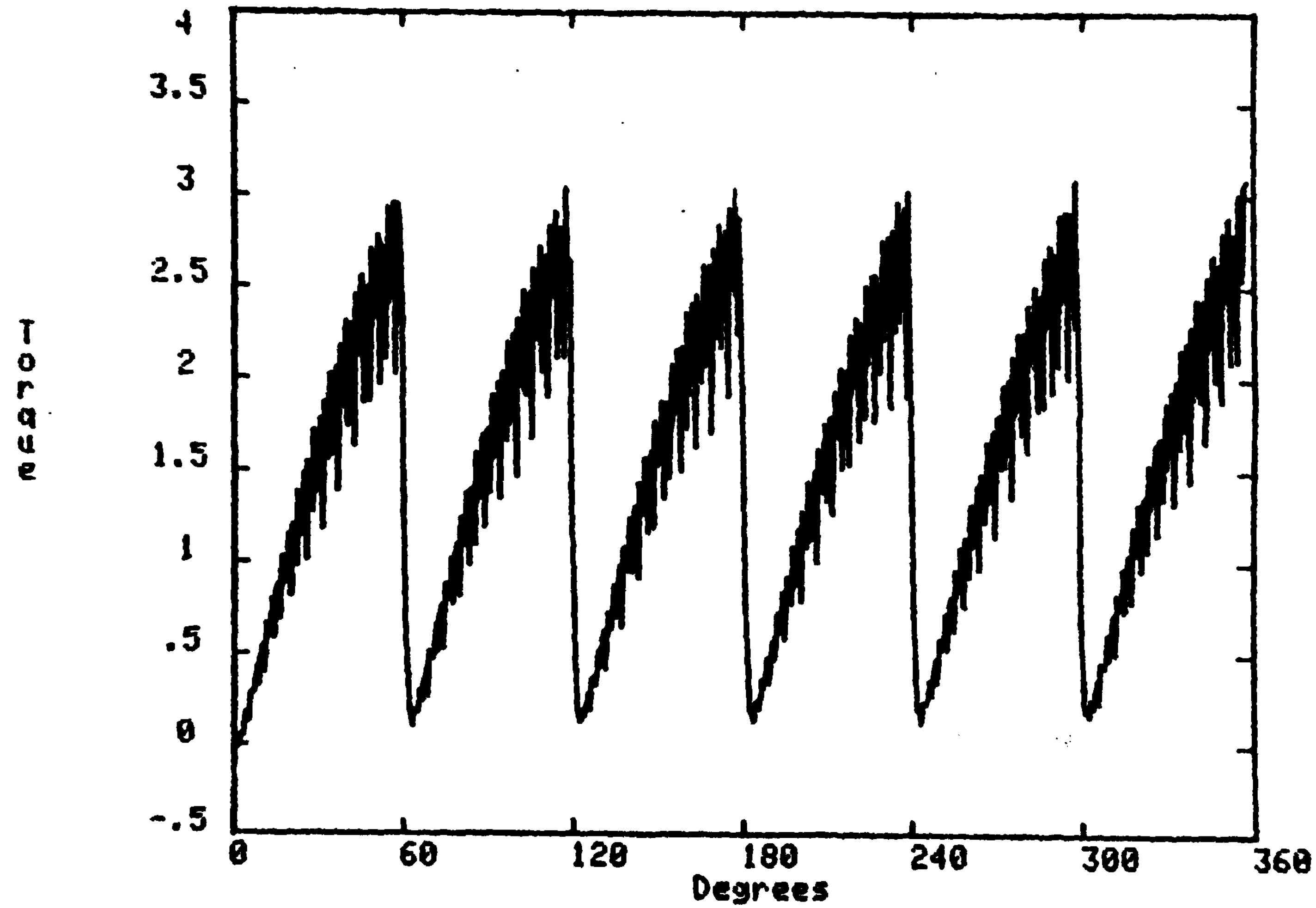


Figure 23. Developed Torque - 2250 RPM, No Modulation

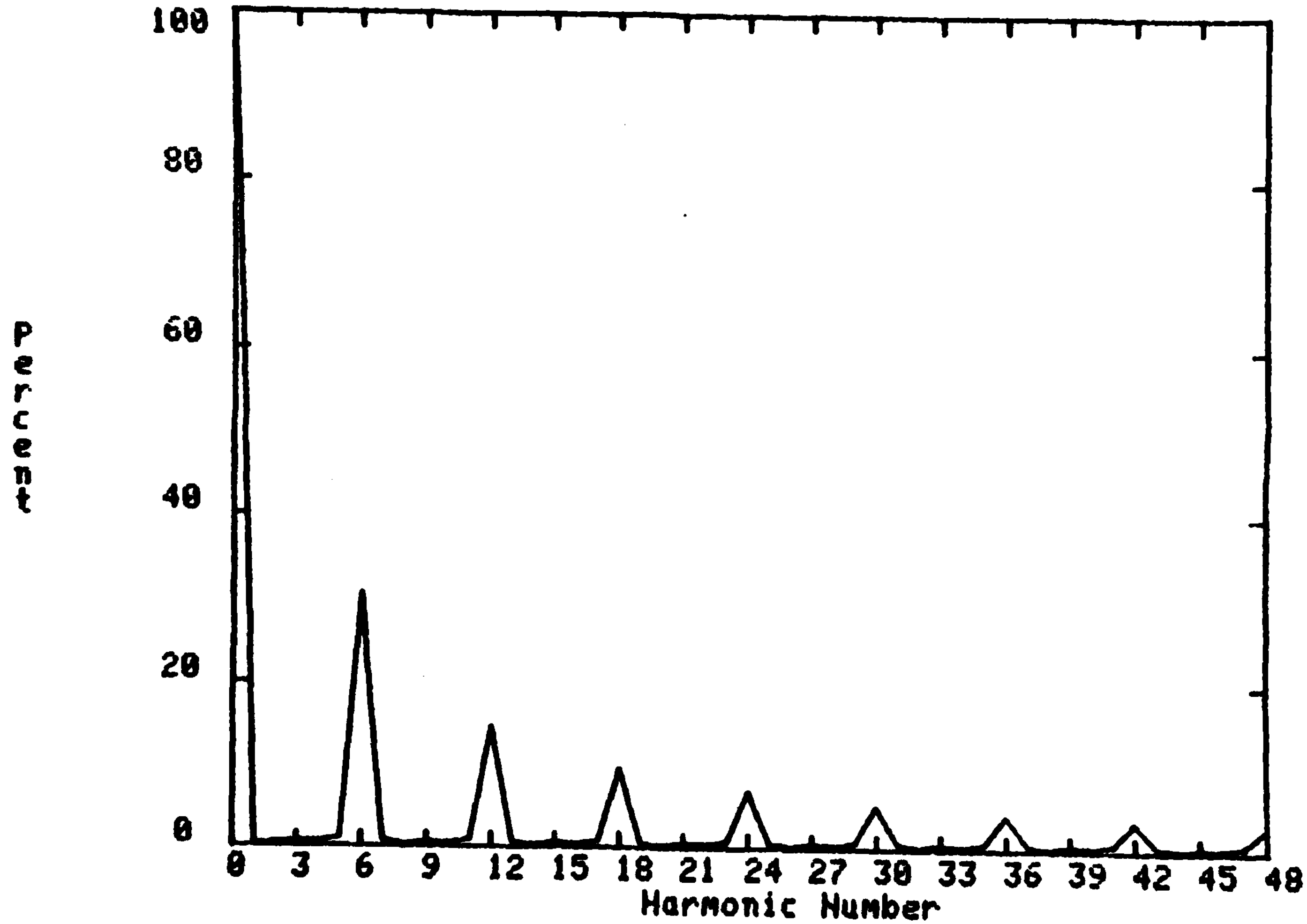


Figure 24. Fourier Spectrum of Developed Torque - 2250 RPM, No Modulation

A plot of phase current i_1 is given by Figure 25. The associated Fourier spectrum is shown by Figure 26.

A plot of the instantaneous developed torque is given by Figure 27. The Fourier spectrum of the developed torque is shown by Figure 28.

2.2.3 Elimination of Sixth and Twelfth Harmonics of Torque. Both the sixth and twelfth harmonics of torque are eliminated by removing the fifth, seventh, eleventh, and thirteenth harmonics of phase current.

Input Data

Motor speed, n_m = 2250 rpm
Motor frequency, f = 75 Hz
SCR firing angle, α_s = 47.31°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 38.73°
α_2 = 42.13°
α_3 = 52.25°
α_4 = 61.93°
α_5 = 66.86°

Output Data

Average torque, T_{dav} = 1.5003 N·m
RMS phase current, I_{RMS} = 74.64 A

A plot of phase current i_1 is given by Figure 29. The associated Fourier spectrum of i_1 is shown by Figure 30.

A plot of the instantaneous developed torque is given by Figure 31. The Fourier spectrum of the developed torque is shown by Figure 32.

2.2.4 Elimination of Sixth and Twelfth Harmonics of Torque (α = 15°). The angle from the onset of positive phase current to the positive going crossing of the associated motor phase counter emf is changed from 30° to 15° to examine effect on current harmonics.

Input Data

Motor speed, n_m = 2250 rpm
Motor frequency, f = 75 Hz
SCR firing angle, α_s = 59.84°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 38.73°
α_2 = 42.13°
α_3 = 52.25°
α_4 = 61.93°
α_5 = 38.73°

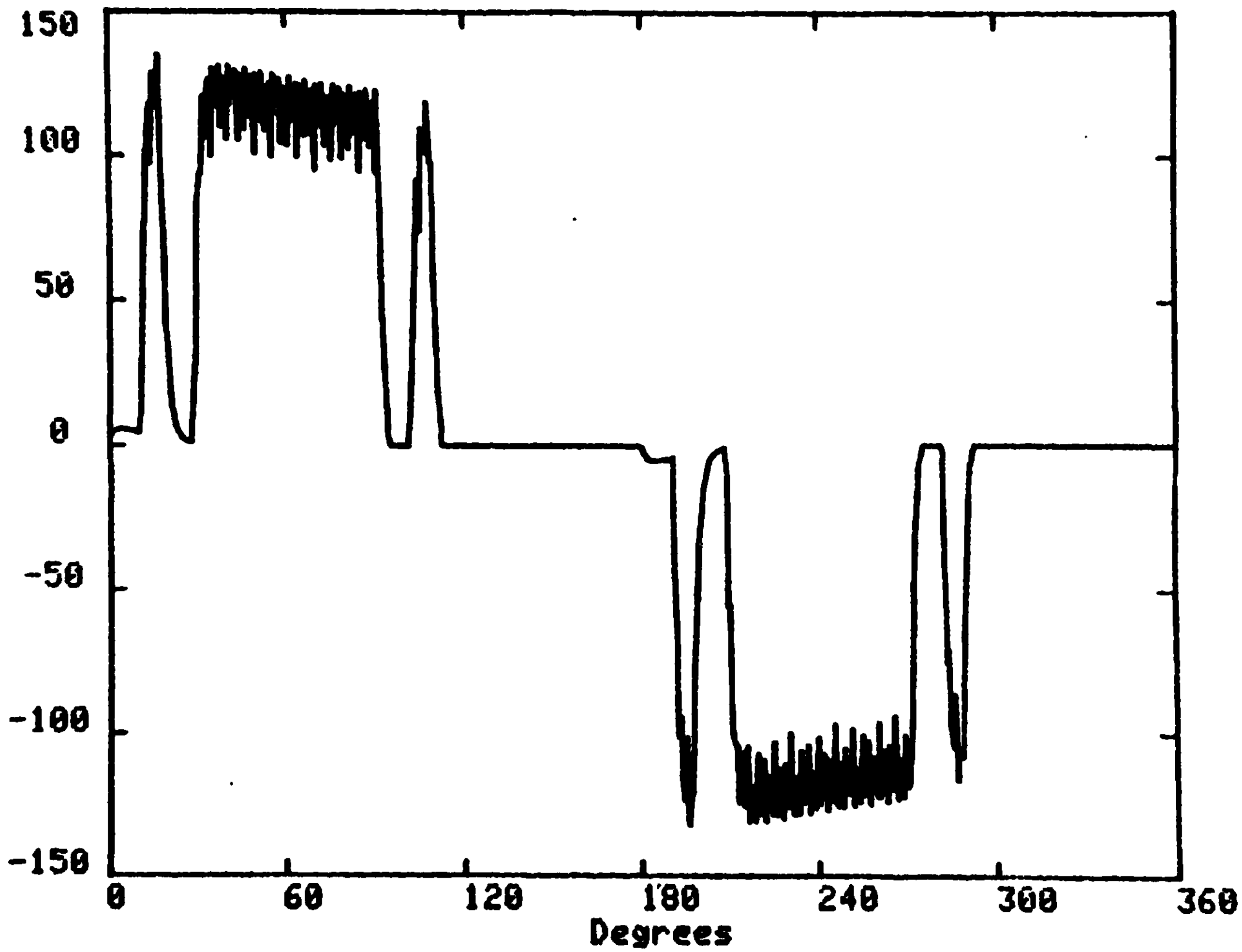


Figure 25. Phase Current - 2250 RPM, Fifth and Seventh Harmonics Eliminated

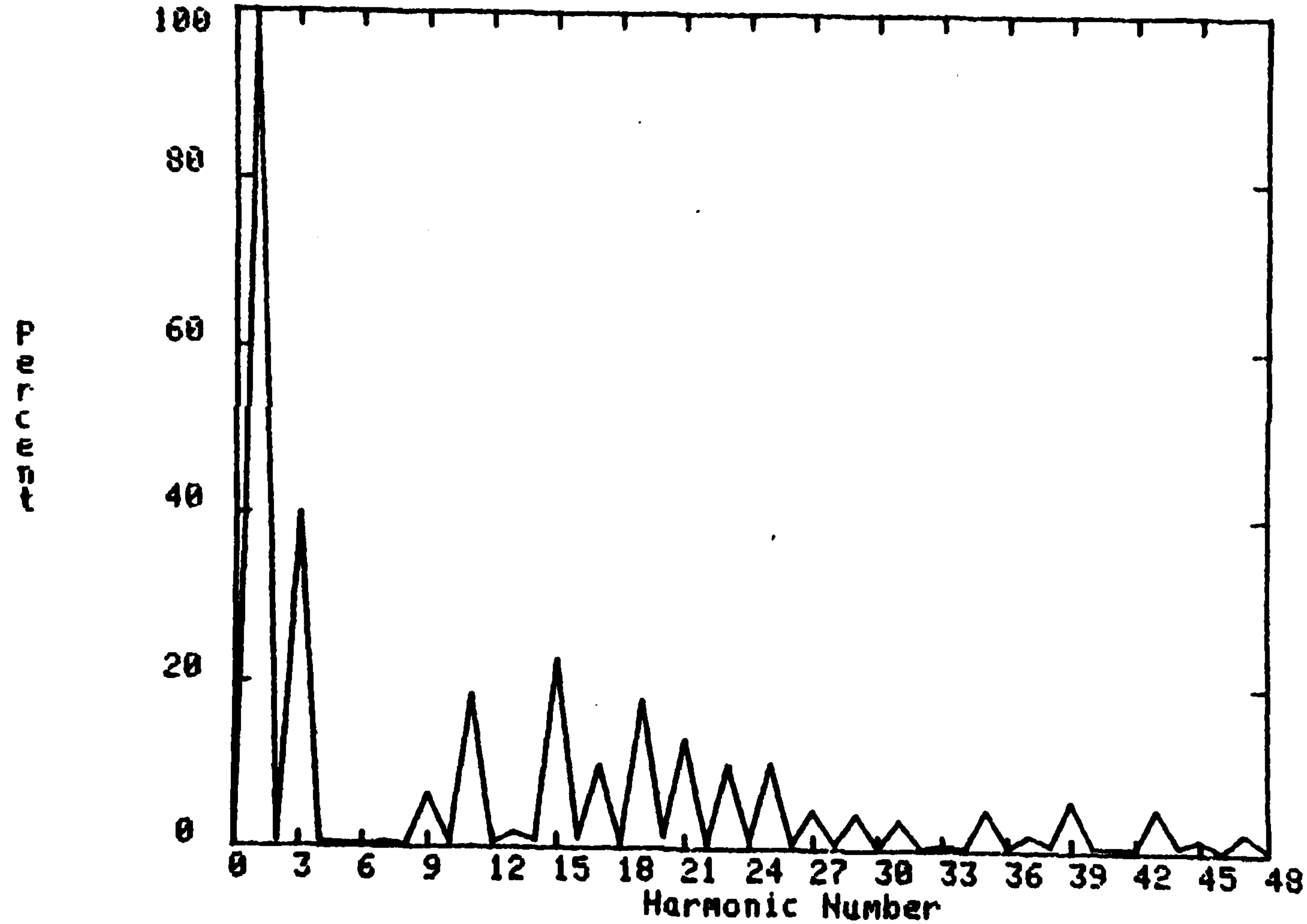


Figure 26. Fourier Spectrum of Phase Current - 2250 RPM, Fifth and Seventh Harmonics Eliminated

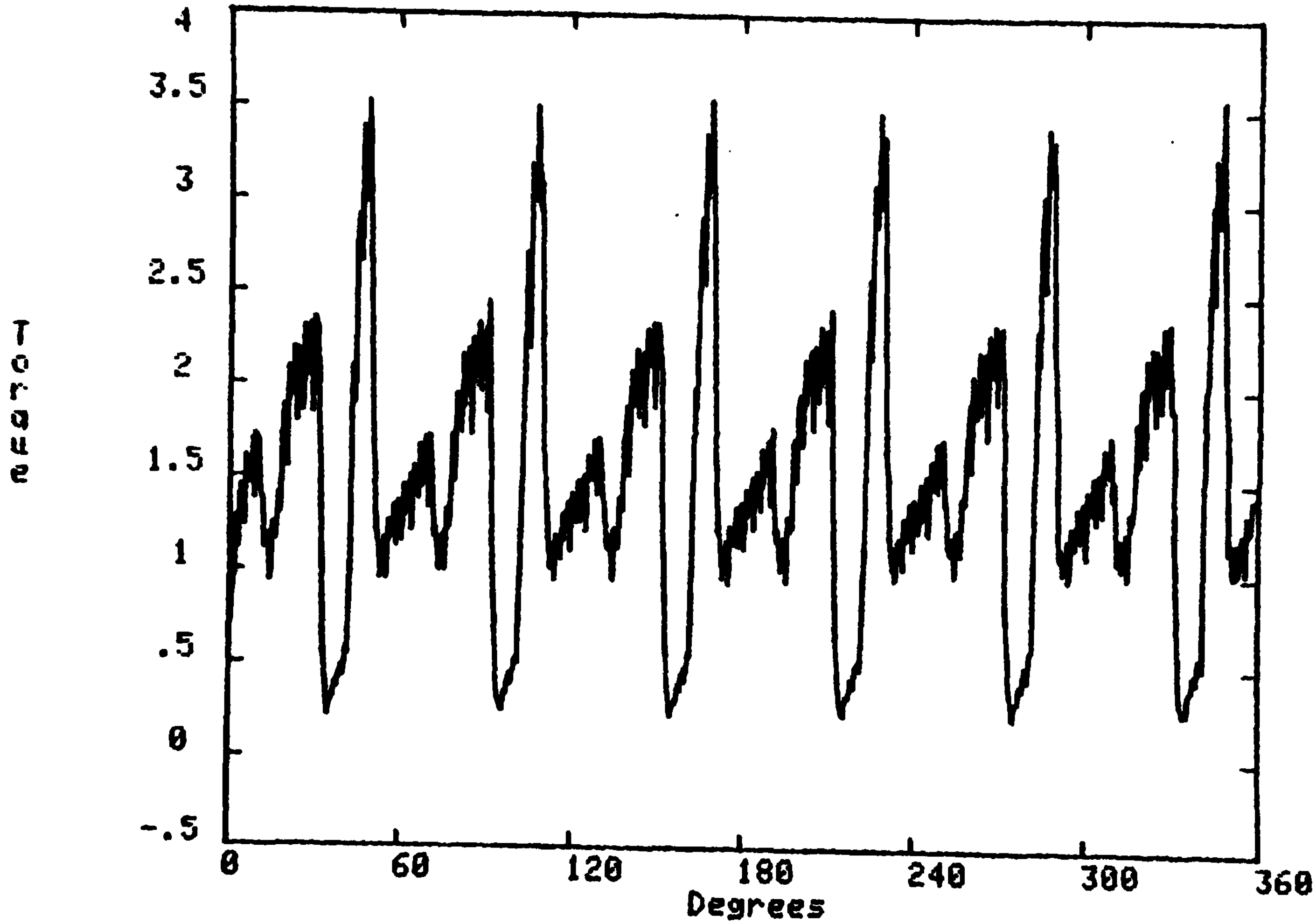


Figure 27. Developed Torque - 2250 RPM, Sixth Harmonic Eliminated

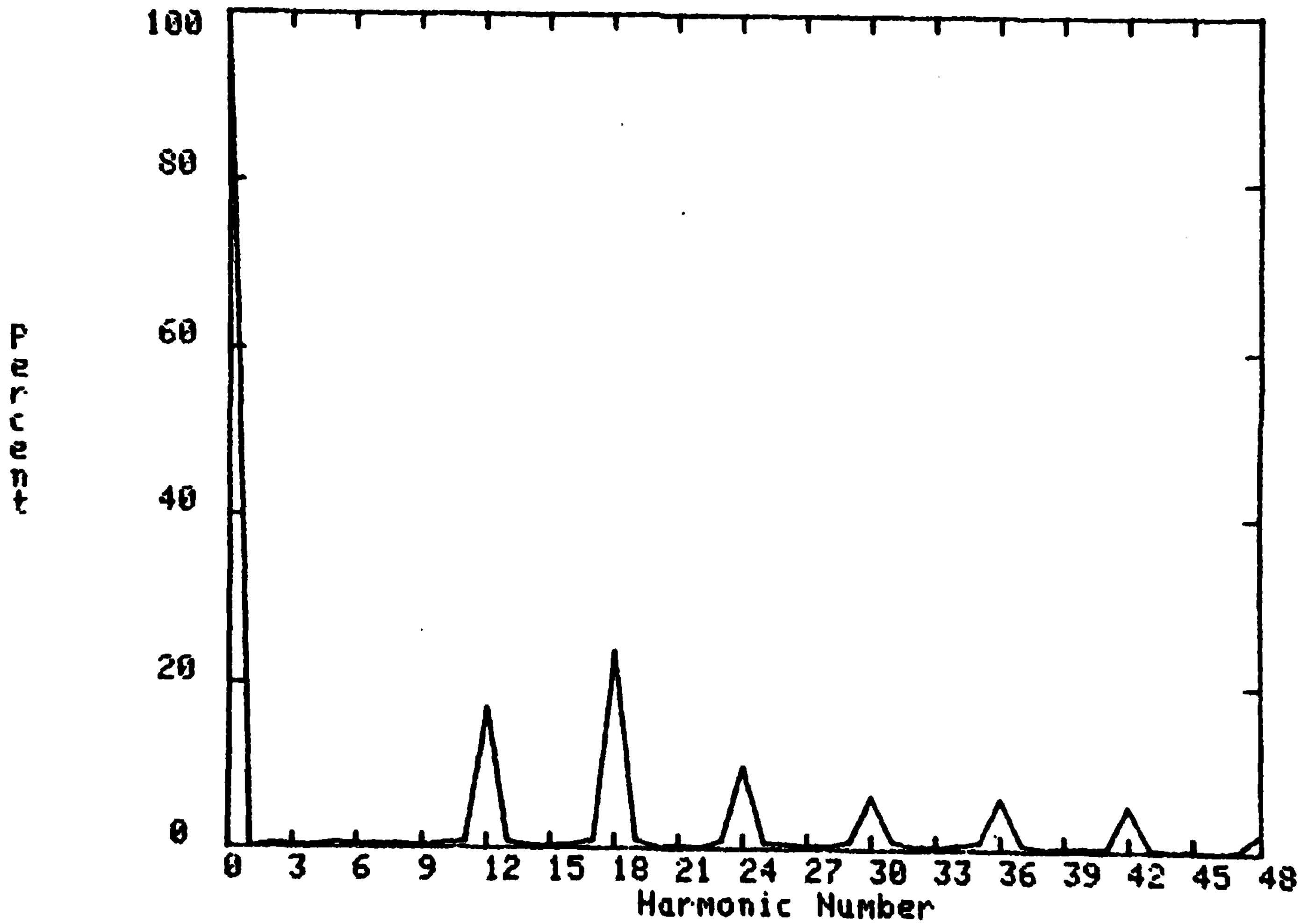


Figure 28. Fourier Spectrum of Developed Torque - 2250 RPM, Sixth Harmonic Eliminated

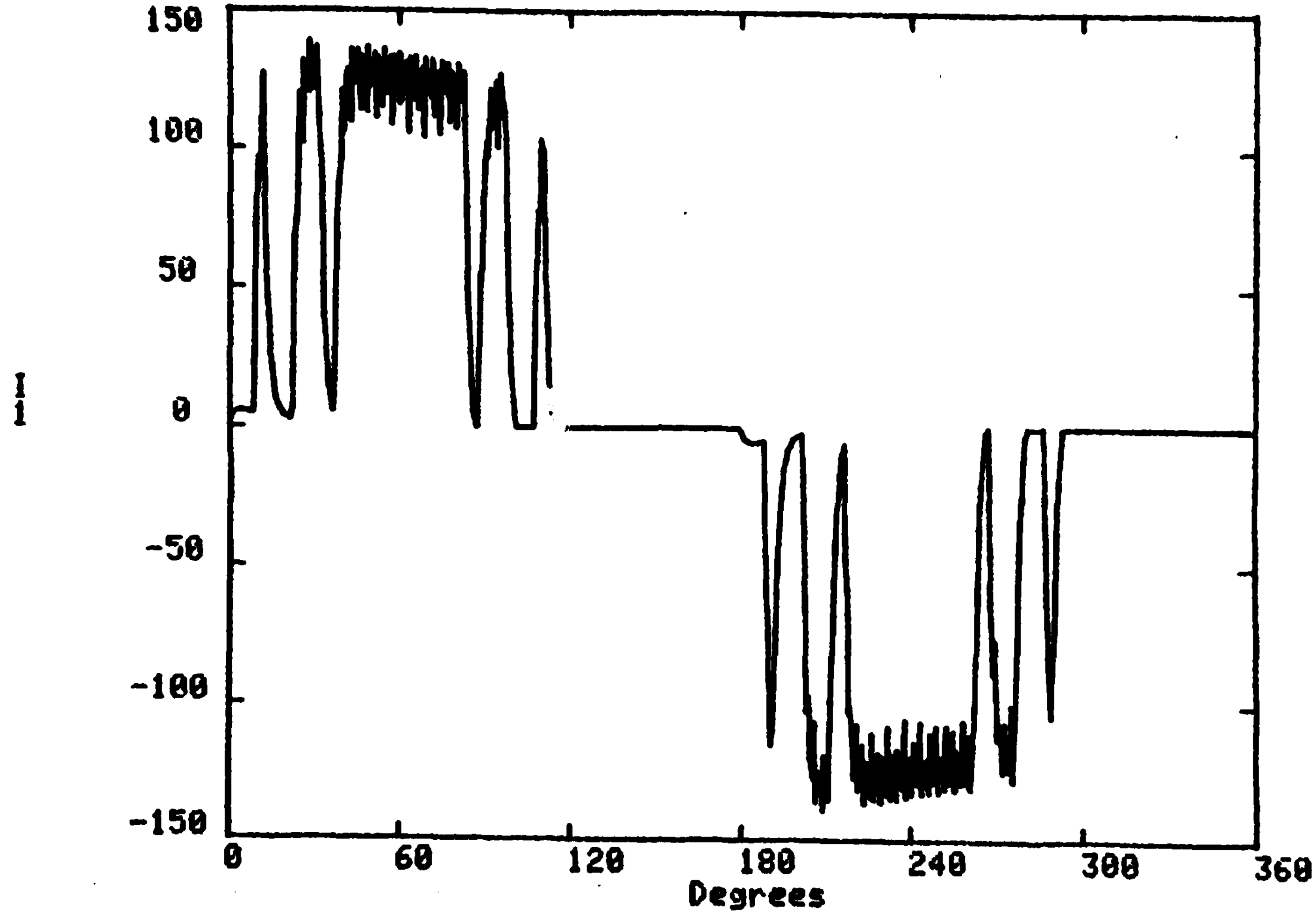


Figure 29. Phase Current - 2250 RPM, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

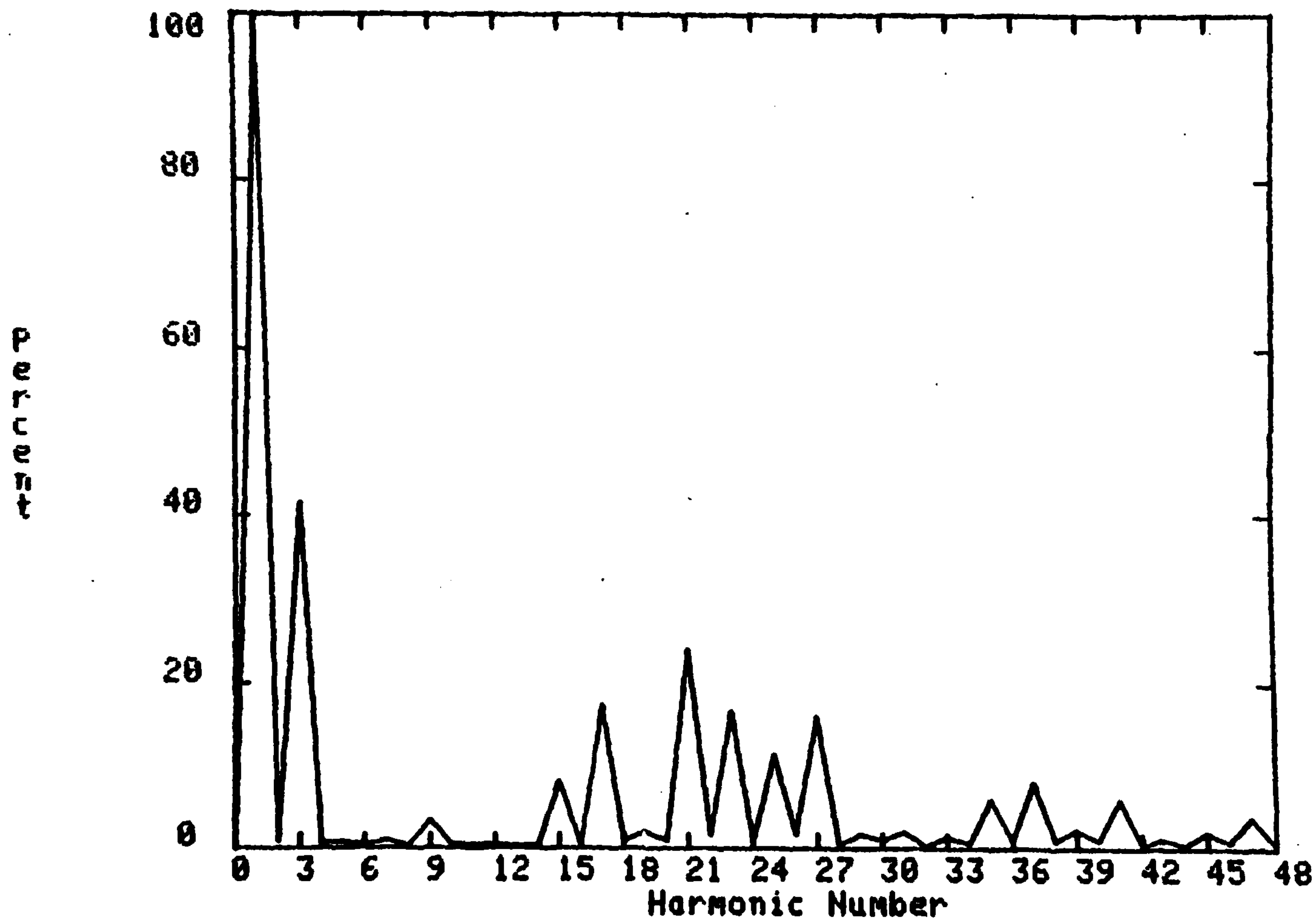


Figure 30. Fourier Spectrum of Phase Current - 2250 RPM, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

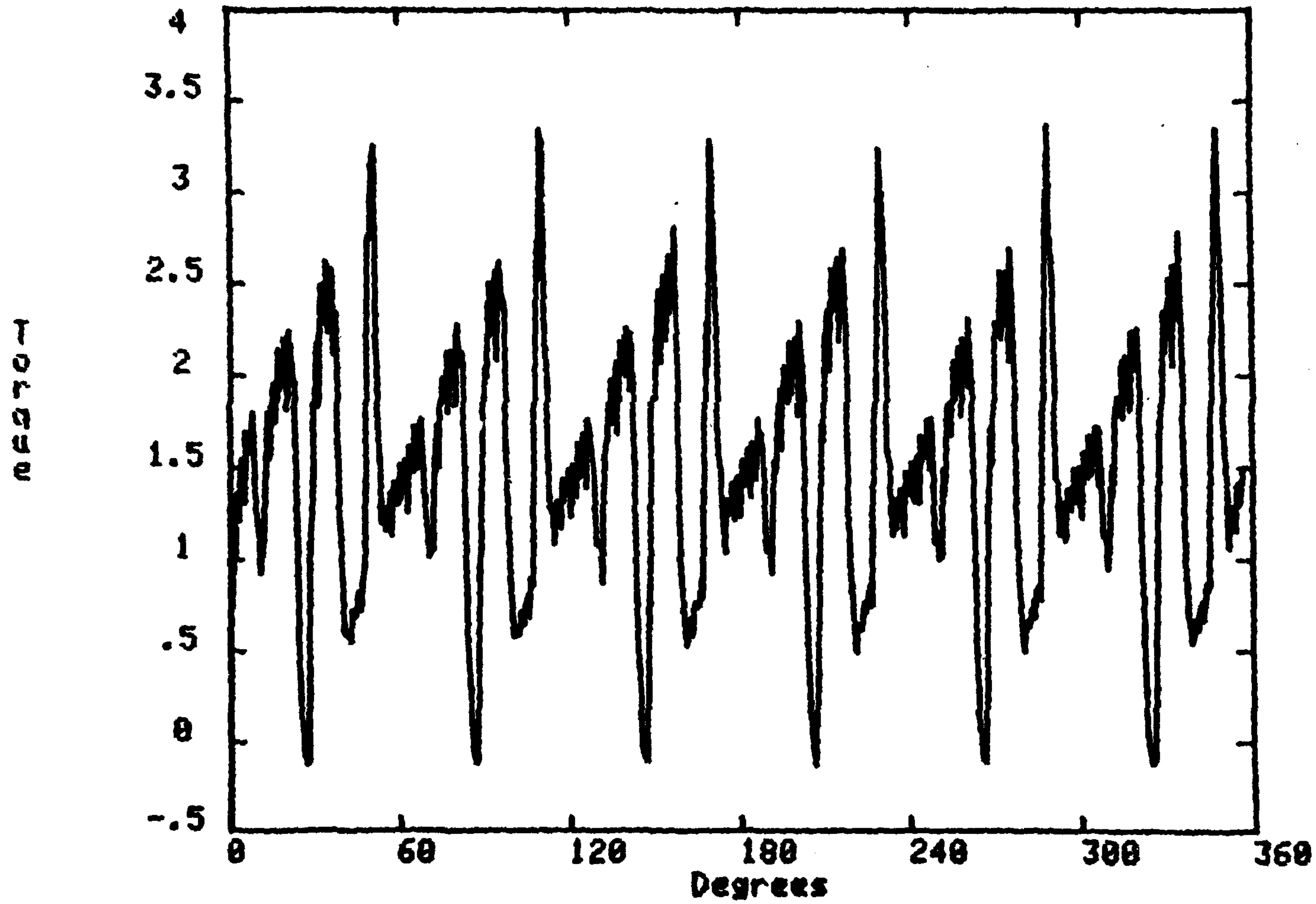


Figure 31. Developed Torque - 2250 RPM, Sixth and Twelfth Harmonics Eliminated

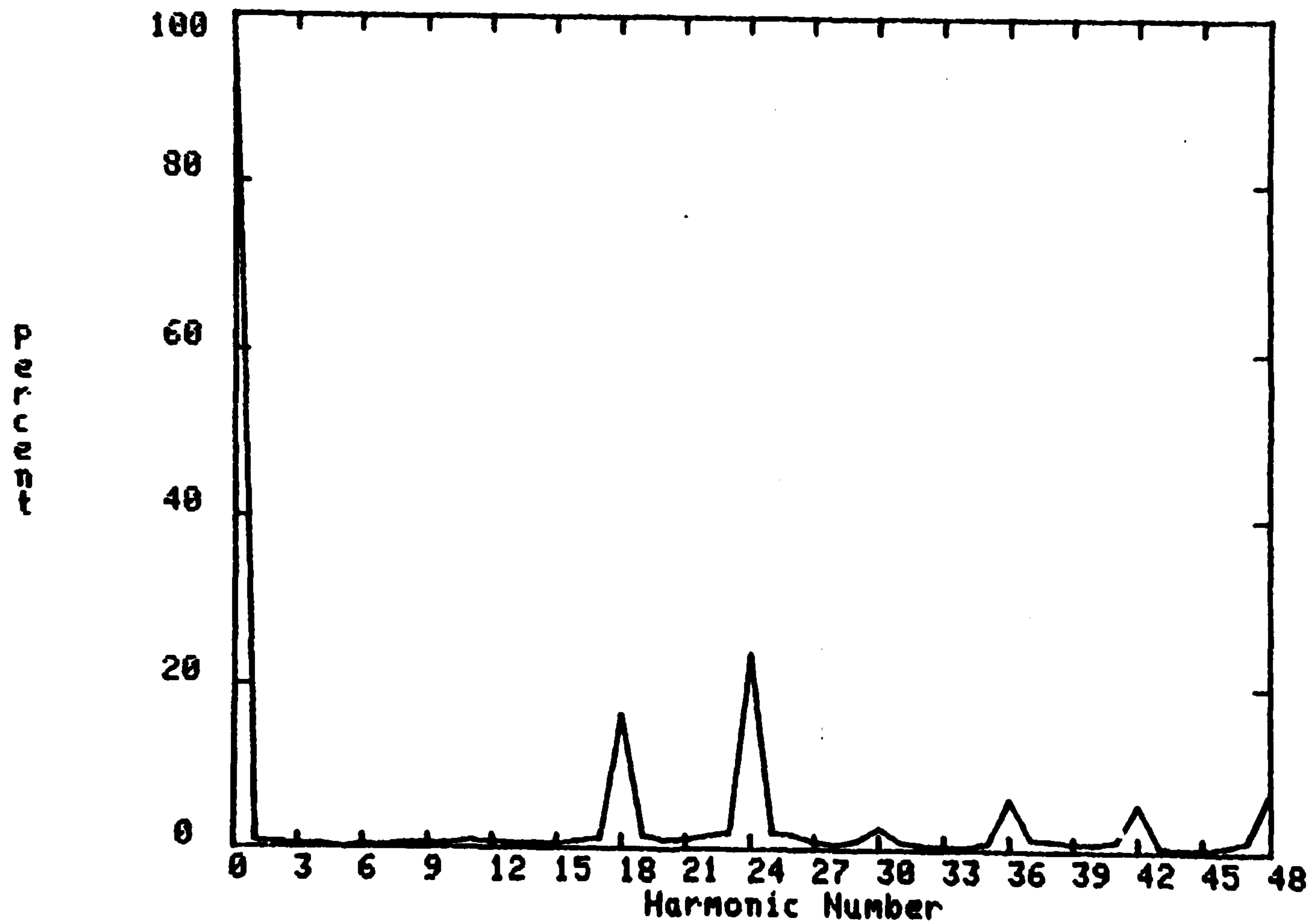


Figure 32. Fourier Spectrum of Developed Torque - 2250 RPM, Sixth and Twelfth Harmonics Eliminated

Output Data

Average torque, $T_{dav} = 1.502$ N·m
RMS phase current, $I_{RMS} = 53.33$ A

A plot of phase current i_1 is given by Figure 33. The associated Fourier spectrum is shown by Figure 34.

A plot of instantaneous developed torque is given by Figure 35. The Fourier spectrum of the developed torque is shown by Figure 36.

The re-introduced harmonic magnitudes are reduced to a lower level than for $\alpha = 30^\circ$.

2.2.5 Elimination of Sixth and Twelfth Harmonics of Torque ($\alpha = 40^\circ$). The angle from onset of positive phase current to the positive going crossing of the associated motor phase counter emf is set at 40° to examine the effect on current harmonics.

Input Data

Motor speed, n_m = 2250 rpm
Motor frequency, f = 75 Hz
SCR firing angle, $\alpha_s = 16.77^\circ$
Specified average torque, $T_{dav} = 1.5$ N·m
Angles of modulation function, $\alpha_1 = 38.73^\circ$
$\alpha_2 = 42.13^\circ$
$\alpha_3 = 52.25^\circ$
$\alpha_4 = 61.93^\circ$
$\alpha_5 = 66.86^\circ$

Output Data

Average torque, $T_{dav} = 1.5003$ N·m
RMS phase current, $I_{RMS} = 107.52$

A plot of phase current i_1 is given by Figure 37. The associated Fourier spectrum of phase current is shown by Figure 38.

A plot of instantaneous developed torque is given by Figure 39. The Fourier spectrum of developed torque is shown by Figure 40.

Although the angle power factor angle ($\alpha + 30^\circ$) has been extended to 75° resulting in a large current to maintain the same power flow, no significant re-introduction of undesired harmonics is noted.

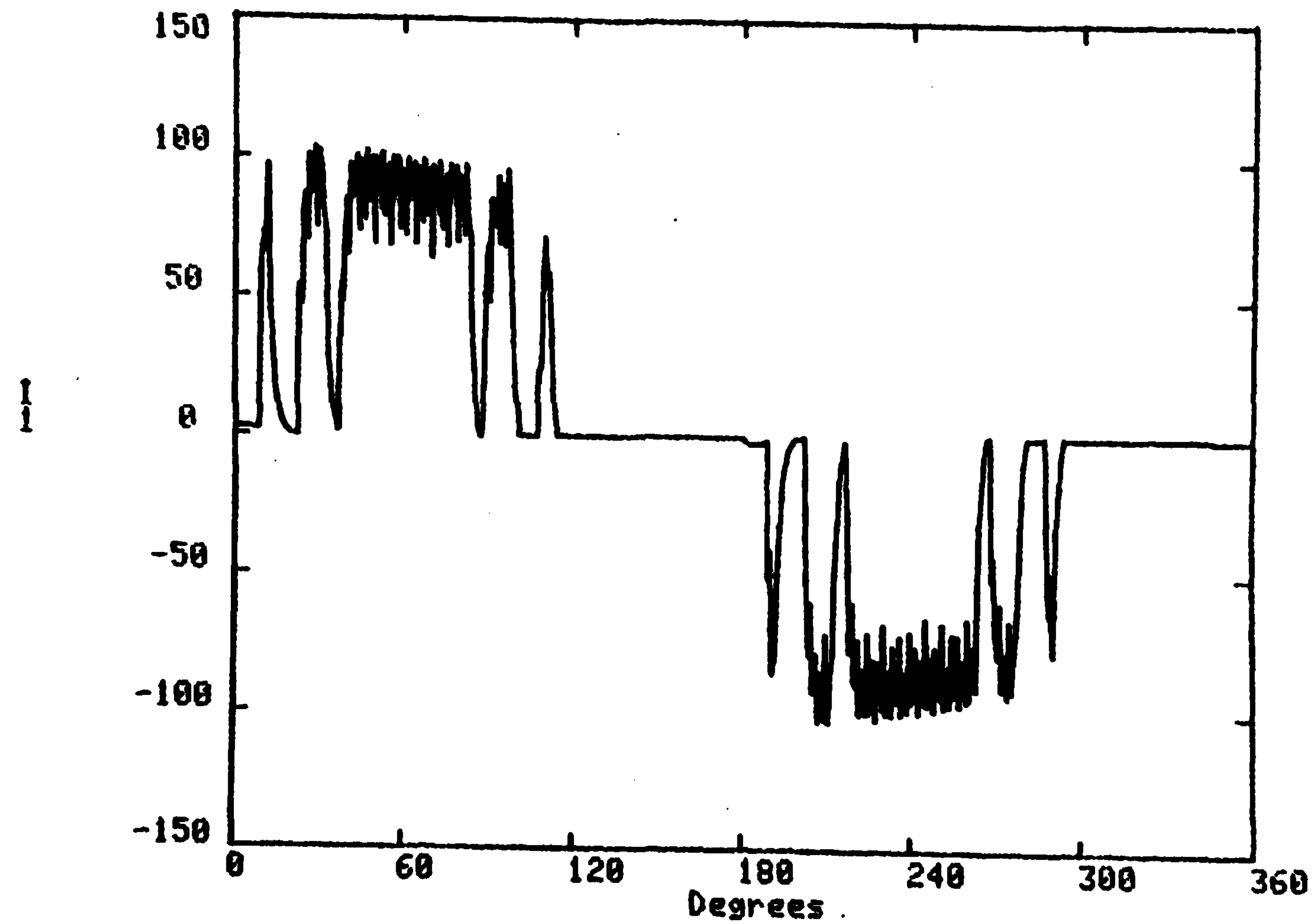


Figure 33. Phase Current - 2250 RPM, $\alpha = 15^o$, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

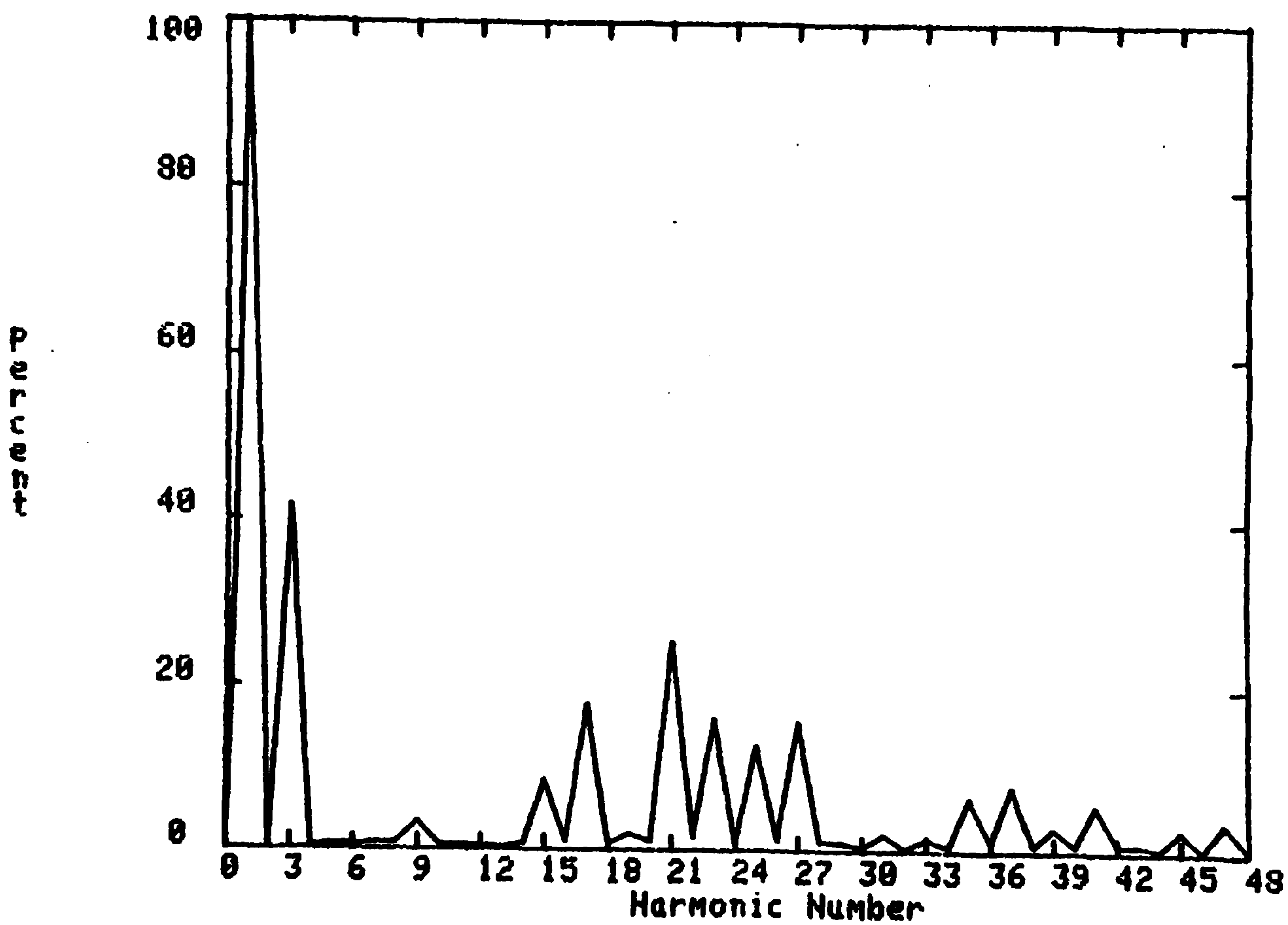


Figure 34. Fourier Spectrum of Phase Current - 2250 RPM, $\alpha = 15^o$, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

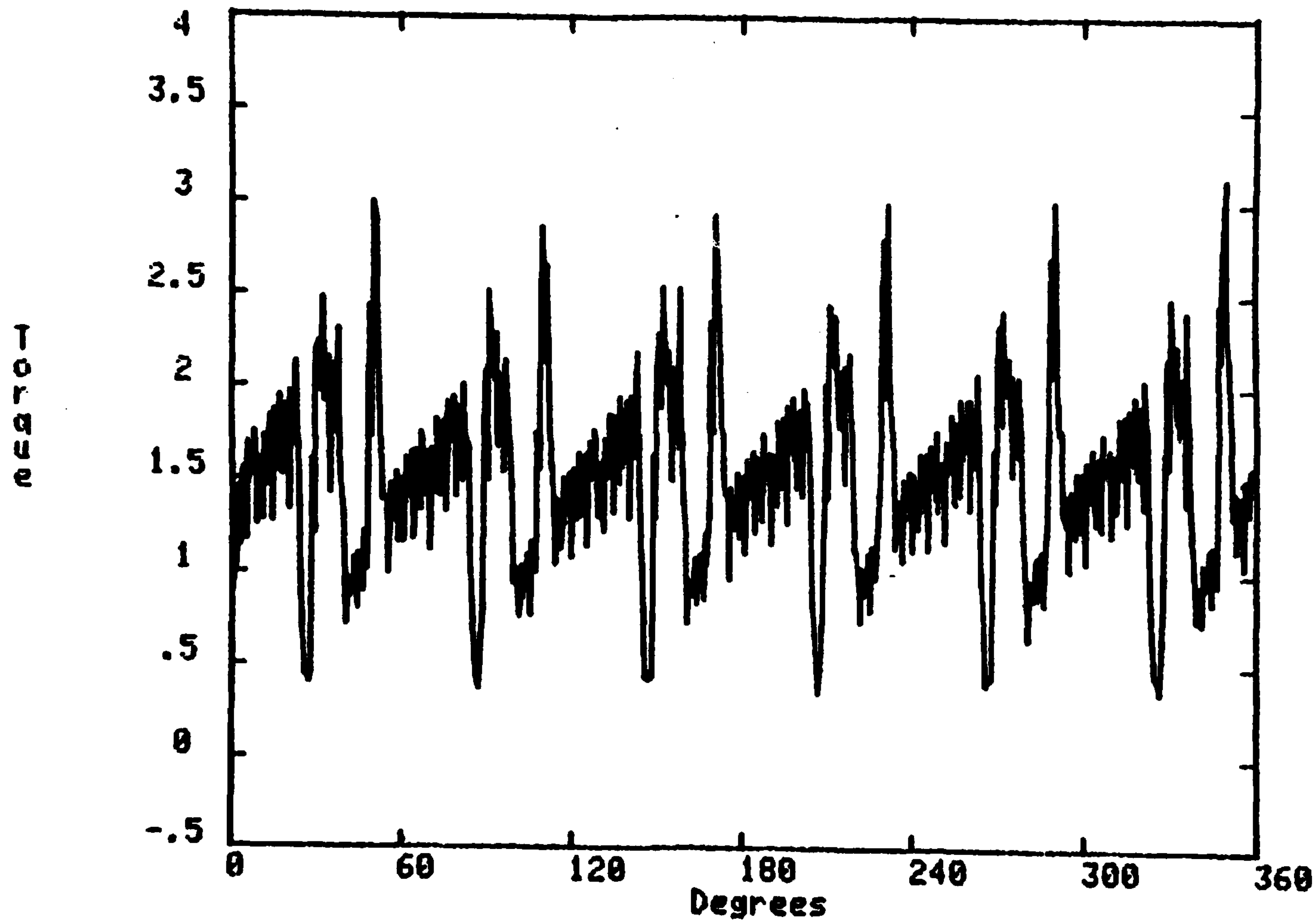


Figure 35. Developed Torque - 2250 RPM, α = 15°, Sixth and Twelfth Harmonics Eliminated

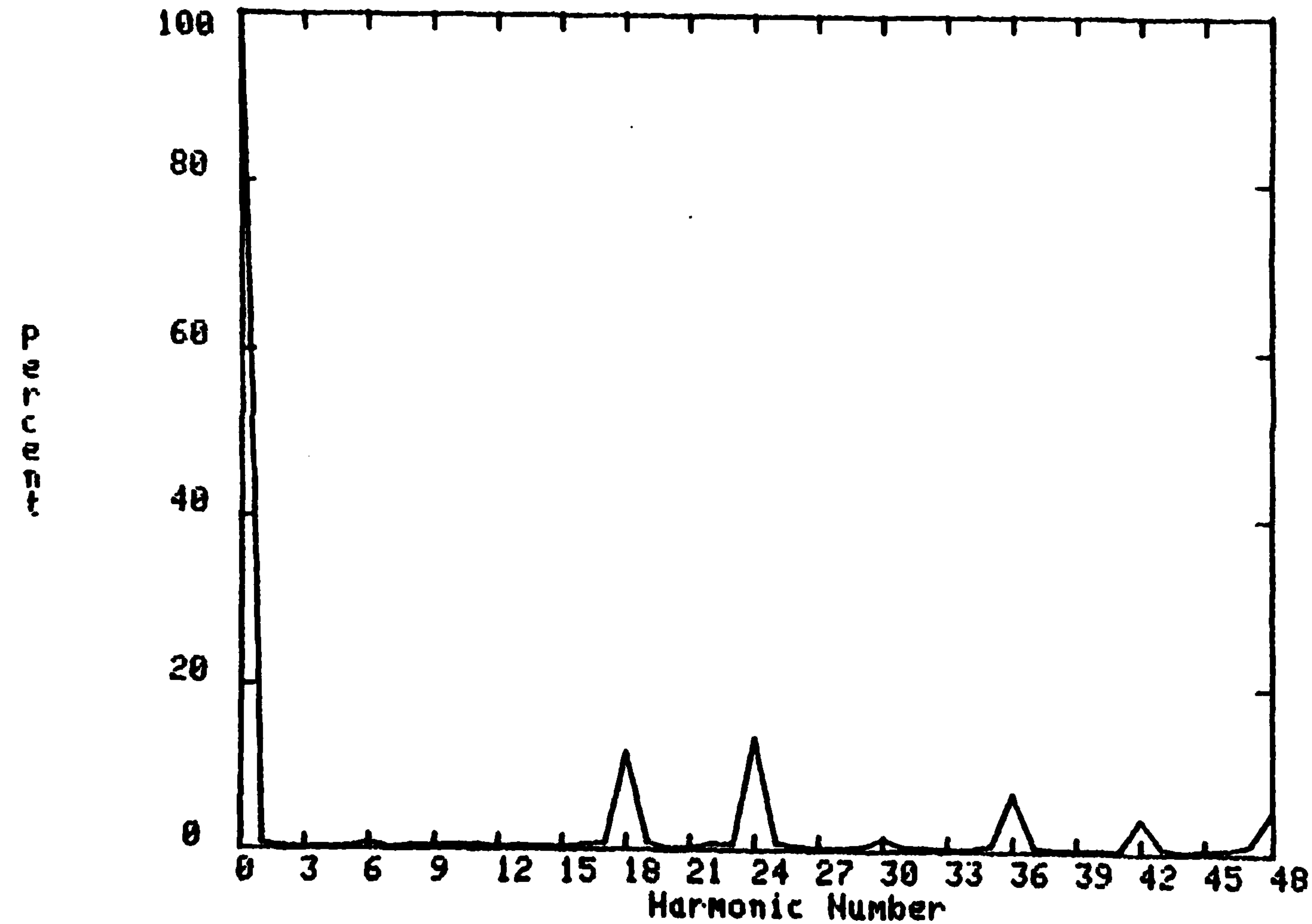


Figure 36. Fourier Spectrum of Developed Torque - 2250 RPM, α = 15°, Sixth and Twelfth Harmonics Eliminated

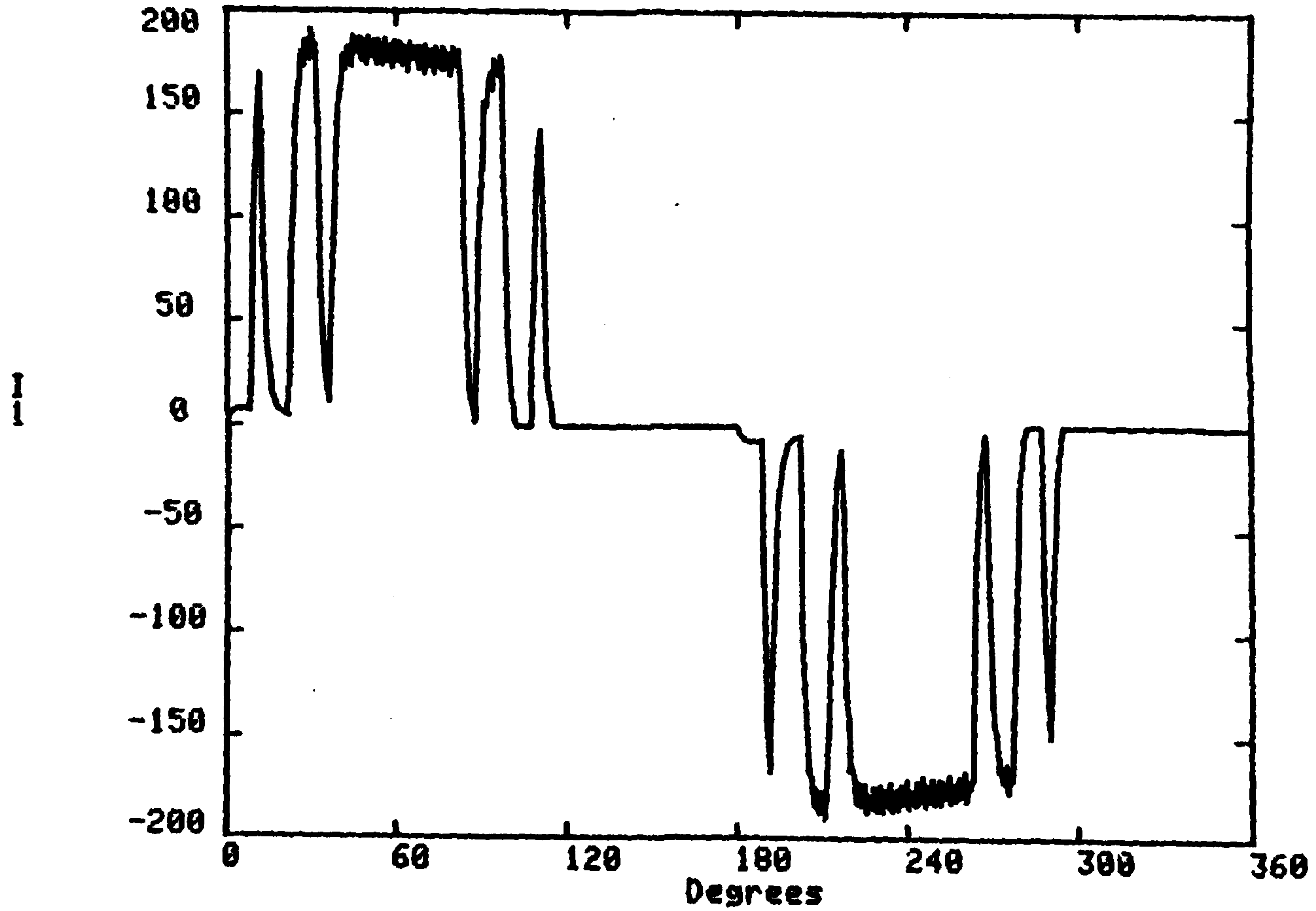


Figure 37. Phase Current - 2250 RPM, $\alpha = 40^{o}$, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

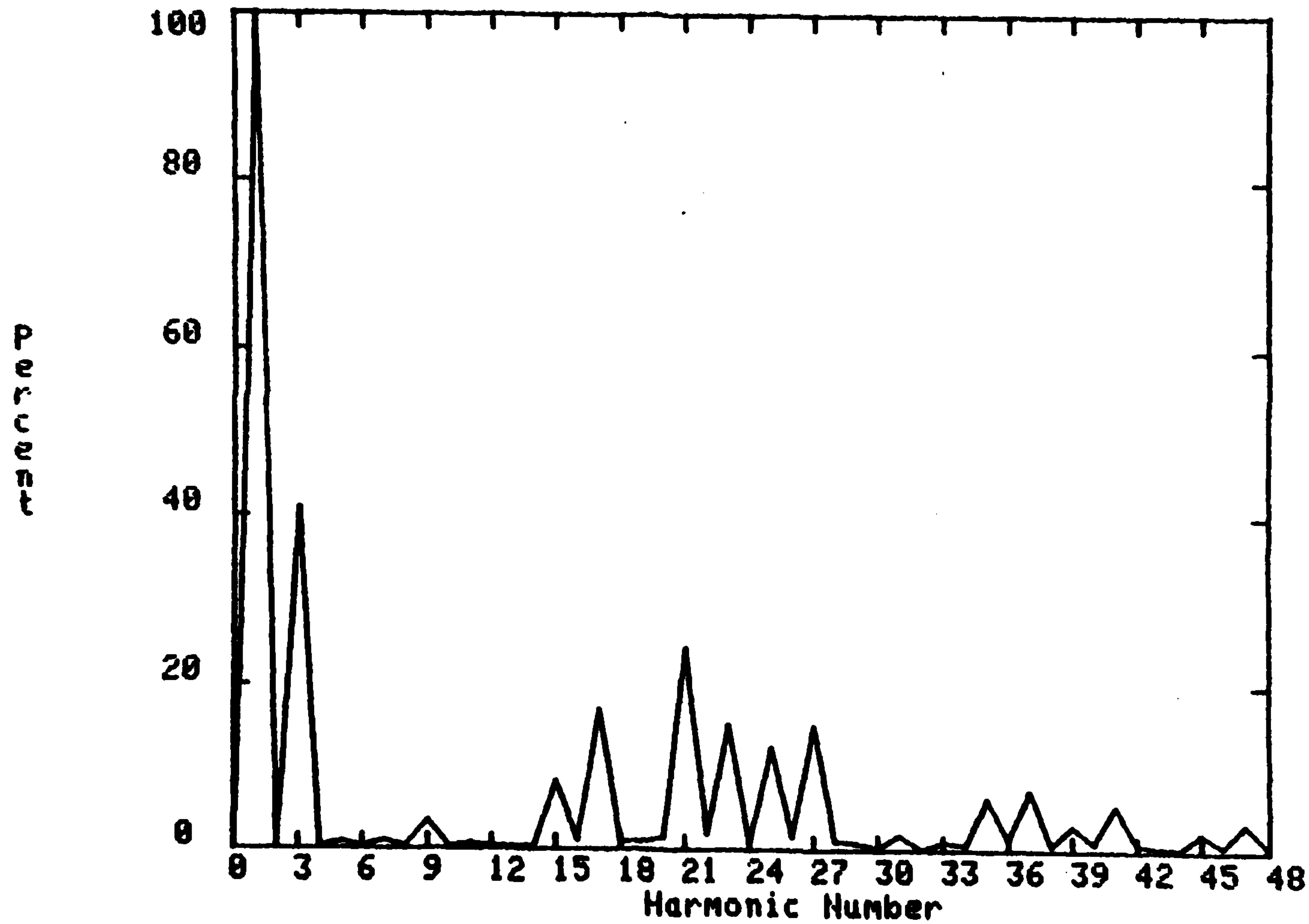


Figure 38. Fourier Spectrum of Phase Current - 2250 RPM, $\alpha = 40^{o}$, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

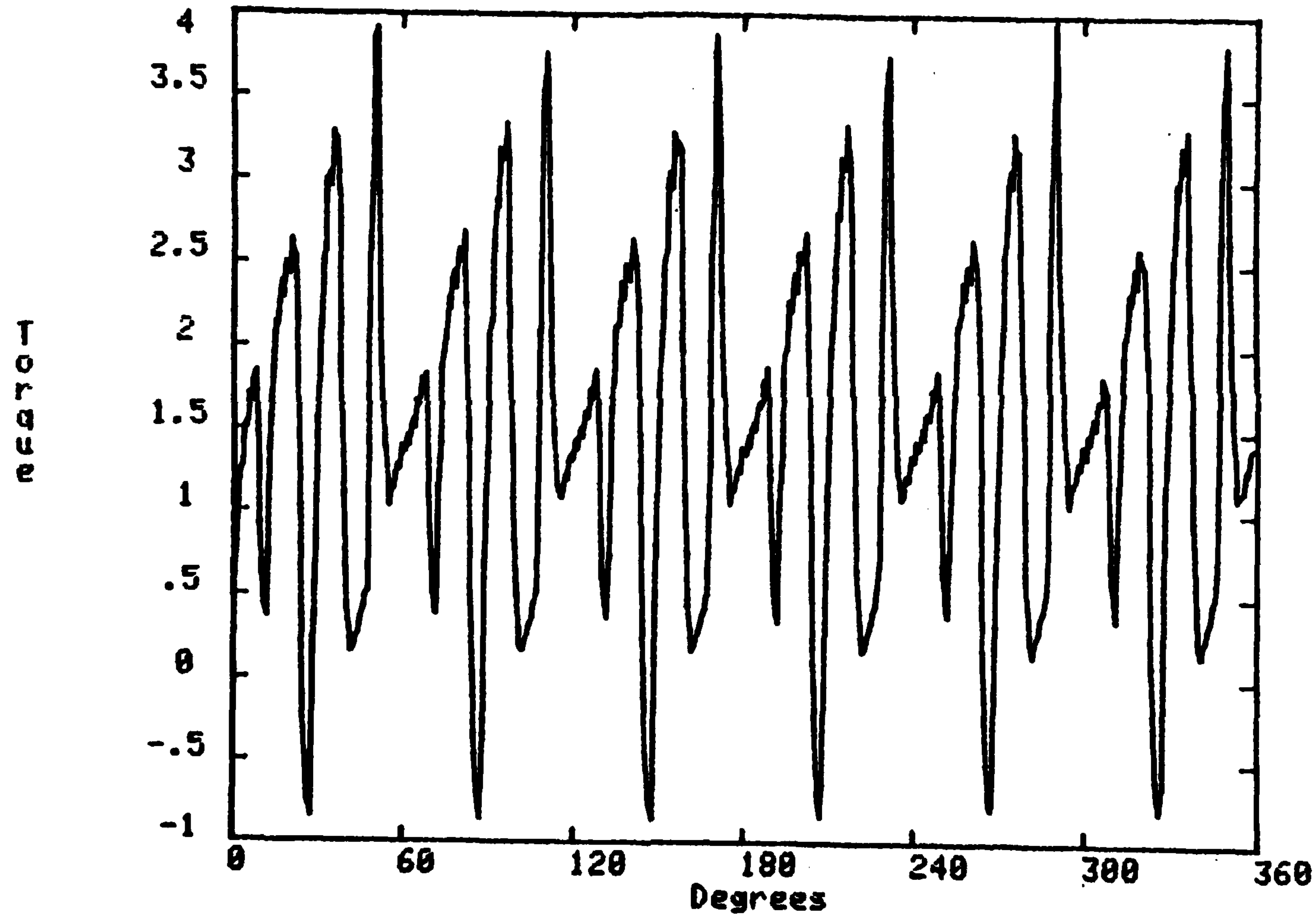


Figure 39. Developed Torque - 2250 RPM, $\alpha = 40^{\circ}$, Sixth and Twelfth Harmonics Eliminated

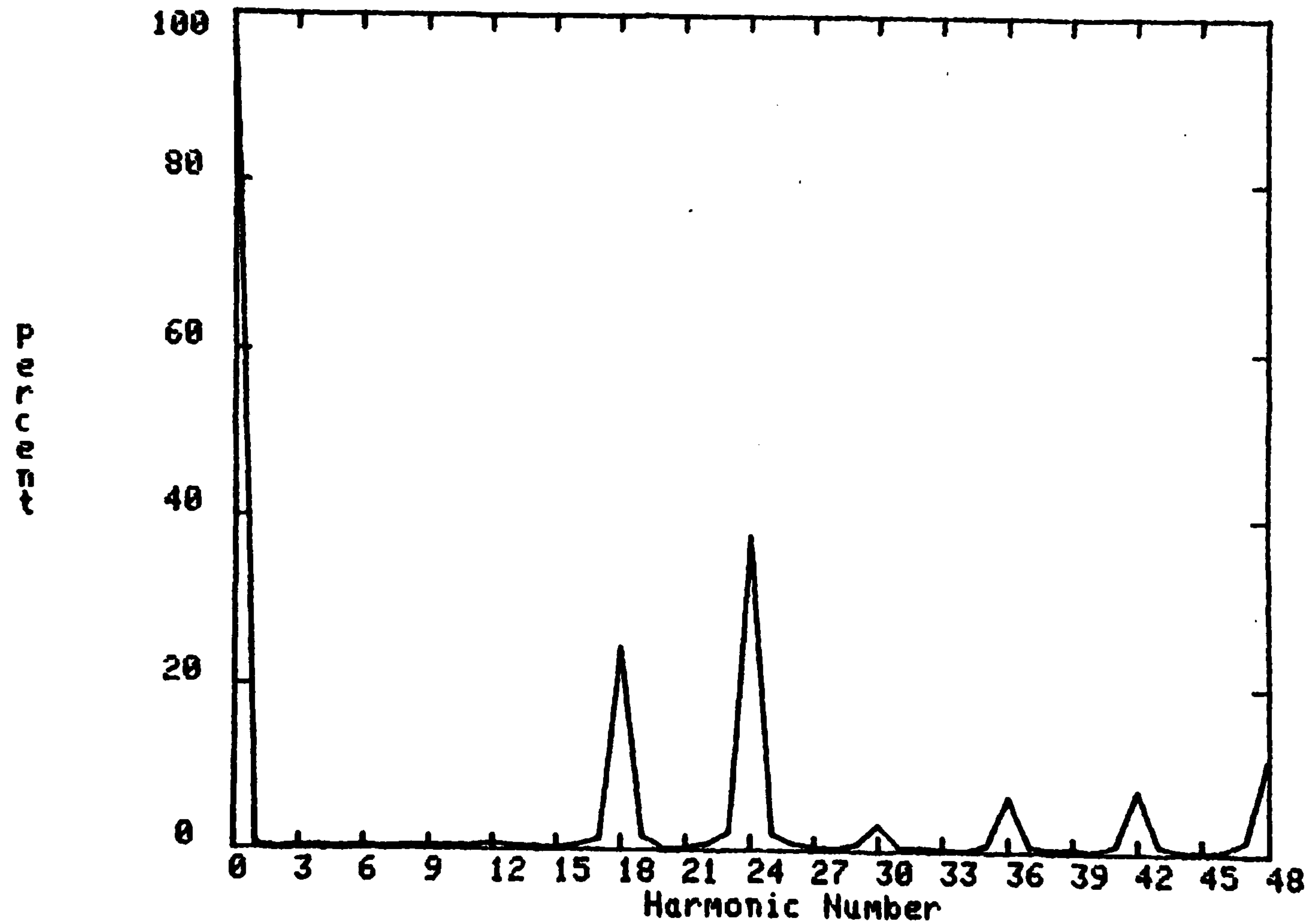


Figure 40. Fourier Spectrum of Developed Torque - 2250 RPM, $\alpha = 40^{\circ}$, Sixth and Twelfth Harmonics Eliminated

2.3 450 RPM (1% Speed Case)

2.3.1 Unmodulated Phase Voltage.

Input Data

Motor speed, n_m = 450 rpm
Motor frequency, f = 15 Hz
SCR firing angle, α_s = 64.52°
Specified average torque, T_{dav} = 1.5 N·m

Output Data

Average torque, T_{dav} = 1.4996 N·m
RMS phase current, I_{RMS} = 67.1 A

A plot of phase current i_1 is given by Figure 41. The associated Fourier spectrum is shown by Figure 42.

A plot of instantaneous developed torque is given by Figure 43. A Fourier spectrum of developed torque is shown by Figure 44.

The instantaneous wave form plots for the low speed case do not display all of the ripples due to the lack of capacity of the plot routine to store sufficient points.

2.3.2 Elimination of Sixth Harmonic of Torque. The sixth harmonic of torque is eliminated by removing the fifth and seventh harmonics of phase current.

Input Data

Motor speed, n_m = 450 rpm
Motor frequency, f = 15 Hz
SCR firing angle, α_s = 52.53°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 40.76°
α_2 = 47.73°
α_3 = 58.65°

Output Data

Average torque, T_{dav} = 1.4996 N·m
RMS phase current, I_{RMS} = 75.2 A

A plot of phase current i_1 is given by Figure 45. The associated Fourier spectrum is shown by Figure 46.

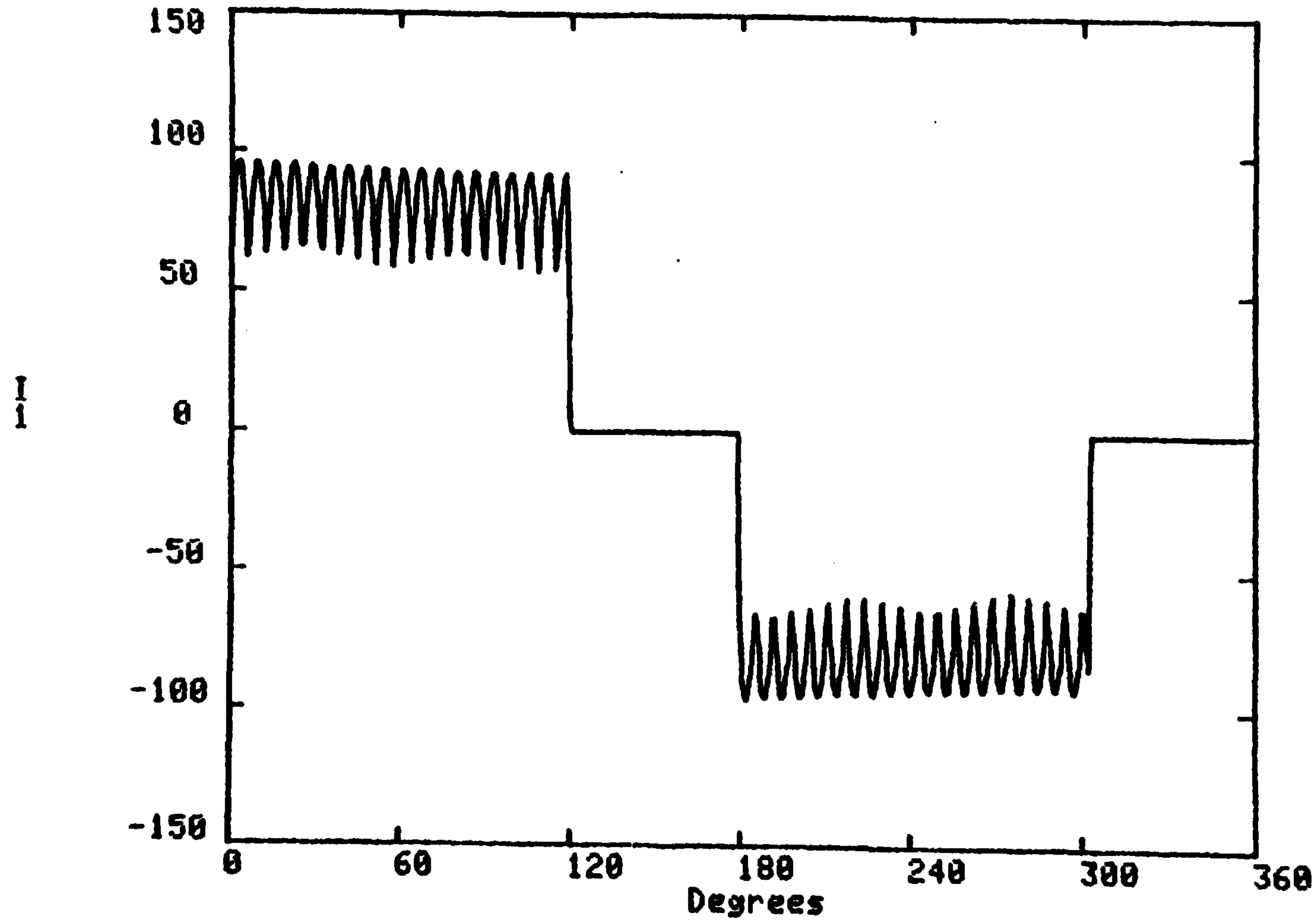


Figure 41. Phase Current - 450 RPM, No Modulation

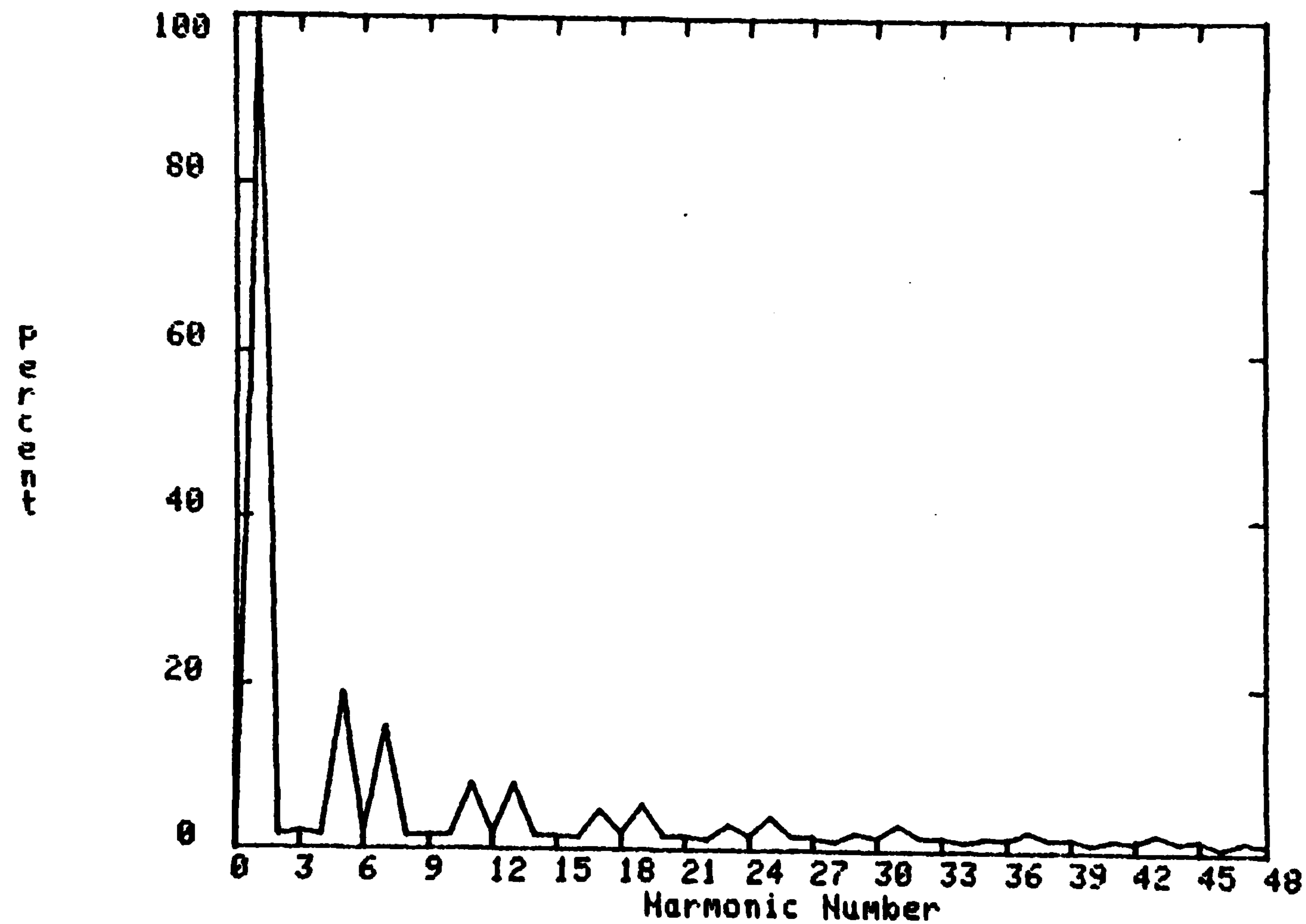


Figure 42. Fourier Spectrum of Phase Current - 450 RPM, No Modulation

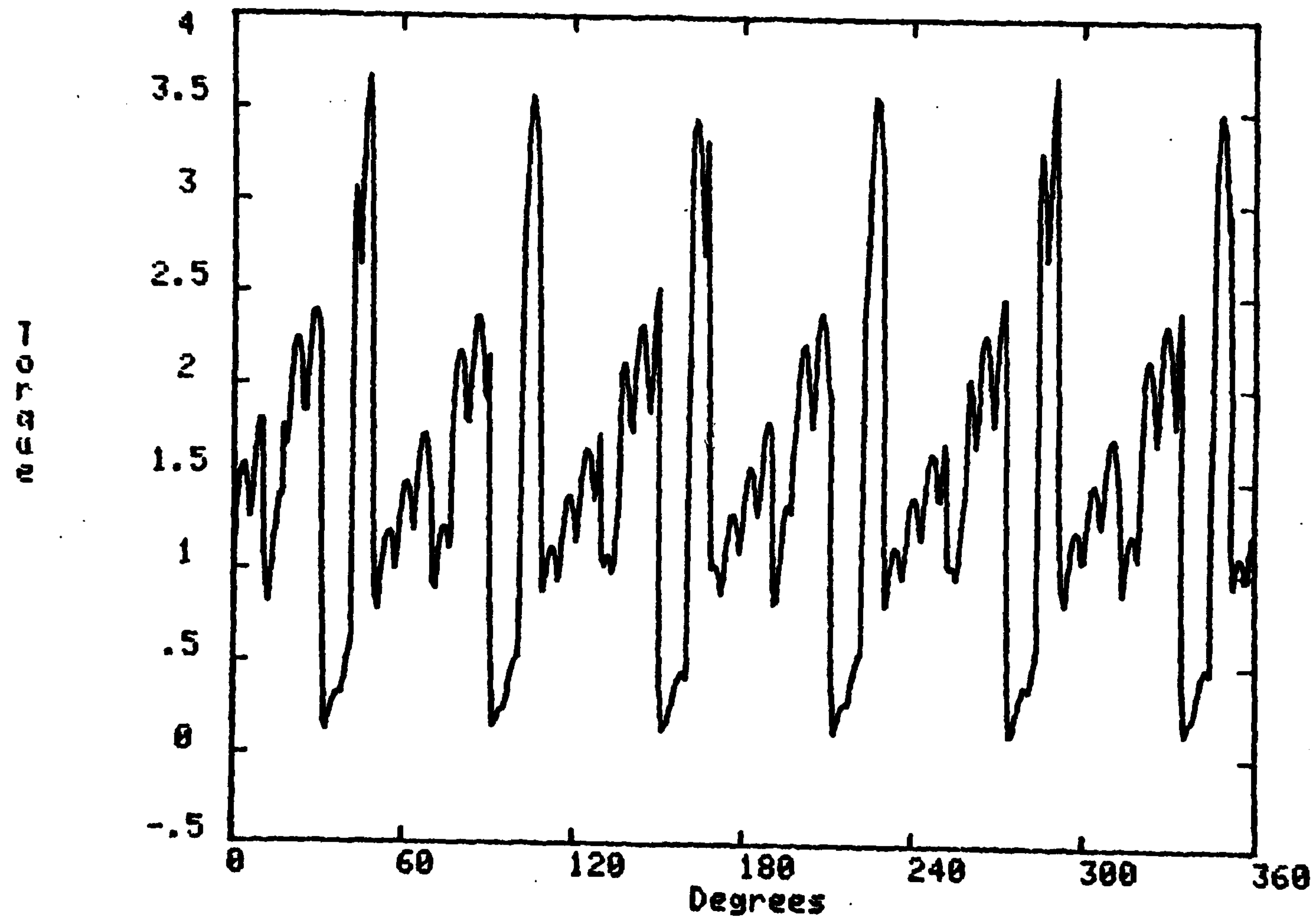


Figure 43. Developed Torque - 450 RPM, No Modulation

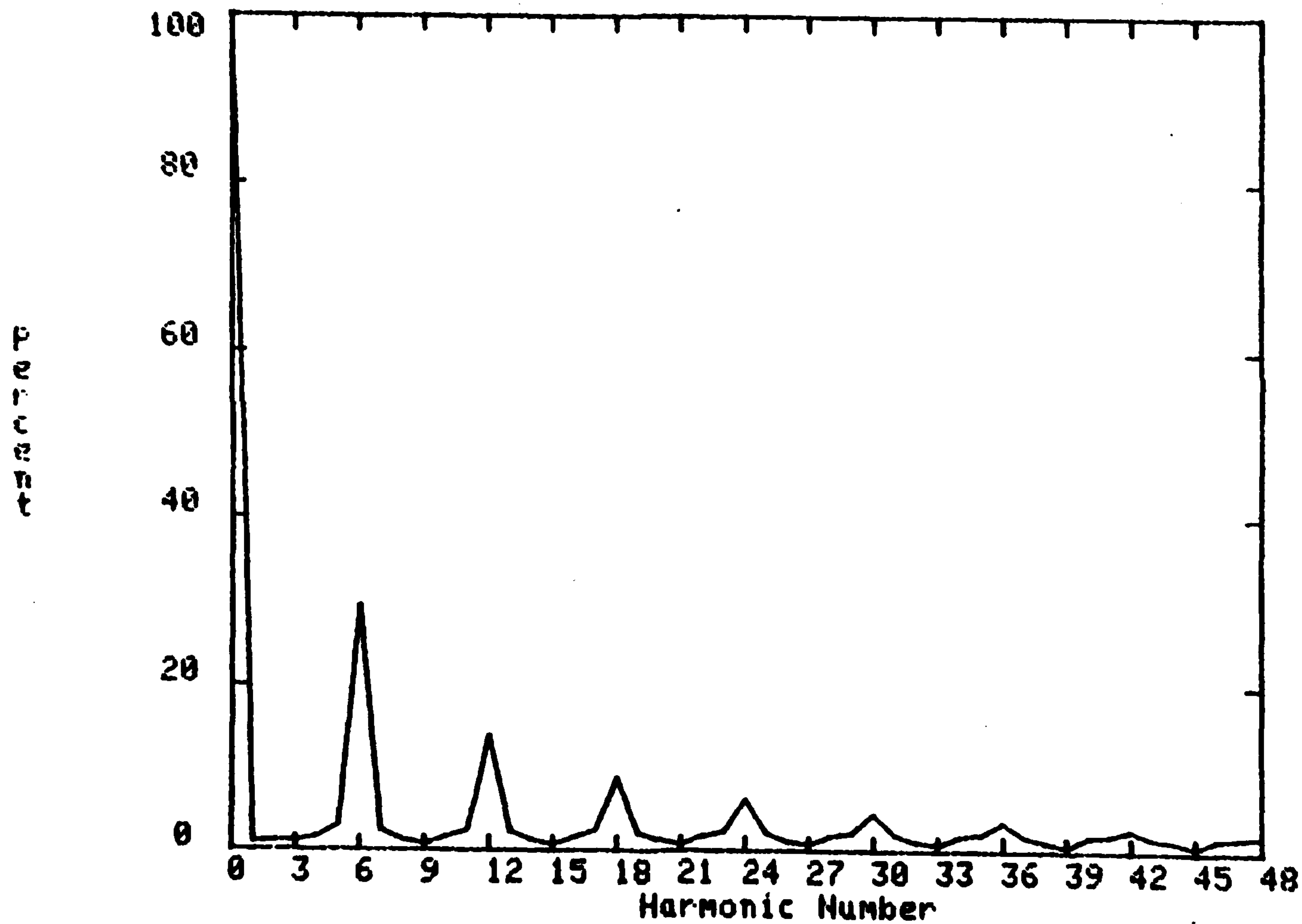


Figure 44. Fourier Spectrum of Developed Torque - 450 RPM, No Modulation

A plot of instantaneous developed torque is given by Figure 47. The Fourier spectrum of developed torque is shown by Figure 48.

2.3.3 Elimination of Sixth and Twelfth Harmonics of Torque. Both the sixth and twelfth harmonics of torque are eliminated by removing the fifth, seventh, eleventh, and thirteenth harmonics of phase current.

Input Data

Motor speed, n_m = 450 rpm
Motor frequency, f = 15 Hz
SCR firing angle, α_s = 49.78°
Specified average torque, T_{dav} = 1.5 N·m
Angles of modulation function, α_1 = 38.73°
α_2 = 42.13°
α_3 = 52.25°
α_4 = 61.93°
α_5 = 66.86°

Output Data

Average torque, T_{dav} = 1.507 N·m
RMS phase current I_{RMS} = 77.38 A

A plot of phase current i_1 is given by Figure 49. The associated Fourier spectrum of phase current is shown by Figure 50.

A plot of instantaneous developed torque is given by Figure 51. The Fourier spectrum of developed torque is shown by Figure 52.

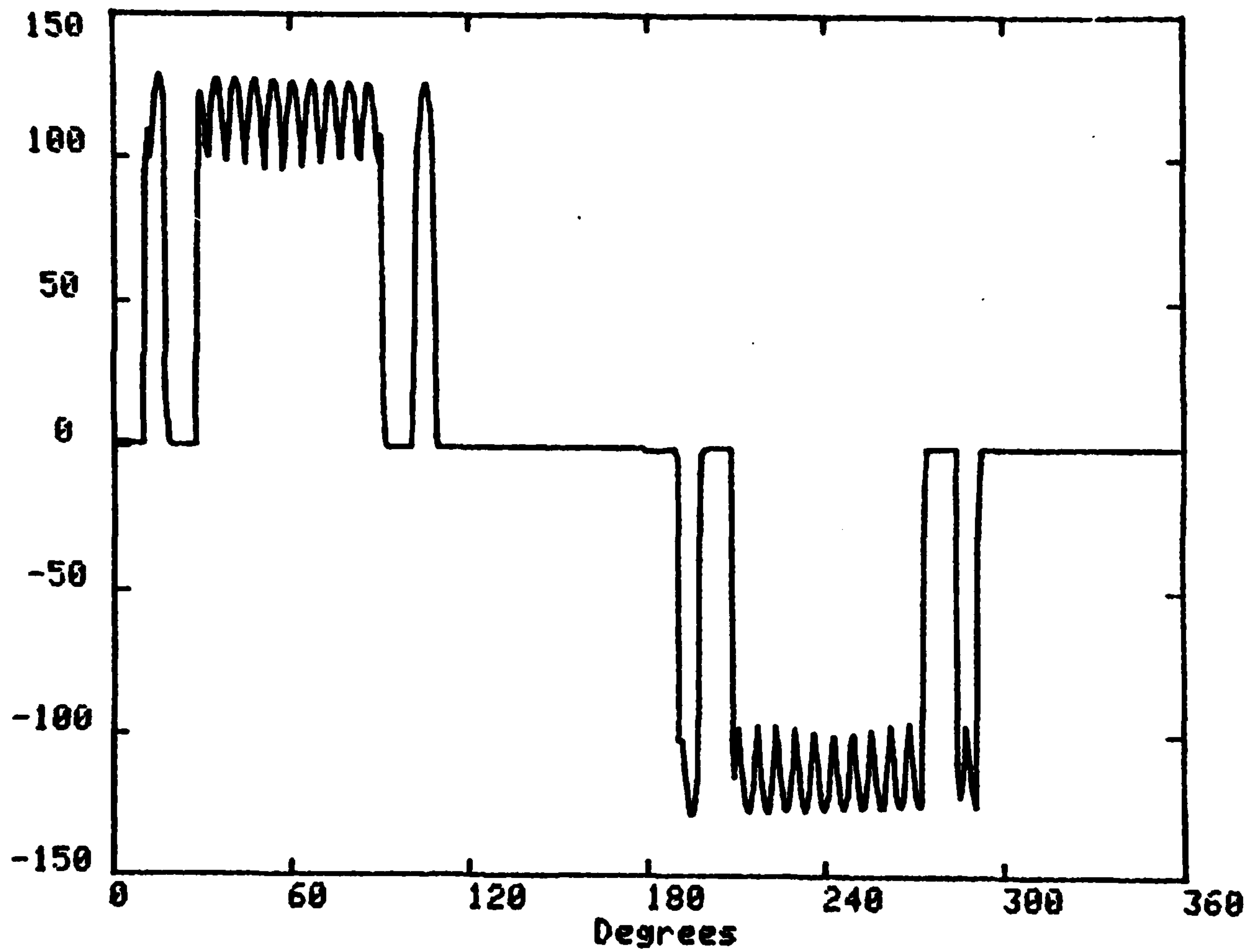


Figure 45. Phase Current - 450 RPM, Fifth and Seventh Harmonics Eliminated

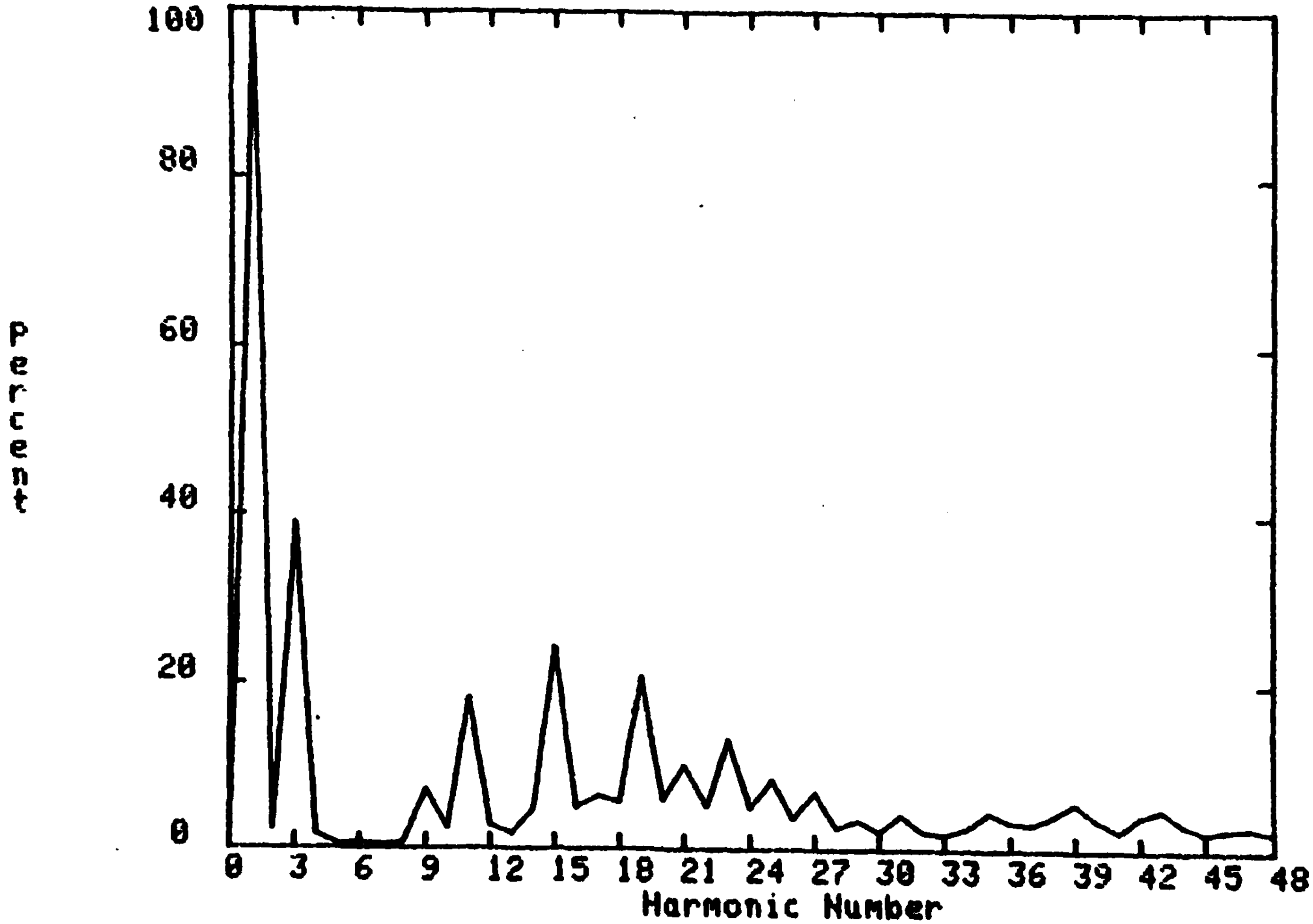


Figure 46. Fourier Spectrum of Phase Current - 450 RPM, Fifth and Seventh Harmonics Eliminated

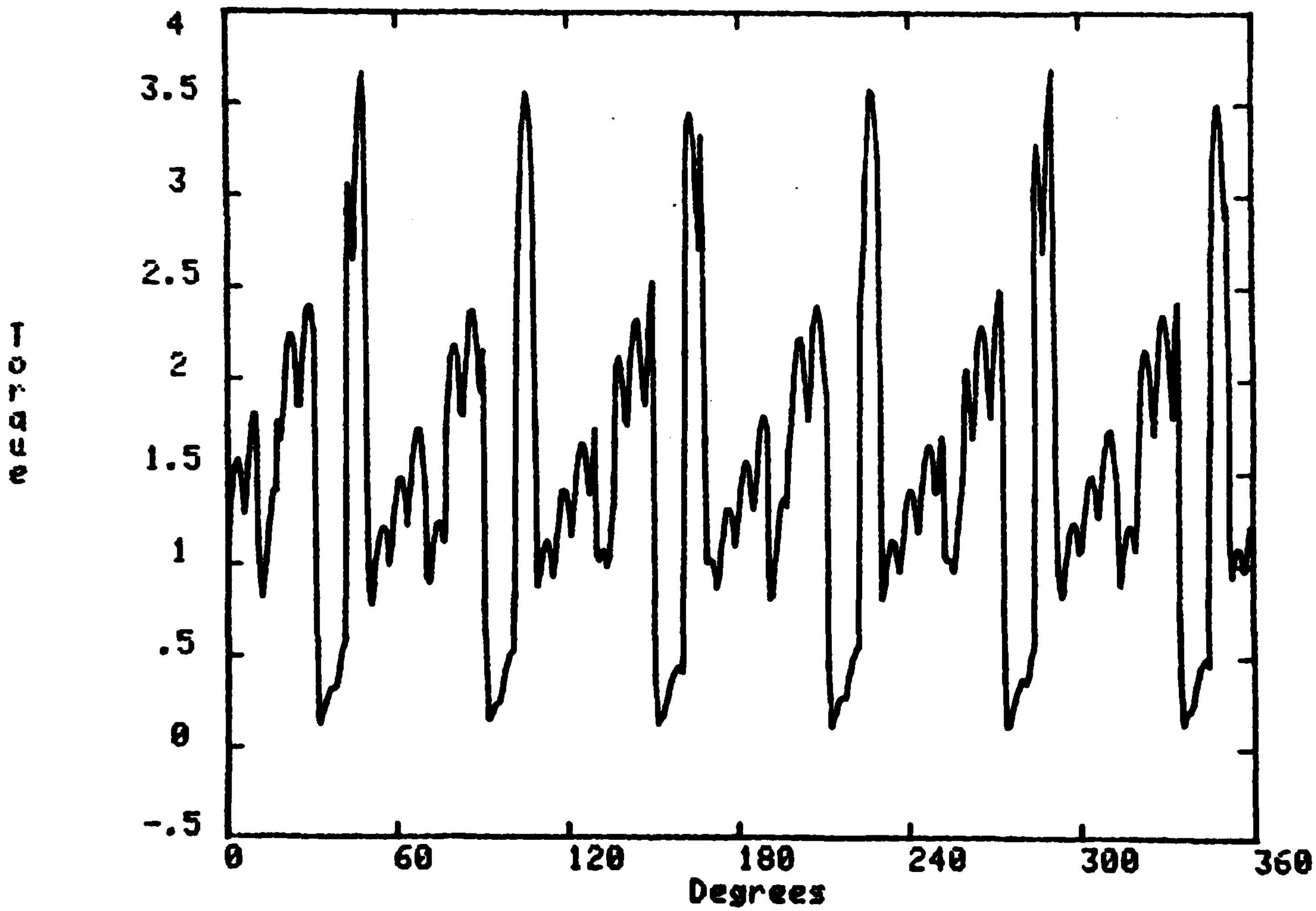


Figure 47. Developed Torque - 450 RPM, Sixth Harmonic Eliminated

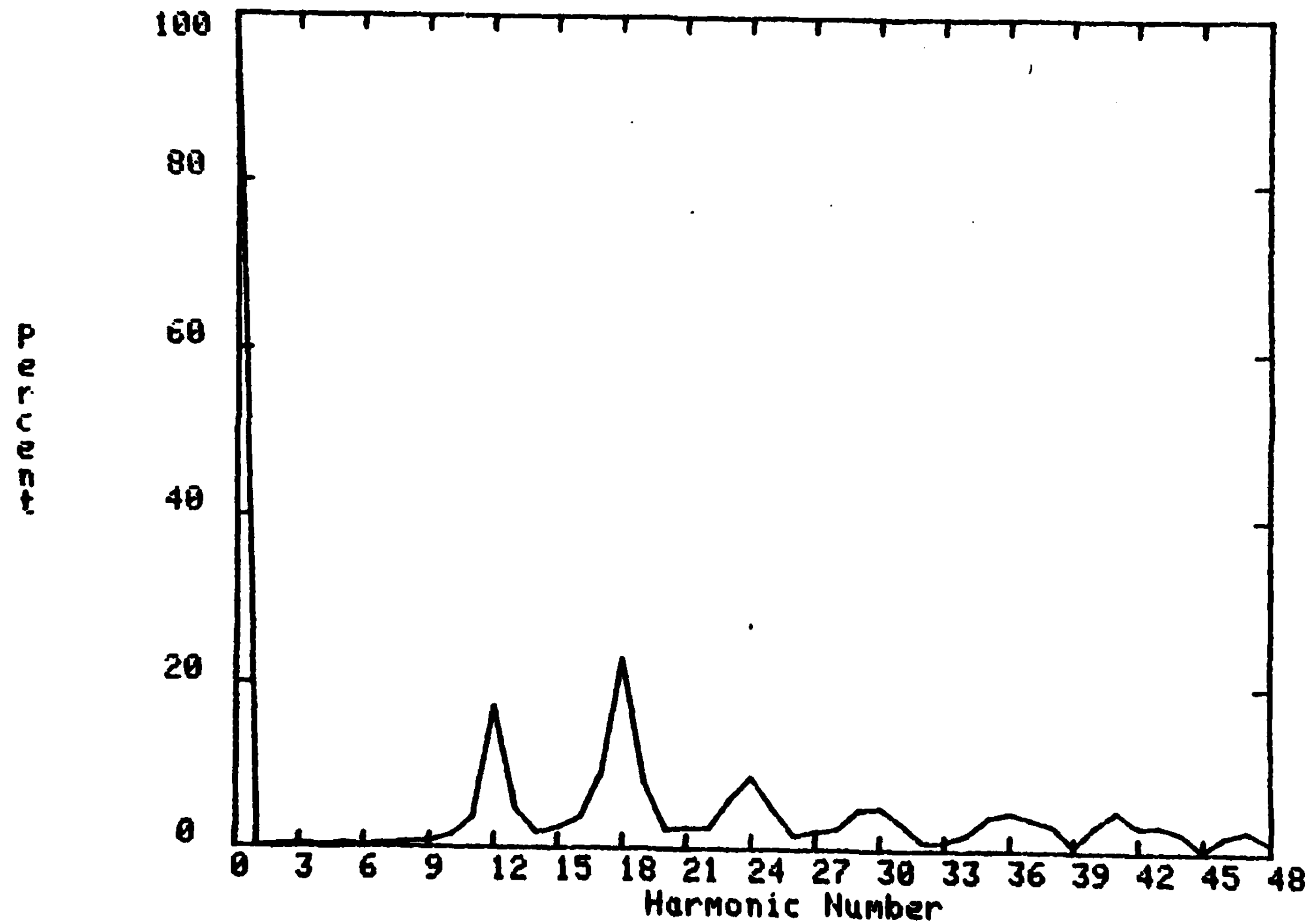


Figure 48. Fourier Spectrum of Developed Torque - 450 RPM, Sixth Harmonic Eliminated

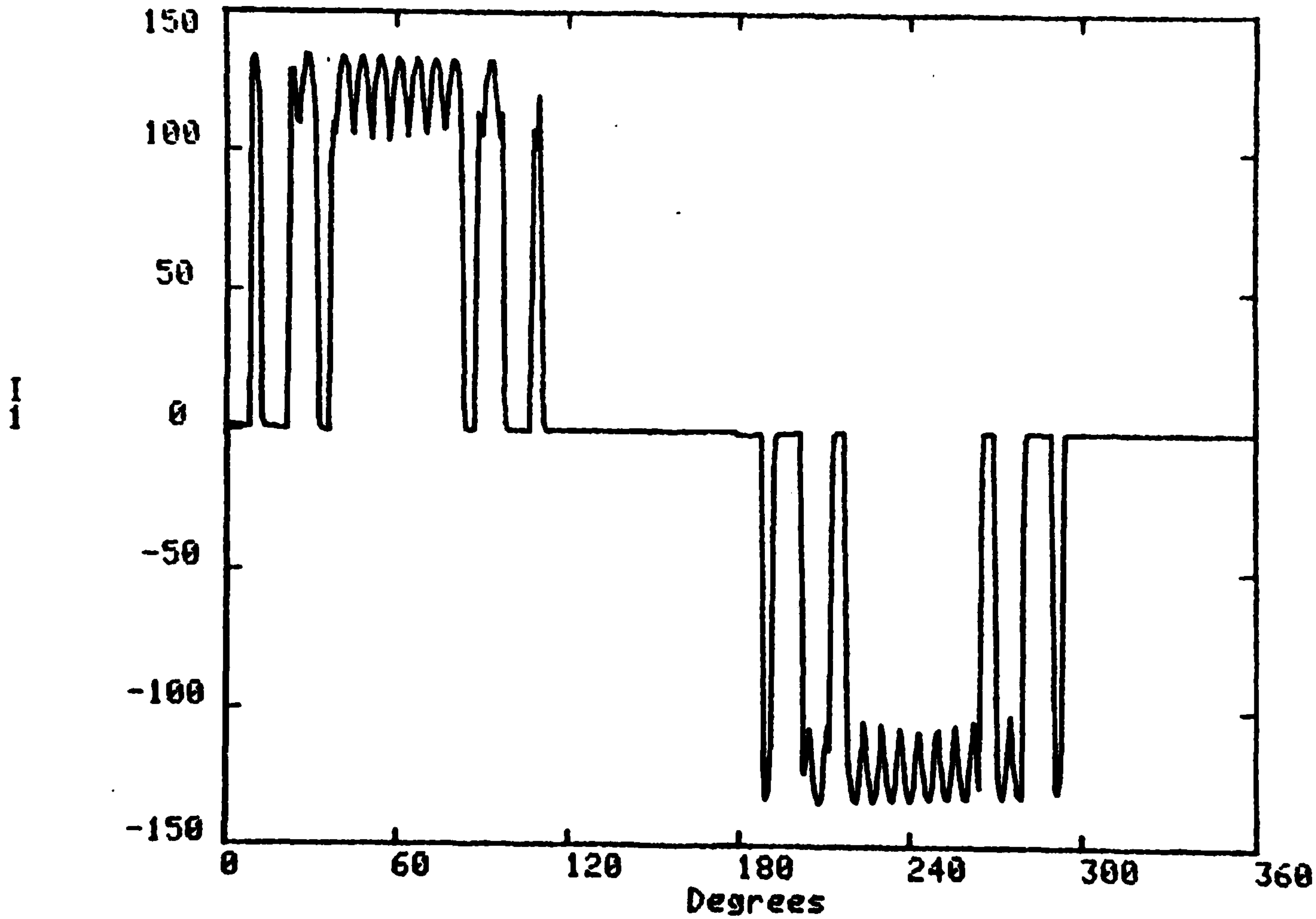


Figure 49. Phase Current - 450 RPM, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

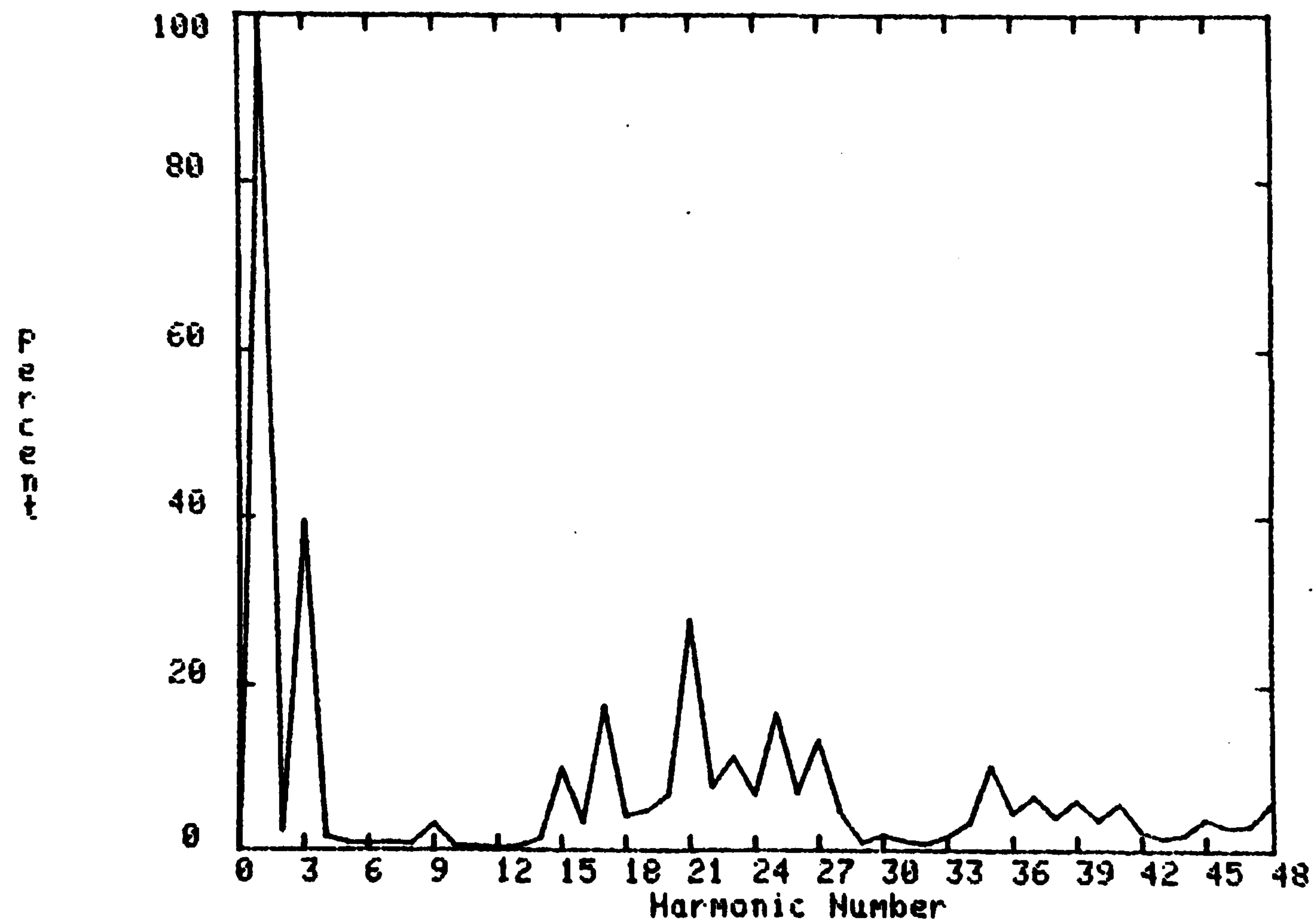


Figure 50. Fourier Spectrum of Phase Current - 450 RPM, Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

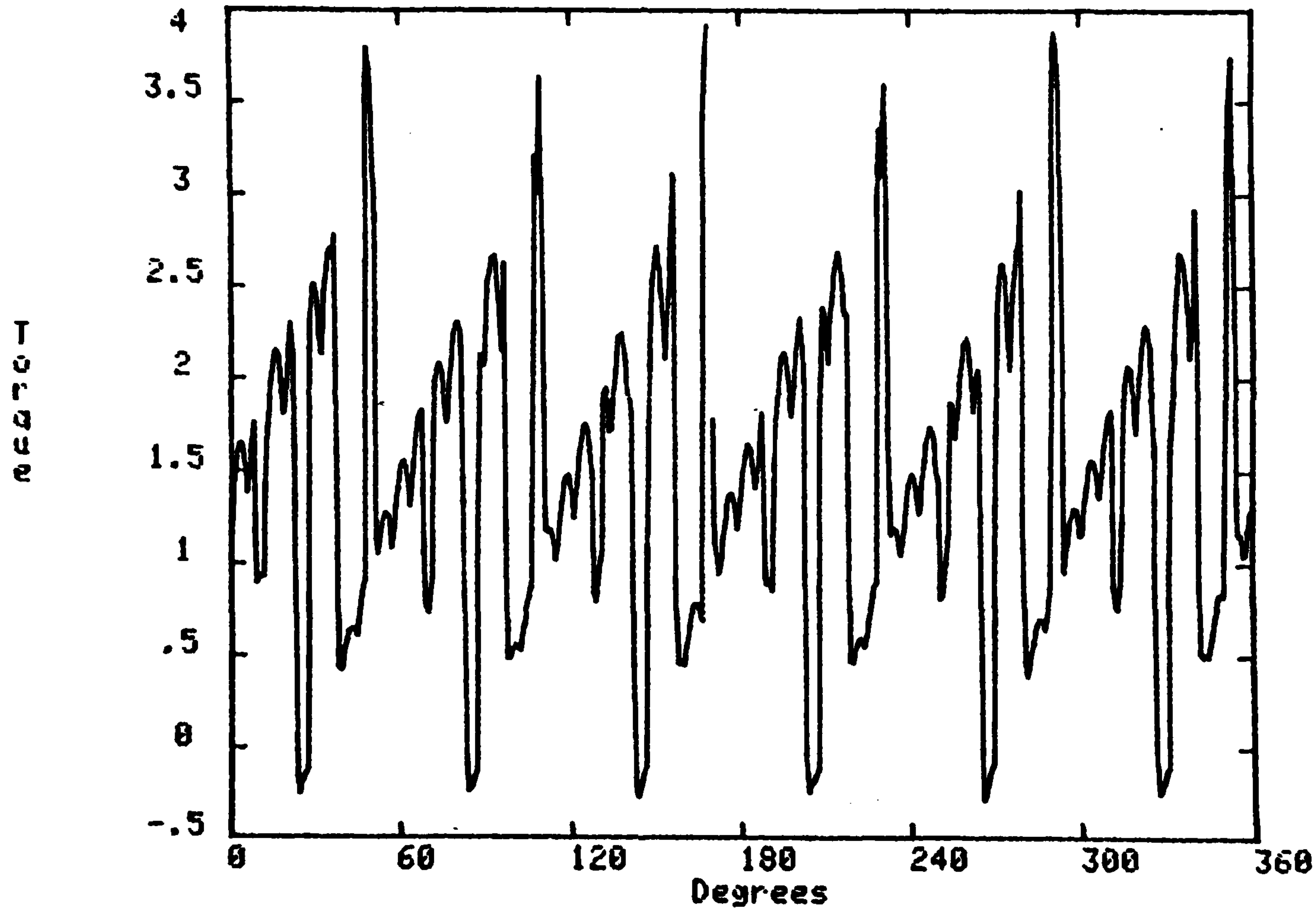


Figure 51. Developed Torque - 450 RPM, Sixth and Twelfth Harmonics Eliminated

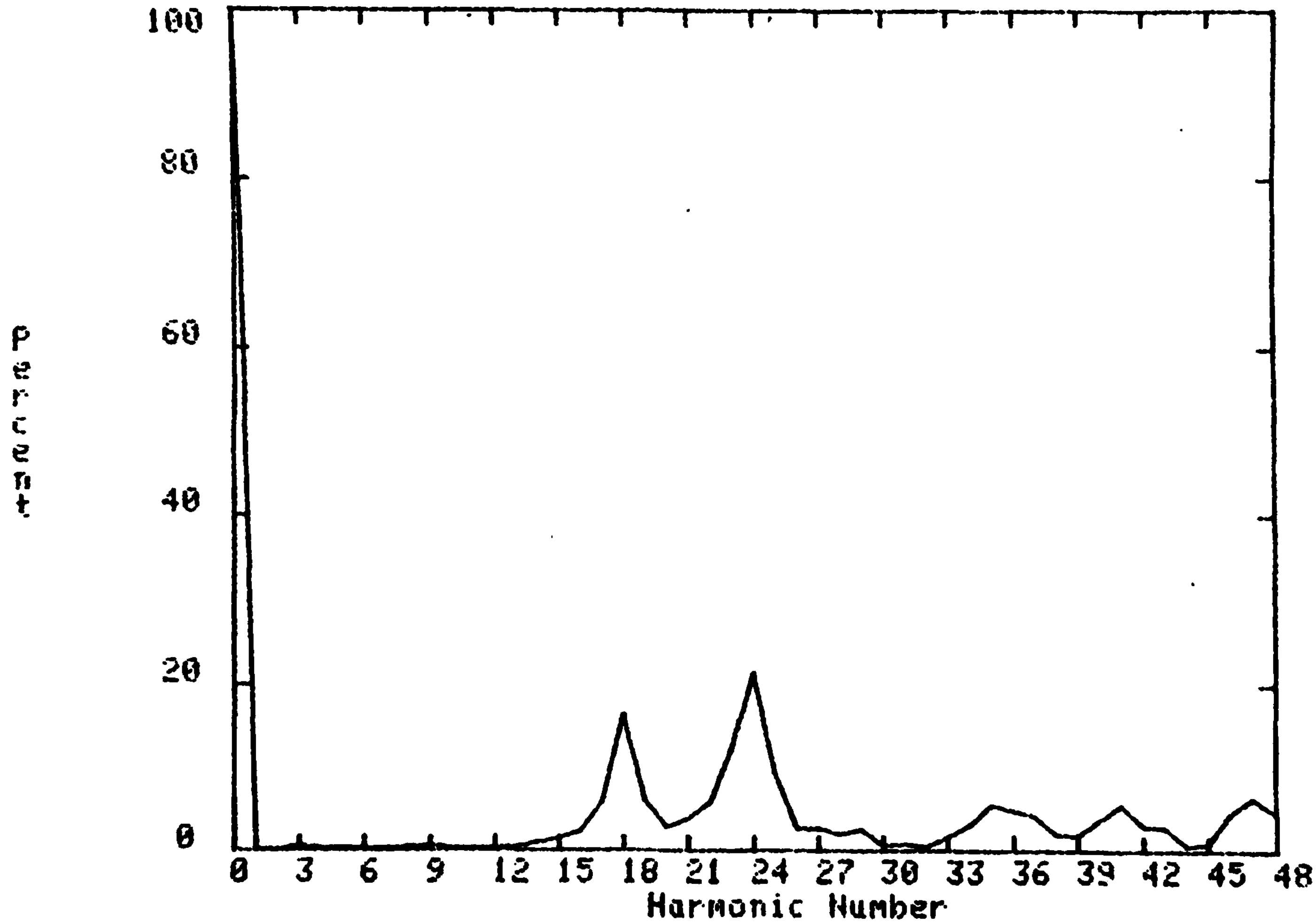


Figure 52. Fourier Spectrum of Developed Torque - 450 RPM, Sixth and Twelfth Harmonics Eliminated

SECTION IV

DISCUSSION

1.0 INCREASE IN OHMIC LOSSES

Since modulation of the wave form definitely redistributes the harmonic content of the phase current wave form, an examination of the change in winding ohmic losses is in order. For the 4500 rpm case it is noted that the RMS phase current is found to be as follows:

(a). $I_{RMS} = 72.77$ A for no modulation

(b). $I_{RMS} = 74.24$ A for fifth and seventh harmonic eliminated

(c). $I_{RMS} = 74.59$ A for fifth, seventh, eleventh, and thirteenth harmonic eliminated

The increase in ohmic losses assuming a constant winding resistance is given by the ratio of RMS current for a changed condition to the RMS value of current for the unmodulated wave form. It is concluded that the ohmic losses are increased by 2.1% for the case of elimination of fifth and seventh harmonics. The ohmic losses are increased by 2.5% for the case of elimination of the fifth, seventh, eleventh, and thirteenth harmonics.

Calculated data for the 2250 rpm and 450 rpm cases show increases of 7 to 15%; however, there is no theoretical basis for greater increase in these cases. The reason for this apparent larger increase for the lower speed cases is a numerical problem due to storing and calculating the RMS value based on a limited number of data points (2000) for transmittal to the plot routine. As a result, the high frequency ripple currents are sampled at irregular intervals giving rise to inaccuracy in calculation of the RMS value. For the case of 4500 rpm, the data point limit allows storage of several points during each ripple cycle, thus, the resulting RMS values are considerably more accurate than for the other two speed cases.

2.0 CYCLOCONVERTER MODE CHANGE

As discussed and illustrated by Figure 2, a neutral connection is necessary to successfully eliminate selected phase current harmonics. Since harmonic elimination is only necessary at speeds near zero, and since an ohmic loss penalty is incurred as a result, it might be desireable to not operate with PWM control over the full range of speed. The Harmonic Eliminator Switch in the neutral line of Figure 2 could be opened for higher speeds and the cycloconverter mode of operation changed to phase-control from synchronous envelope.

3.0 EXTENSION TO HIGHER HARMONIC ELIMINATION

For specific illustration, this study has only considered elimination of torque harmonics through the twelfth. As motor speed approaches zero, it may be necessary to eliminate even higher torque harmonics. Theoretically, there is no limitation to be encountered.

SECTION V

CONCLUSIONS AND RECOMMENDATIONS

The work performed by this study shows that systematic reduction of harmonic torque pulsations in brushless DC drives is feasible by use of selective current harmonic elimination. Further, there appears to be only a small ohmic loss penalty.

Further theoretical study should be made to properly assess quantitatively the effect of increased motor leakage inductance and of interphase reactors on "reintroducing" the harmonics eliminated from the phase voltage due to extending the commutation overlap.

Also, a hardware realization of a midpoint cycloconverter brushless DC drive using the control principles formulated in this study should be made to verify the results.

SECTION VI

REFERENCES

[1]. E. Ohno, T. Kishimoto, and M. Akamatsu, "The Thyristor Commutatorless Motor", IEEE Trans. Mag., Vol. MAG-3, Sept. 1967, pp. 236-240.

[2]. T. Tsachiya, "Basic Characteristics of Cycloconverter-Type Commutatorless Motors", IEEE Trans. IGA, Vol. IGA-7, No. 4, July-August 1970, pp. 349-356.

[3]. N. Sato and V.V. Semenos, "Adjustable Speed Drive with a Brushless DC Motor", IEEE Trans. IGA, Vol. IGA-7, No. 4, July-Aug. 1971, pp. 539-543.

[4]. E.P. Cornell and D.W. Novotny, "Commutation by Armature Induced Voltage in Self-Controlled Synchronous Machines", IEEE Trans. PAS, Vol. PAS-93, 1974, pp. 760-766.

[5]. N. Sato, "A Brushless DC Motor with Armature Induced Voltage Commutation", IEEE Trans. PAS, Vol. PAS-91, July-Aug. 1972, pp. 1485-1492.

[6]. J.M.D. Murphy, Thyristor Control of AC Motors, (Pergamon Press, Oxford, 1973), pp. 140-149.

[7]. F.J. Bourbeau, "Synchronous Motor Railcar Propulsion", IEEE Trans. IAS, Vol. IA-13, No. 1, Jan-Feb. 1977, pp. 8-17.

[8]. T. Maeno and M. Kobata, "AC Commutatorless and Brushless Motor", IEEE Trans. PAS, Vol. PAS-91, July-Aug. 1972, pp. 1476-1484.

[9]. Y. Shrinryo, I. Hosono, and K. Syoji, "Commutatorless DC Drive for Steel Rolling Mill", IEEE-IGA Conference Record, 1977 Annual Meeting, pp. 263-271.

[10]. A.C. Williamson, N.A.H. Issa, and A.R.A.M. Makky, "Variable-Speed Inverter-Fed Synchronous Motor Employing Natural Commutation", Proc. IEE, Vol. 125, No. 2, Feb. 1978, pp. 118-120.

[11]. N.A. Demerdash, T.W. Nehl, and E. Maslowski, "Dynamic Modeling of Brushless DC Motors in Electric Propulsion and Electromechanical Actuation by Digital Techniques", IEEE-IAS Conference Record, 1980 Annual Meeting, Sept. 28 - Oct. 3, 1980, pp. 570-579.

[12]. B.R. Pelly, Thyristor Phase-Controlled Converters and Cycloconverters, (Wiley Interscience, New York, 1971) pp. 145-180.

[13]. F.G. Turnbull, "Selected Harmonic Reduction in Static DC-AC Inverters", IEEE Trans. Commun. Electron., Vol. 83, July 1964, pp. 374-378.

[14]. H.S. Patel and R.G. Hoft, "Generalized Techniques of Harmonic Elimination and Voltage Control in Thyristor Inverters: Part I - Harmonic Elimination", IEEE Trans. IA, Vol. IA-9, No. 3, May/June 1973, pp. 310-317.

[15]. H.S. Patel and R.G. Hoft "—— Part II - Voltage Control Techniques", IEEE Trans. IA, Vol. IA-10, No. 5, Sept/Oct. 1974, pp. 666-673.

APPENDIX A

HARMONICS OF PWM TIME FUNCTION

A modulation function $h(\omega t)$ was derived in Appendix B and used in Section II - 5.0 to form a modulated phase voltage. For example, $v_{an} = h(\omega t)\ v_d(t)$.

Harmonic content of a modulated voltage is of concern. The study is best understood by use of a typical example. Let $h(\omega t)$ contain all odd harmonics except the fifth and seventh, then

$$h(\omega t) = a_1 \cos \omega t + a_3 \cos 3\omega t + a_9 \cos 9\omega t + a_{11} \cos 11\omega t + \cdots + a_n \cos n\omega t + \cdots \quad (A.1)$$

Assume the $v_d(t)$ contains a DC term plus one high frequency harmonic, then

$$v_d(t) = b_0 + b_m \cos m\omega t \quad (A.2)$$

Forming v_{an} as the product of $h(\omega t)$ and $v_d(t)$ and simplifying yields

$$v_d(t) = c_1 \cos \omega t + c_3 \cos 3\omega t + c_9 \cos 9\omega t + c_{11} \cos 11\omega t + \cdots + d_{m+1} \cos(m+1)\omega t + d_{m-1} \cos(m-1)\omega t + d_{m+3} \cos(m+3)\omega t + d_{m-3} \cos(m-3)\omega t + \cdots \quad (A.3)$$

It is seen from (A.3) that through the cross product terms, the PWM expression may contain frequencies that were selectively eliminated from $h(\omega t)$. However, the coefficients $(d_{m+1}, d_{m-1}, \cdots)$ of (A.3) are products of a_n and b_m. Since a_n decreases as $1/n$, if m is large, then the coefficients of the low frequency components of $h(\omega t)$ resulting cross product terms will be negligibly small. It is concluded that the frequency of the AC source must be large compared with the motor frequency if the PWM function is to not contain the harmonics selectively eliminated from the modulating function $h(\omega t)$. But, at near zero speed, the source frequency is large compared with the motor frequency.

APPENDIX B

MODULATION FUNCTION PROGRAMS

1.0 INITIAL SOLUTION

As first procedure, a candidate or initial modulation function, $h(\omega t)$, is found to serve as a basis for an optimization search to determine the final modulation function.

1.1 Theory of Harmonic Elimination

Extending Patel and Hoft's pulse width modulation technique [14] to include a slack variable, it is possible to find a modulation function that has selected harmonics elimination while existing on the quarter wave interval from 30^{o} to 90^{o}.

The Fourier series representation of the modulation function assuming odd quarter wave symmetry is

$$f(\omega t) = \sum_{1}^{\infty} a_n \sin(n\omega t) , \quad \text{for odd } n$$

where

$$a_n = \int_{\alpha_1}^{\alpha_2} \sin(n\omega t)d\omega t + \int_{\alpha_3}^{\alpha_4} \sin(n\omega t)d\omega t + \cdots \int_{\alpha_{M-1}}^{\alpha_M} \sin(n\omega t)d\omega t, \quad \text{(B.1)}$$

$$30^{o} < \alpha_1 < \alpha_2 < \cdots < \alpha_M < 90^{o}$$

This reduces to

$$a_n = \frac{4}{n\pi} \sum_{k=1}^{M} (-1)^{k+1}\cos(n\alpha_k) \quad \text{(B.2)}$$

for M both odd and even. The number of harmonics eliminated is in general the same as the number of switching transitions per quarter cycle. For M greater than 2 the equations cannot be solved in closed form, so numerical methods are needed to obtain solutions. The following normalized system of equations is to be used to obtain solutions:

$$\begin{aligned} f_{n1} &= \sum_{1}^{M} (-1)^{k+1}\cos(n_1\alpha_k) = 0 \\ f_{n2} &= \sum_{1}^{M} (-1)^{k+1}\cos(n_2\alpha_k) = 0 \quad \text{(B.3)} \\ &\vdots \\ f_{nM} &= \sum_{1}^{M} (-1)^{k+1}\cos(n_M\alpha_k) = 0 \end{aligned}$$

where $n_1, n_2, \cdots n_M$ are the harmonics to be eliminated.

Many of the solutions of (B.3) are meaningless or impractical, so it is necessary to search by brute force using a loose convergence criterion for candidate solutions, and then to use a numerical method such as Newton-Raphson to obtain a refined solution.

The basic method of selecting candidate modulation function for further optimization is illustrated by the flow chart of Figure B.1. A candidate set is one that meets the following criteria:

$$f_{squar} < f_{min}$$

$$f_{squar} = f_{n1}^2 + f_{n2}^2 \cdots f_{nM}^2$$

If for the candidate modulation function, $30^\circ < a_1 < a_2 < \cdots < a_M < 90^\circ$, then the solution is considered for optimization. Otherwise, the solution is discarded.

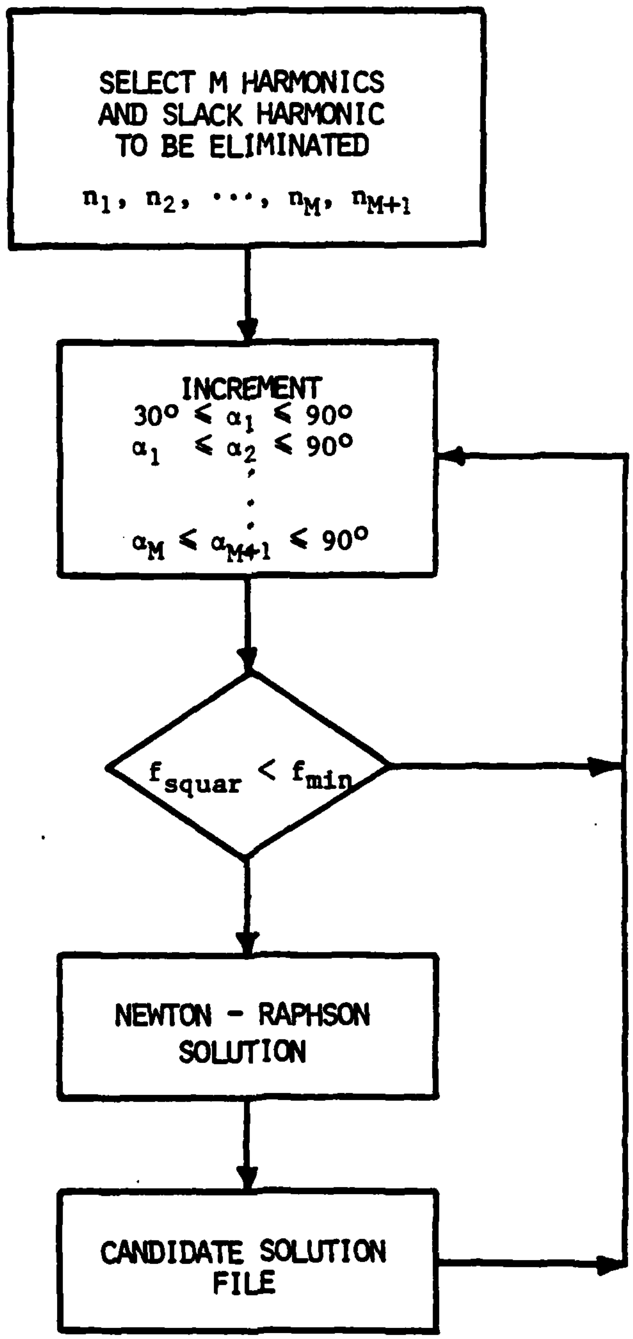


Figure B.1. Flow Chart for Selection of Candidate Modulation Functions

1.2 Initial Solution Program

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                    c
c              main program                                          c
c                                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       n       - number of harmonics to be eliminated
c                 this includes the slack variable
c       inc     - this is the amount that each alpha
c                 will be incremented by during the
c                 infinite search
c       fmin    - newton rapson method is used to
c                 search for a solution if the sum
c                 of squares is less than fmin
c       r(i)    - these are the harmonics to be reduced
c       amin    - this is the minimum value of alpha
c                 for the infinite search
c       iwr     - not used
c       del     - not used
        dimension a(10),r(10),f(10)
        common pi,conv,iwr,del
        real inc
        call sopen(0.0)
        pi=3.141592653
        k1min=0
        k2min=0
        k3min=0
        k4min=0
        k5min=0
        k6min=0
        k1=1
        k2=1
        k3=1
        k4=1
        k5=1
        k6=1
        conv=180.0/pi
        read(9,*) n
        write(6,*) ' n=',n
        read(9,*) inc
        write(6,*) ' inc=',inc
        read(9,*) fmin
        write(6,*) ' fmin=',fmin
        read(9,*) (r(i),i=1,7)
        write(6,*) ' harmonics=',(r(i),i=1,n)
        read(9,*) amin
        write(6,*) ' amin=',amin
        read(9,*) iwr
        write(6,*) ' iwr=',iwr
```

```
      max=int( (90.0-amin)/inc )
      k1max=int( (90.0-amin)/inc )
      k2max=max
      k3max=max
      k4max=max
      k5max=max
      k6max=max
      isep=int( 1.0/inc ) +1
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     the following are essentially do loops but were
c     necessary to get past a problem with the compiler
c
      k1min=1
100   k1=k1min-1
105   k1=k1+1
      if( n .eq. 1 ) go to 1000
      k2min=k1+isep
200   k2=k2min-1
205   k2=k2+1
      if( n .eq. 2 ) go to 1000
      k3min=k2+isep
300   k3=k3min-1
305   k3=k3+1
      if( n .eq. 3 ) go to 1000
      k4min=k3+isep
400   k4=k4min-1
405   k4=k4+1
      if( n .eq. 4 ) go to 1000
      k5min=k4+isep
500   k5=k5min-1
505   k5=k5+1
      if( n .eq. 5 ) go to 1000
      k6min=k5+isep
600   k6=k6min-1
606   k6=k6+1
      if( n .eq. 6 ) go to 1000
1000  a(1)=( float(k1)*inc+amin )/conv
      a(2)=( float(k2)*inc+amin )/conv
      a(3)=( float(k3)*inc+amin )/conv
      a(4)=( float(k4)*inc+amin )/conv
      a(5)=( float(k5)*inc+amin )/conv
      a(6)=( float(k6)*inc+amin )/conv
      call ff( a,r,f,n )
      call square( f,fsquar,n )
      if( fsquar .lt. fmin ) call look( a,r,f,fsquar,n )
      if( k6 .lt. k6max .and. n .ge. 6 ) go to 606
      if( k5 .lt. k5max .and. n .ge. 5 ) go to 505
      if( k4 .lt. k4max .and. n .ge. 4 ) go to 405
      if( k3 .lt. k3max .and. n .ge. 3 ) go to 305
```

```
      if( k2 .lt. k2max .and. n .ge. 2 ) go to 205
      if( k1 .lt. k1max .and. n .ge. 1 ) go to 105
      write(6,3100)
      write(6,3000) ( a(i)*conv,i=1,n )
      call sclose(0.0)
3000  format( 10(e15.7,5x) )
3100  format(' a=')
3200  format(' harmonics=')
      stop
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                    c
c              subroutine a                                          c
c                                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine fa( a,da,n )
        dimension a(10),da(10)
        common pi,conv,iwr,del

        do 200 i=1,n
          a(i)=a(i)-da(i)
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine  close                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sclose(dummy)
        close(1)
        close(2)
        close(3)
        close(4)
        close(7)
        close(8)
        close(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine df                                          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine fdf( a,r,f,df,n )
        dimension a(10),r(10),f(10),df(10,10)
        common pi,conv,iwr,del

        do 200 i=1,n
          sign=1.0
                do 100 j=1,n
                  df(i,j)=-sign*r(i)*sin( r(i)*a(j) )
                  sign=-sign
100             continue
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine f                                           c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine ff( a,r,f,n )
        dimension a(10),r(10),f(10)
        common pi,conv,iwr,del
        do 200 i=1,n
          sign=1.0
          f(i)=0.0
                do 100 j=1,n
                  f(i)=f(i)+sign*cos( r(i)*a(j) )
                  sign=-sign
100             continue
200     continue
        return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                       c
c               subroutine look                                         c
c                                                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine look( a,r,f,fsquar,n )
        dimension da(10),r(10),a(10),f(10),df(10,10),dfi(10,10)
        common pi,conv,iwr,del
        write(6,3050)
        write(6,3050)
        write(6,3100)
        write(6,3000) ( a(i)*conv,i=1,n )
        write(6,3150)
        write(6,3000) ( f(i),i=1,n )
        write(6,3200)
        write(6,3000) fsquar
        kount=30
        fmin=0.01
        delta=0.001


        do 2500 loop=1,kount
        call fdf( a,r,f,df,n )
        call matv( df,n,det,dfi )
        call mult1( f,dfi,da,n,n )
        call fa( a,da,n )
        call ff( a,r,f,n )
        call square( f,fsquar,n )
        if( loop .gt. 5 .and. fsquar .lt. fmin ) go to 2900
2500    continue
2900    write(6,3050)
        write(6,3100)
        write(6,3000) ( a(i)*conv,i=1,n )
        write(6,3150)
        write(6,3000) ( f(i),i=1,n )
        write(6,3200)
        write(6,3000) fsquar
3000    format( 10(e10.4,5x) )
3050    format( ' ' )
3100    format( ' a=' )
3150    format( ' f=' )
3200    format( ' fsquar=' )
4000    format('test',i3)
5000    return
        end
```

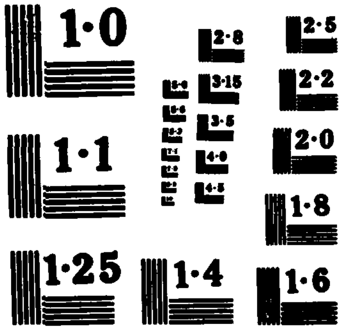
AD-A144 216 HARMONIC CONTROL TO REDUCE TORQUE PULSATIONS IN BRUSHLESS DC MOTOR DRIVES(U) KENTUCKY UNIV RESEARCH FOUNDATION LEXINGTON J J CATHEY MAR 84 2/2

UNCLASSIFIED AFOSR-TR-84-0607 AFOSR-83-0189 F/G 20/3 NL

END

FILMED

DTIC


1·0
2·8
2·5
3·15
2·2
3·5
1·1
2·0
4·0
4·5
1·8
1·25
1·4
1·6

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C             SUBROUTINE MATV                                           C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE MATV(A,N,DET,B)
        DIMENSION INDEX(20,2),IPVOT(20),A(10,10),B(10,10),PIVOT(20)
        ONE=1.0
        ZERO=0.0

C
        DO 10 I=1,N

          DO 5 J=1,N
            B(I,J)=A(I,J)
5         CONTINUE

10      CONTINUE
        DET=ONE
        DO 15 J=1,N
15       IPVOT(J)=0

        DO 75 I=1,N
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                              C
C               FOLLOWING 15 STATEMENTS                        C
C                FOR SEARCH FOR PIVOT ELEMENTS                 C
C                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        T=ZERO
        DO 40 J=1,N
          IF(IPVOT(J)-1) 20,40,20
20        DO 35 K=1,N
            IF(IPVOT(K)-1) 25,35,100
25          CONTINUE
          D1=ABS(T)
          D2=ABS( B(J,K) )
          IF(D1-D2)30,35,35
30          IROW=J
            ICOL=K
            T=B(J,K)
35        CONTINUE

40      CONTINUE
```

```
        IPVOT(ICOL)=IPVOT(ICOL)+1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
C               FOLLOWING 10 STATEMENTS PUT                      C
C               PIVOT ELEMENT ON DIAGONAL                        C
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        IF(IROW-ICOL) 45,55,45
45        DET=-DET
        DO 50 L=1,N
          T=B(IROW,L)
          B(IROW,L)=B(ICOL,L)
50        B(ICOL,L)=T
55      INDEX(I,1)=IROW
        INDEX(I,2)=ICOL
        PIVOT(I)=B(ICOL,ICOL)
        DET=DET*PIVOT(I)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                         C
C          FOLLOWING 3 STATEMENTS TO DIVIDE PIVOT       C
C          ROW BY PIVOT ELEMENT                           C
C                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        B(ICOL,ICOL)=ONE
        DO 60 L=1,N
60      B(ICOL,L)=B(ICOL,L)/PIVOT(I)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                         C
C          FOLLOWING 7 STATEMENTS TO REDUCE               C
C          NON-PIVOT ROWS                                 C
C                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        DO 75 LI=1,N
          IF(LI-ICOL) 65,75,65
65        T=B(LI,ICOL)
          B(LI,ICOL)=ZERO
            DO 70 L=1,N
70          B(LI,L)=B(LI,L)-B(ICOL,L)*T
75      CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C              FOLLOWING 11 STATEMENTS TO                               C
C              INTERCHANGE COLUMNS                                      C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      DO 95 I=1,N
        L=N-I+1
        IF(INDEX(L,1)-INDEX(L,2) ) 85,95,85
85      JROW=INDEX(L,1)
        JCOL=INDEX(L,2)
          DO 90 K=1,N
              T=B(K,JROW)
              B(K,JROW)=B(K,JCOL)
              B(K,JCOL)=T
90        CONTINUE
95    CONTINUE
100   RETURN
      END
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine multiply                                    c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mult1( a,b,c,i0,k0 )
        dimension a(10),b(10,10),c(10)
        do 200 i=1,i0
          c(i)=0.0
          do 100 k=1,k0
            c(i)=c(i)+b(i,k)*a(k)
100       continue
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine  open                                       c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sopen(dummy)
        open(unit=9,file="input")
        rewind(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine square                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine square( f,fsquar,n )
        dimension f(10)
        fsquar=0.0
        do 200 i=1,n
          fsquar=fsquar+f(i)*f(i)
200     continue
        return
        end
```

2.0 OPTIMIZATION OF MODULATION FUNCTION

2.1 Discussion of Procedure

A candidate solution for the modulation function is selected from the output of the Initial Solution Program for optimization. The slack harmonic of the initial solution is changed in small increments and fed to a Newton-Raphson routine to find the new set of solutions. A continuous set of α's results from this procedure having "nice" properties. First, the new set of solutions is well behaved in that meaningless sets of solutions are not encountered and convergence is quick. Second, the procedure gives a "visual" picture of the behavior of the solutions in relation to the slack variable.

Since infinite number of modulation functions exist, some method is needed to select the "best" solution. The criteria used in this study was minimizing the sum of the squares of the magnitudes of the next set of current harmonics to be eliminated beyond those of present concern. For example, if the fifth and seventh harmonics were being eliminated from h(t), then the procedure is to minimize the sum of the square of the magnitudes of the eleventh and thirteenth harmonics.

A flow chart illustrating optimization of the modulation function is given by Figure B.2.

Figure B.3 illustrates the form of the modulation function for the case of elimination of fifth and seventh harmonics and for the case of elimination of fifth, seventh, eleventh, and thirteenth harmonics.

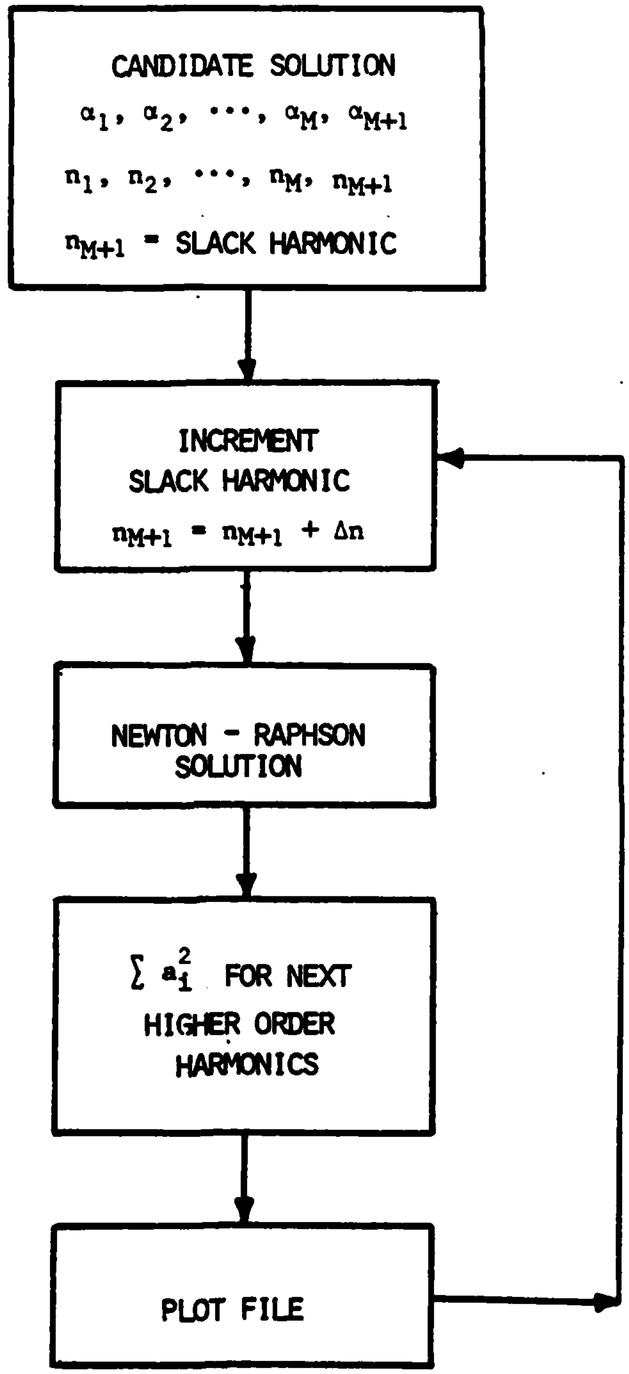


Figure B.2. Flow Chart for Optimizing Modulation Function

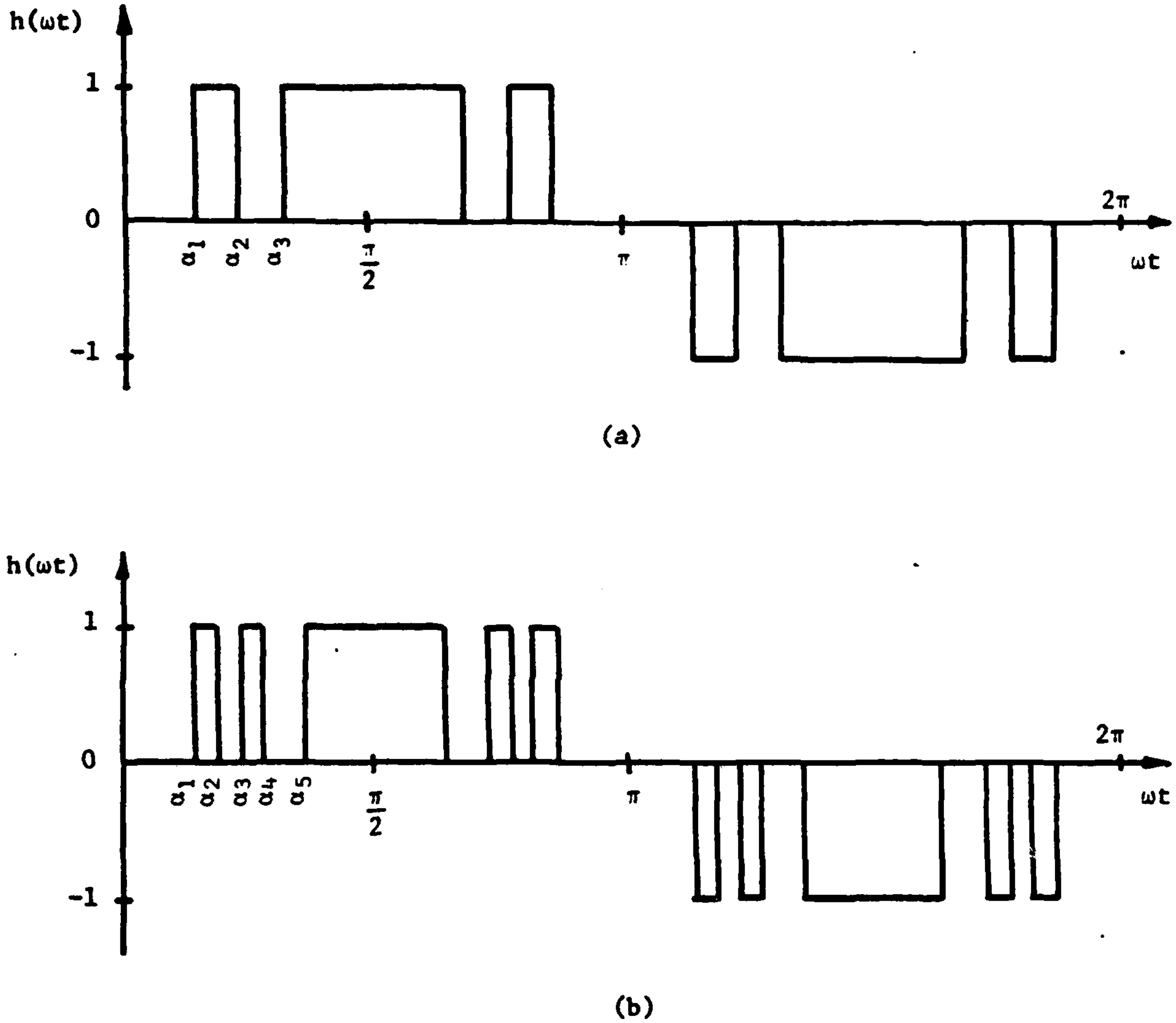


Figure B.3. Form of Modulation Functions.
(a) Fifth and Seventh Harmonics Eliminated
(b) Fifth, Seventh, Eleventh, and Thirteenth Harmonics Eliminated

2.2 Modulation Function Optimization Program

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c              main program                                            c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       n       - number of harmonics to be eliminated
c                 n+1 harmonics are eliminated
c       r(i)    - harmonics to be eliminated
c       a(i)    - these are the angles from the infinite
c                 search .i.e. this is the seed
c       rmax    - upper limit of slack variable
c       rmin    - lower limit of slack variable
c       points  - number of solutions or points to be saved
c       msq     - number of harmonics to be squared and sumed
c       tr(i)   - harmonics to be summed and squared excluding tr(1)
c       tr(1)   - any misc. harmonic that the user wishes to view
c       incre   - used for increasing or decreasing the slack variable
c                 incre=1 increment the slack variable from min to max
c                 incre=0 decrement the slack variable from max to min
c       hmag    - this is the sum of sqaures
        dimension a(10),r(10),f(10),h(10),tr(10)
        common pi,conv
        call sopen(0.0)
        pi=3.141592653
        conv=180.0/pi
        read(9,*) n
        write(6,*) ' n=',n
        read(9,*) (r(i),i=1,5)
        write(6,*) ' harmonics=',(r(i),i=1,n)
        read(9,*) (a(i),i=1,5)
        write(6,*) ' alphas=',(a(i),i=1,n)
        read(9,*) rmax
        write(6,*) ' rmax=',rmax
        read(9,*) rmin
        write(6,*) ' rmin=',rmin
        read(9,*) points
        write(6,*) ' points=',points
        read(9,*) msq
        write(6,*) ' msq=',msq
        read(9,*) (tr(i),i=1,5)
        write(6,*) ' hsquar harmonics=',(tr(i),i=2,msq+1)
        read(9,*) incre
        write(6,*) ' incre=',incre

        rinc=( rmax-rmin )/points
        maxr=int( rmax/rinc )
        minr=int( rmin/rinc )
        irmax=maxr-minr
```

```
      do 50 i=1,5
       a(i)=a(i)/conv
50    continue
      do 100 ir=minr,maxr
       if( incre .eq. 1 ) r(n+1)=float(ir)*rinc
       if( incre .eq. 0 ) r(n+1)=float( maxr+minr-ir )*rinc
       call ff( a,r,f,n+1 )
       call square( f,fsquar,n+1 )
       call look( a,r,f,fsquar,n+1 )
       call shar( a,tr,h,n+1,msq+1 )
       call shsq( h,hsquar,msq )
      hmag=sqrt( hsquar )
       write(1,*) ( a(i)*conv,i=1,n+1)
       write(2,1000) h(1)
       write(3,1000) h(2)
       write(4,1000) h(3)
       write(7,1000) r(n+1)
       write(8,1000) hmag
100   continue
      call sclose(0.0)
1000  format( e10.4 )
3000  format( 10(e15.7,5x) )
3100  format(' a=')
3200  format(' harmonics=')
      stop
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine a                                           c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine fa( a,da,n )
        dimension a(10),da(10)
        common pi,conv

        do 200 i=1,n
          a(i)=a(i)-da(i)
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine  close                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sclose(dummy)
        close(1)
        close(2)
        close(3)
        close(4)
        close(7)
        close(8)
        close(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine df                                          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine fdf( a,r,f,df,n )
        dimension a(10),r(10),f(10),df(10,10)
        common pi,conv

        do 200 i=1,n
          sign=1.0
                do 100 j=1,n
                  df(i,j)=-sign*r(i)*sin( r(i)*a(j) )
                  sign=-sign
100             continue
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine f                                           c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine ff( a,r,f,n )
        dimension a(10),r(10),f(10)
        common pi,conv
        do 200 i=1,n
          sign=1.0
          f(i)=0.0
                do 100 j=1,n
                  f(i)=f(i)+sign*cos( r(i)*a(j) )
                  sign=-sign
100             continue
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c              subroutine harmonics                                    c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine shar( a,tr,h,n,m )
        dimension a(10),h(10),tr(10)
        common pi,conv

        do 200 i=1,m
          sign=1.0
          h(i)=0.0
                do 100 j=1,n
                  h(i)=h(i)+4.0/( tr(i)*pi )*sign*cos( tr(i)*a(j) )
                  sign=-sign
100             continue
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine hsquare                                     c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine shsq( h,hsquar,n )
        dimension h(10)
        hsquar=0.0
        do 200 i=2,n+1
          hsquar=hsquar+h(i)*h(i)
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine look                                        c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine look( a,r,f,fsquar,n )
        dimension a(10),r(10),f(10),df(10,10),dfi(10,10)
        common pi,conv
        kount=30
        fmin=0.01
        do 2500 loop=1,kount
        call fdf( a,r,f,df,n )
        call matv( df,n,det,dfi )
        call mult1( f,dfi,da,n,n )
        call fa( a,da,n )
        call ff( a,r,f,n )
        call square( f,fsquar,n )
        if( loop .gt. 5 .and. fsquar .lt. fmin ) go to 2900
2500    continue
2900    dummy=0.0
3000    format( 10(e15.7,5x) )
3050    format( '0' )
3100    format( ' a=' )
3150    format( ' f=' )
3200    format( ' fsquar=' )
4000    format('test',i3)
        return
        end
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C             SUBROUTINE MATV                                          C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE MATV(A,N,DET,B)
        DIMENSION INDEX(20,2),IPVOT(20),A(10,10),B(10,10),PIVOT(20)
        ONE=1.0
        ZERO=0.0

C
        DO 10 I=1,N

          DO 5 J=1,N
            B(I,J)=A(I,J)
5         CONTINUE

10      CONTINUE
        DET=ONE
        DO 15 J=1,N
15       IPVOT(J)=0

        DO 75 I=1,N
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                             C
C               FOLLOWING 15 STATEMENTS                       C
C                FOR SEARCH FOR PIVOT ELEMENTS                C
C                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        T=ZERO
        DO 40 J=1,N
          IF(IPVOT(J)-1) 20,40,20
20        DO 35 K=1,N
            IF(IPVOT(K)-1) 25,35,100
25          CONTINUE
          D1=ABS(T)
          D2=ABS( B(J,K) )
          IF(D1-D2)30,35,35
30          IROW=J
            ICOL=K
            T=B(J,K)
35        CONTINUE

40      CONTINUE
```

```
        IPVOT(ICOL)=IPVOT(ICOL)+1
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                               C
C                FOLLOWING 10 STATEMENTS PUT                    C
C                PIVOT ELEMENT ON DIAGONAL                      C
C                                                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        IF(IROW-ICOL) 45,55,45
45        DET=-DET
        DO 50 L=1,N
          T=B(IROW,L)
          B(IROW,L)=B(ICOL,L)
50        B(ICOL,L)=T
55      INDEX(I,1)=IROW
        INDEX(I,2)=ICOL
        PIVOT(I)=B(ICOL,ICOL)
        DET=DET*PIVOT(I)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                        C
C           FOLLOWING 3 STATEMENTS TO DIVIDE PIVOT       C
C           ROW BY PIVOT ELEMENT                         C
C                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        B(ICOL,ICOL)=ONE
        DO 60 L=1,N
60      B(ICOL,L)=B(ICOL,L)/PIVOT(I)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                        C
C           FOLLOWING 7 STATEMENTS TO REDUCE             C
C           NON-PIVOT ROWS                               C
C                                                        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        DO 75 LI=1,N
          IF(LI-ICOL) 65,75,65
65        T=B(LI,ICOL)
          B(LI,ICOL)=ZERO
            DO 70 L=1,N
70          B(LI,L)=B(LI,L)-B(ICOL,L)*T
75      CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C               FOLLOWING 11 STATEMENTS TO                           C
C               INTERCHANGE COLUMNS                                  C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      DO 95 I=1,N
        L=N-I+1
        IF(INDEX(L,1)-INDEX(L,2) ) 85,95,85
85      JROW=INDEX(L,1)
        JCOL=INDEX(L,2)
          DO 90 K=1,N
              T=B(K,JROW)
              B(K,JROW)=B(K,JCOL)
              B(K,JCOL)=T
90        CONTINUE
95    CONTINUE
100   RETURN
      END
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
c               subroutine multiply                                   c
c                                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine mult1( a,b,c,i0,k0 )
        dimension a(10),b(10,10),c(10)
        do 200 i=1,i0
          c(i)=0.0
          do 100 k=1,k0
            c(i)=c(i)+b(i,k)*a(k)
100       continue
200     continue
        return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                     c
c               subroutine  open                                      c
c                                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sopen(dummy)
        open(unit=1,file="data0/alpha")
        rewind(1)
        open(unit=2,file="data0/fundamental")
        rewind(2)
        open(unit=3,file="data0/har2")
        rewind(3)
        open(unit=4,file="data0/har3")
        rewind(4)
        open(unit=7,file="data0/imghar")
        rewind(7)
        open(unit=8,file="data0/hmag")
        rewind(8)
        open(unit=9,file="input")
        rewind(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine square                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine square( f,fsquar,n )
        dimension f(10)
        fsquar=0.0
        do 200 i=1,n
          fsquar=fsquar+f(i)*f(i)
200     continue
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine square                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine square( f,fsquar,n )
        dimension f(10)
        fsquar=0.0
        do 200 i=1,n
          fsquar=fsquar+f(i)*f(i)
200     continue
        return
        end
```

APPENDIX C

FOURIER SPECTRUM PROGRAM

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c            frequency spectrum                                        c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       tor     - input data (this is a real variable)
c       x       - complex variable containing the input data
c                 before the fft subroutine call and
c                 the complex frequency after the fft subroutine call
c       mag     - magnitude of the complex frequency
c       n       - number of data points (must be a power of 2)
c       inv     - used by fft
c                 inv=0    fft
c                 inv=1    inverse fft
c       maxi    - used to decided highest frequency to be plotted
c       avemag  - used to normalize data to 100% of the largest
c                 harmonic
        dimension tor(1030)
        complex x(1030),cmplx
        real mag
        call sopen(0.0)

        pi=3.1415926535
        rms=0.0
        read(9,*) n
        write(6,*) ' n=',n
        read(9,*) inv
        write(6,*) ' inv=',inv
        read(9,*) maxi
        write(6,*) ' maxi=',maxi
        read(9,*) iwrite
        write(6,*) ' iwrite=',iwrite

        do 100 i=1,n
        if( inv .eq. 0 ) read(1,2000) tor(i)
        if( inv .eq. 0 ) x(i)=cmplx( tor(i),0.0 )
        if( inv .eq. 1 ) read(4,2000) x(i)
        if( iwrite .eq. 1 .and. inv .eq. 0 ) write(4,*) x(i)
100     continue

        call fft( x,n,inv )

        do 200 i=1,maxi
        mag=cabs( x(i) )
        phase=atan2( yy,xx )
        avemag=amax1( cabs(x(1)),cabs(x(2)),cabs(x(3)),cabs(x(4)) )
        rms=rms+mag**2
        write(3,*) mag/avemag*100.0
```

```
200     continue

        write(6,*) "avemag=",avemag
        write(6,*) "rms=",sqrt( rms*2.0 )
        write(6,*) "rms normalized=",sqrt( rms/2.0 )/avemag

2000    format( e10.4 )
        call sclose(0.0)
        stop
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine  close                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sclose(dummy)
        close(1)
        close(2)
        close(3)
        close(4)
        close(7)
        close(8)
        close(9)
        return
        end
```

```
c
c
c
c**The following fft subroutine is taken almost verbatim
c**from Ahmed N., Rao K.R., Orthogonal Transforms for
c**Digital Signal Processing, Springer-Verlag, 1975., pp79
c
c
c
      subroutine  fft(x,n,inv)
c*************************************************************************
c
c  this program implements the fft algorithm to compute the discrete
c  fourier coefficients of a data sequence of n points
c
c  calling sequence from the main program:
c  call fft(x,n,inv)
c      n: number of data points
c      x: complex array containing the data sequence. In the end dft
c         coeffs. are returned in the array. Main program should
c         declare it as--     complex x(1024)
c      inv: flag for inverse
c           inv=0  for forward transform
c           inv=1  for inverse transform
c
c*************************************************************************
      complex  x(1030),w,t,cmplx
c
c       calculate the # of iterations (log. n to the base 2)
c
      iter=0
      irem=n
   10 irem=irem/2
      if (irem.eq.0)  go  to  20
      iter=iter+1
      go  to  10
   20 continue
      sign=-1
      if  (inv.eq.1) sign=1
      nxp2=n
      do  50 it=1,iter
c
c     computation  for  each iteration
c     nxp:number  of  points  in a partition
c     nxp2:nxp/2
      nxp=nxp2
      nxp2=nxp/2
      wpwr=3.141592/float(nxp2)
      do  40  m=1,nxp2
c
```

```
c     calculate the multipier
c
      arg=float(m-1)*wpwr
      w=cmplx(cos(arg),sign*sin(arg))
      do  40 mxp=nxp,n,nxp
c
c     computation  for  each  partition
c
      j1=mxp-nxp+m
      j2=j1+nxp2
      t=x(j1)-x(j2)
      x(j1)=x(j1)+x(j2)
   40 x(j2)=t*w
   50 continue
c
c     unscramble the  bit reversed  dft  coeffts.
c
      n2=n/2
      n1=n-1
      j=1
      do  65 i=1,n1
      if(i.ge.j) go  to  55
      t=x(j)
      x(j)=x(i)
      x(i)=t
   55 k=n2
   60 if(k.ge.j) go  to  65
      j=j-k
      k=k/2
      go  to  60
   65 j=j+k
      if (inv.eq.1) go  to 75
      do 70 i=1,n
   70 x(i)=x(i)/float(n)
   75 continue
      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c                subroutine  open                                      c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine sopen(dummy)
        open(unit=1,file="torque")
        rewind(1)
        open(unit=2,file="data0/phase")
        rewind(2)
        open(unit=3,file="data0/Percent")
        rewind(3)
        open(unit=4,file="file1")
        rewind(4)
        open(unit=7,file="data0/misc2")
        rewind(7)
        open(unit=8,file="data0/misc3")
        rewind(8)
        open(unit=9,file="input")
        rewind(9)
        return
        end
```

APPENDIX D

PERFORMANCE PROGRAM

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               program to calculate developed power-                  c
c                                                                      c
c               speed points for brushless dc motor                    c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c       rpm     - speed of dc motor
c       wm      - angular speed of dc motor
c       p       - number of poles
c       t       - time
c       k0      - number of intervals between print outs
c       h       - interval of integration
c       w       - electrical angular frequency of pmm
c       ws      - electrical angular frequency of pmg
c       alpha   - commutation advance angle (degrees)
c       ra      - stator resistance
c       xla     - stator leakage inductance
c       r0      - choke coil resistance
c       xlo     - choke coil inductance
c       xk      - motor constant
c       xkg     - generator constant
c       vd      - inverter input voltage
c       x       - array of stator phase currents
c       tp      - period
c       ang     - electrical angular displacement
c       xlamda  - electrical angular displacement (pmg)
c       m       - number of phases
c       pd      - total developed power
c       td      - total developed torque
c       pd1     - phase 1 developed power
c       pd2     - phase 2 developed power
c       pd3     - phase 3 developed power
c       e(1)    - excitation voltage of phase 1
c       e(2)    - excitation voltage of phase 2
c       e(3)    - excitation voltage of phase 3
c
c
        dimension f(6),v(6),x(6),xx(6),a(6),r(6),e(6),alphs(2)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        real iav,irms
        dummy=0.0
        call popen(dummy)
        pi=3.1415926536
        conv=pi/180.0
        read(9,*) istop
```

```
write(6,*) ' istop=',istop
read(9,*) alpha
write(6,*) ' alpha=',alpha
read(9,*) ( a(i),i=1,5 )
write(6,*) ' a=',( a(i),i=1,5 )
read(9,*) h
write(6,*) ' h=',h
read(9,*) r0
write(6,*) ' r0=',r0
read(9,*) xl0
write(6,*) ' xl0=',xl0
read(9,*) ra
write(6,*) ' ra=',ra
read(9,*) xla
write(6,*) ' xla=',xla
read(9,*) rd
write(6,*) ' rd=',rd
read(9,*) rpm0
write(6,*) ' rpm0=',rpm0
read(9,*) xk
write(6,*) ' xk=',xk
read(9,*) xkg
write(6,*) ' xkg=',xkg
read(9,*) pmax
write(6,*) ' pmax=',pmax
read(9,*) pmin
write(6,*) ' pmin=',pmin
read(9,*) vdc
write(6,*) ' vdc=',vdc
read(9,*) points
write(6,*) ' points=',points
read(9,*) ivdon
write(6,*) ' ivdon=',ivdon
read(9,*) imodon
write(6,*) ' imodon=',imodon
read(9,*) ichon
write(6,*) ' ichon=',ichon
read(9,*) iwgon
write(6,*) ' iwgon=',iwgon
read(9,*) ( x(i),i=1,3 )
write(6,*) ' x=',( x(i),i=1,3 )
read(9,*) iarea
write(6,*) ' iarea=',iarea
read(9,*) xwg
write(6,*) ' xwg=',xwg
read(9,*) ifull
write(6,*) ' ifull=',ifull

do 4 j=1,5
if( imodon .eq. 1 ) a(j)= a(j)*pi/180.0
```

```
      if( imodon .eq. 0 ) a(j)=pi/2.0
      if( imodon .eq. 0 ) a(1)=30.00*pi/180.0
4     continue
      m=3
      pfw=(rpm0/45000.0)**2*330.0
      wm=rpm0*pi/30.0
      if( iwgon .eq. 1 ) wg=7950*pi
      if( iwgon .eq. 0 ) wg=xwg*wm
      wm=abs(wm)
      p=4.0
      t=0.0
      w=p/2.0*wm
      ws=2.0*wg
      vm=sqrt(3.0)*xkg*wg
      write(6,*) ' vm=',vm
      alpha=alpha*conv
      area=0.5+0.5*( pi/3.0-a(1)+a(2)-a(3)+a(4)-a(5) )/( pi/3.0 )
      if( imodon .ne. 0 .and. iarea .eq. 1 ) vdc=vdc/area
      if( ifull .eq. 1 ) alphs(1)=acos( (vdc*sqrt(2.0))/(1.0*vm*1.35) )
      if( ifull .eq. 1 ) alphs(2)=acos( (vdc*sqrt(2.0))/(2.0*vm*1.35) )
      if( ifull .eq. 0 ) alphs(1)=acos( (vdc*sqrt(2.0))/(0.5*vm*1.35) )
      if( ifull .eq. 0 ) alphs(2)=acos( (vdc*sqrt(2.0))/(1.0*vm*1.35) )
      write(6,*) ' alphs(1)=',alphs(1)/conv
      write(6,*) ' alphs(2)=',alphs(2)/conv
      write(6,*) ' area=',area
      write(6,*) ' f=',w/(2.0*pi)
      write(6,*) ' fs=',ws/(2.0*pi)
      write(6,*) ' mag of e1=',xk*w/2.0
      xx(1)=0.0
      xx(2)=0.0
      xx(3)=0.0
      pd=0.0
      pin=0.0
      iav=0.0
      irms=0.0
      e1ave=0.0
      vdave=0.0
      e1rms=0.0
      sum=0.0
      tp=2.0*pi/w
      tmin=pmin*tp
      tmax=pmax*tp
      tmxmn=tmax-tmin
      scale=30.0/points
      k0=int( tmxmn/tp*scale/abs(rpm0)/h )
      kount=0
100   index=1
      rewind(9)
      read(9,*) istop
      if( istop .eq. 1 ) go to 305
```

```
      kount=kount+1
200   ang=w*t
      iprin1=int(ang/2.0/pi)
      ang=ang-float(iprin1)*2.0*pi
      e(1)=xk*w*2.0/p*sin(ang-alpha)
      e(2)=xk*w*2.0/p*sin(ang-alpha-2.0/3.0*pi)
      e(3)=xk*w*2.0/p*sin(ang-alpha+2.0/3.0*pi)
      wst=ws*t0
      if( wst .ge. 2.0*pi ) t0=0.0
      wst=ws*t0
      if( ifull .eq. 0 ) call half( wst,xlamda )
      if( ifull .eq. 1 ) call full( wst,xlamda )
      call fv( a,ang,v,vd,alphs,xlamda )
      call ff( index,t,f,x,xi,ang,del,v )
      if( ichon .eq. 1 ) call check( m,x,f )
      call derk( m,x,f,t,t0,h,index )
      if( index .eq. 2 ) go to 200
      if( t .lt. tmin ) go to 100
      f(1)=( x(1)-xx(1) )/h
      f(2)=( x(2)-xx(2) )/h
      f(3)=( x(3)-xx(3) )/h
      do 86 kk=1,3
        xx(kk)=x(kk)
86    continue
      call ffi( fi,f,ang )
      call fvap( wst,vap,alphas )
      v1p=vd-fi*xl0/2.0-xi*r0-x(1)*(r(1)-ra)
      vscr=vap-v1p
      if( ang .lt. 3.0/3.0*pi .and. wst .lt. 2.0/3.0*pi )
    c   vcsr=x(1)*(r(1)-ra)
      if( ang .ge. 3.0/3.0*pi .and. wst .ge. 3.0/3.0*pi .and.
    c   wst .lt. 4.0/3.0*pi ) vscr=x(1)*(r(1)-ra)
85    if( t .ge.tmax ) go to 300
      if( kount .lt. k0 ) go to 100
300   sum=sum+1.0
      pd1=x(1)*e(1)
      pd2=x(2)*e(2)
      pd3=x(3)*e(3)
      td=( pd1+pd2+pd3 )/wm
      pd=pd+( pd1+pd2+pd3 )
      pin=pin+( v(1)*x(1)+v(2)*x(2)+v(3)*x(3) )
      irms=irms+x(1)**2
      iav=iav+abs( x(1) )
      e1rms=e1rms+e(1)**2
      e1ave=e1ave+abs( e(1) )
      vdave=vdave+vd
      hsin6=hsin6+td*sin(6.0*ang)
      hcos6=hcos6+td*cos(6.0*ang)
      hsin12=hsin12+td*sin(12.0*ang)
      hcos12=hcos12+td*cos(12.0*ang)
```

```
      hsin18=hsin18+td*sin(18.0*ang)
      hcos18=hcos18+td*cos(18.0*ang)
      hsin24=hsin24+td*sin(24.0*ang)
      hcos24=hcos24+td*cos(24.0*ang)
      tx=t-tmxmn
      txx=t*w*180.0/pi
      write(1,2000) x(1)
      write(2,3000) txx
      write(3,2000) td
      write(4,2000) v(1)
      write(7,2000) vd
      write(8,2000) x(1)+x(2)+x(3)
      kount=0
      if( t .lt. tmax ) go to 100
305   pd=pd/sum
      pin=pin/sum
      irms=irms/sum
      irms=sqrt( irms )
      iav=iav/sum
      e1rms=e1rms/sum
      e1rms=sqrt( e1rms )
      e1ave=e1ave/sum
      vdave=vdave/sum
      hsin6=hsin6/sum
      hcos6=hcos6/sum
      h6=hsin6**2+hcos6**2
      h6=sqrt( h6 )/6.0
      hsin12=hsin12/sum
      hcos12=hcos12/sum
      h12=hsin12**2+hcos12**2
      h12=sqrt( h12 )/12.0
      hsin18=hsin18/sum
      hcos18=hcos18/sum
      h18=hsin18**2+hcos18**2
      h18=sqrt( h18 )/18.0
      hsin24=hsin24/sum
      hcos24=hcos24/sum
      h24=hsin24**2+hcos24**2
      h24=sqrt( h24 )/24.0
600   tsav=(pd-pfw)/wm
      write(6,*) ' tsav=',tsav
      eff=(pd-pfw)/pin*100.0
      if( eff .lt. 100.0 ) go to 710
      eff=pin/(pd+pfw)*100.0
710   write(6,*) ' pin=',pin
      write(6,*) ' eff=',eff
      write(6,*) ' iav=',iav
      write(6,*) ' irms=',irms
      write(6,*) ' e1ave=',e1ave
      write(6,*) ' vdave=',vdave
```

```
      write(6,*) ' h6=',h6
      write(6,*) ' h12=',h12
      write(6,*) ' h18=',h18
      write(6,*) ' h24=',h24
1000  format( 5(e12.4,5x) )
2000  format( e10.4 )
3000  format( f8.3 )
      call pclose(dummy)
      stop
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine check                                       c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine check( m,x,f )
        dimension x(6),f(6)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        do 10 i=1,m-1
          signf=abs( f(i) )/f(i)
          signx=abs( x(i) )/x(i)
          if( abs( f(i) ) .gt. 1.0e08 ) f(i)=signf*1.0e08
          if( abs( x(i) ) .gt. 1.0e06 ) x(i)=signx*1.0e06
10      continue
        f(3)=-( f(1)+f(2) )
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c                    subroutine pclose                                 c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine pclose(dummy)
        close(1)
        close(2)
        close(3)
        close(4)
        close(7)
        close(8)
        close(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine derk                                        c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c       derk is a fourth-order, fixed increment runge-kutta
c       integration routine
c
c       m       - number of simultaneous differential equations
c       x       - array of dependent variables
c       f       - array of derivatives of independent varibles
c       h       - increment
c       t       - time
c       index   - indicator
c       1. if exit routine with index=1, solution found at=t+h
c       2. if exit routine with index=2, go back to re-evaulate
c
        subroutine derk( m,x,f,t,t0,h,index )
        dimension x(6),f(6),q(400)
        if( index .eq. 2 ) go to 19
18      kxx=0
        index=2
        do 35 i=1,m
        j=i+300
35      q(j)=x(i)
19      kxx=kxx+1
        go to (1,2,3,4),kxx
1       do 5 i=1,m
        q(i)=f(i)*h
5       x(i)=x(i)+q(i)/2.00
        t=t+h/2.00
        t0=t0+h/2.00
        return
2       do 6 i=1,m
        j=i+100
        k=i+300
        q(j)=f(i)*h
6       x(i)=q(k)+q(j)/2.00
        return
3       do 7 i=1,m
        j=i+200
        k=i+300
        q(j)=f(i)*h
7       x(i)=q(k)+q(j)
        t=t+h/2.00
        t0=t0+h/2.00
        return
```

```
4       do 8 i=1,m
        j=i+100
        k=i+200
        l=i+300
8       x(i)=q(l)+( ( q(i)+2.00*q(k)+2.00*q(k)+f(i)*h )/6.00 )
        index=1
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c                subroutine f                                          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        subroutine ff( index,t,f,x,xi,ang,del,v )
        dimension f(6),v(6),x(6)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        rr=rd+ra
        if( index .eq. 1 ) r(1)=rr
        if( index .eq. 1 ) r(2)=rr
        if( index .eq. 1 ) r(3)=rr
        if( ang .ge. 0.0         .and. ang .lt. 1.0/3.0*pi ) goto 10
        if( ang .ge. 1.0/3.0*pi .and. ang .lt. 2.0/3.0*pi ) goto 20
        if( ang .ge. 2.0/3.0*pi .and. ang .lt. 3.0/3.0*pi ) goto 30
        if( ang .ge. 3.0/3.0*pi .and. ang .lt. 4.0/3.0*pi ) goto 40
        if( ang .ge. 4.0/3.0*pi .and. ang .lt. 5.0/3.0*pi ) goto 50
        if( ang .ge. 5.0/3.0*pi .and. ang .lt. 6.0/3.0*pi ) goto 60

10      n1=1
        n2=3
        n3=2
        if( icycle .eq. 1 ) iflag=0
        itest=10
        icycle=0
        go to 100
20      n1=1
        n2=2
        n3=3
        itest=20
        go to 200
30      n1=2
        n2=1
        n3=3
        itest=30
        go to 100
40      n1=2
        n2=3
        n3=1
        itest=40
        go to 200
50      n1=3
        n2=2
        n3=1
        itest=50
        go to 100
60      n1=3
```

```
      n2=1
      n3=2
      itest=60
      icycle=1
      go to 200
100   if( x(n2) .lt. -0.001 .and. index .eq. 1 ) iflag=itest
      if( x(n1) .lt. -0.001 .and. index .eq. 1 ) r(n1)=rr+1000.0*abs( x(n1) )
      if( x(n3) .gt.  0.001 .and. index .eq. 1 ) r(n3)=rr+1000.0*abs( x(n3) )
      if( iflag .eq. itest .and. index .eq. 1 )  r(n2)=rr+1000.0*abs( x(n2) )

      do 150 i=1,3
      if( r(i) .gt. 1000.0 ) r(i)=1000.0
150   continue

      xl=xla

      f(n1)=( -x(n1)*r(n1)+v(n1)-e(n1) )/xl
      f(n2)=( -x(n2)*r(n2)+v(n2)-e(n2) )/xl
      f(n3)=( -x(n3)*r(n3)+v(n3)-e(n3) )/xl
      return

200   if( x(n2) .gt.  0.001 .and. index .eq. 1 ) iflag=itest
      if( x(n3) .gt.  0.001 .and. index .eq. 1 ) r(n3)=rr+1000.0*abs( x(n3) )
      if( x(n1) .lt. -0.001 .and. index .eq. 1 ) r(n1)=rr+1000.0*abs( x(n1) )
      if( iflag .eq. itest .and. index .eq. 1 )  r(n2)=rr+1000.0*abs( x(n2) )

      do 250 i=1,3
      if( r(i) .gt. 1000.0 ) r(i)=1000.0
250   continue

      xl=xla

      f(n1)=( -x(n1)*r(n1)+v(n1)-e(n1) )/xl
      f(n2)=( -x(n2)*r(n2)+v(n2)-e(n2) )/xl
      f(n3)=( -x(n3)*r(n3)+v(n3)-e(n3) )/xl

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine fi                                          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        subroutine ffi( fi,f,ang )
        dimension f(6)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        if( ang .ge. 0.0/3.0*pi .and. ang .lt. 1.0/3.0*pi ) fi=-f(2)
        if( ang .ge. 1.0/3.0*pi .and. ang .lt. 2.0/3.0*pi ) fi= f(1)
        if( ang .ge. 2.0/3.0*pi .and. ang .lt. 3.0/3.0*pi ) fi=-f(3)
        if( ang .ge. 3.0/3.0*pi .and. ang .lt. 4.0/3.0*pi ) fi= f(2)
        if( ang .ge. 4.0/3.0*pi .and. ang .lt. 5.0/3.0*pi ) fi=-f(1)
        if( ang .ge. 5.0/3.0*pi .and. ang .lt. 6.0/3.0*pi ) fi= f(3)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine full-bridge                                 c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        subroutine full( wst,xlamda )
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        if( wst .ge. 0.0        .and. wst .lt. 1.0*pi/3.0 ) xlamda=
     c    wst-0.0
        if( wst .ge. 1.0*pi/3.0 .and. wst .lt. 2.0*pi/3.0 ) xlamda=
     c    wst-1.0*pi/3.0
        if( wst .ge. 2.0*pi/3.0 .and. wst .lt. 3.0*pi/3.0 ) xlamda=
     c    wst-2.0*pi/3.0
        if( wst .ge. 3.0*pi/3.0 .and. wst .lt. 4.0*pi/3.0 ) xlamda=
     c    wst-3.0*pi/3.0
        if( wst .ge. 4.0*pi/3.0 .and. wst .lt. 5.0*pi/3.0 ) xlamda=
     c    wst-4.0*pi/3.0
        if( wst .ge. 5.0*pi/3.0 .and. wst .lt. 6.0*pi/3.0 ) xlamda=
     c    wst-5.0*pi/3.0
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine half-bridge                                 c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        subroutine half( wst,xlamda )
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        if( wst .ge. 0.0        .and. wst .lt. 2.0*pi/3.0 ) xlamda=
     c    wst-0.0
        if( wst .ge. 2.0*pi/3.0 .and. wst .lt. 4.0*pi/3.0 ) xlamda=
     c    wst-2.0*pi/3.0
        if( wst .ge. 4.0*pi/3.0 .and. wst .lt. 6.0*pi/3.0 ) xlamda=
     c    wst-4.0*pi/3.0
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine popen                                       c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine popen(dummy)
        open(unit=1,file="data0/I1")
        rewind(1)
        open(unit=2,file="data0/Time")
        rewind(2)
        open(unit=3,file="data0/Torque")
        rewind(3)
        open(unit=4,file="data0/Van")
        rewind(4)
        open(unit=7,file="data0/Vd")
        rewind(7)
        open(unit=8,file="data0/In")
        rewind(8)
        open(unit=9,file="input")
        rewind(9)
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c             subroutine vd                                            c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine fv( a,ang,v,vd,alphs,xlamda )
        dimension a(6),v(6),alphs(2)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        beta=xlamda+alphs(1)
        if( ivdon .eq. 1 .and. ifull .eq. 1 ) vd=vm*sin( beta+pi/3.0 )
        if( ivdon .eq. 1 .and. ifull .eq. 0 ) vd=vm*sin( beta+pi/6.0 )
        if( ivdon .eq. 0 ) vd=vdc
        shift=30.0/180.0*pi
        theta=shift+ang
        call svxn( vx,theta,a )
        van=vx
        theta=shift+ang-2.0/3.0*pi
        call svxn( vx,theta,a )
        vbn=vx
        theta=shift+ang+2.0/3.0*pi
        call svxn( vx,theta,a )
        vcn=vx

        v(1)=vd*van
        v(2)=vd*vbn
        v(3)=vd*vcn
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c               subroutine fvap                                        c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        subroutine fvap( wst,vap,alps )
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        vaa=0.0
        vab= vm*sin( wst+1.0/3.0*pi+alps )
        vac=-vm*sin( wst+5.0/3.0*pi+alps )
        if( wst .ge. 0.0/3.0*pi .and. wst .lt. 1.0/3.0*pi ) vap= vab
        if( wst .ge. 1.0/3.0*pi .and. wst .lt. 2.0/3.0*pi ) vap= vac
        if( wst .ge. 2.0/3.0*pi .and. wst .lt. 3.0/3.0*pi ) vap= vac
        if( wst .ge. 3.0/3.0*pi .and. wst .lt. 4.0/3.0*pi ) vap= vaa
        if( wst .ge. 4.0/3.0*pi .and. wst .lt. 5.0/3.0*pi ) vap= vaa
        if( wst .ge. 5.0/3.0*pi .and. wst .lt. 6.0/3.0*pi ) vap= vab
        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
c            subroutine  vxn                                             c
c                                                                        c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine svxn( vx,th,a )
        dimension a(6)
        common rr,rd,r0,ra,r(6),pi,xla,xl0,e(6),vm,vdc,ivdon,imodon
        vx=0.0
        xa=0.0
        xb=pi
        xc=2.0*pi
        a(6)=pi/2.0
        th=th+2.0*pi
        th=th-float( int(th/xc) )*xc
        if( th .ge. xa+a(1) .and. th .lt. xa+a(2) ) vx=1.0
        if( th .ge. xa+a(3) .and. th .lt. xa+a(4) ) vx=1.0
        if( th .ge. xa+a(5) .and. th .lt. xa+a(6) ) vx=1.0
        if( th .ge. xb-a(6) .and. th .lt. xb-a(5) ) vx=1.0
        if( th .ge. xb-a(4) .and. th .lt. xb-a(3) ) vx=1.0
        if( th .ge. xb-a(2) .and. th .lt. xb-a(1) ) vx=1.0
        if( th .ge. xb+a(1) .and. th .lt. xb+a(2) ) vx=-1.0
        if( th .ge. xb+a(3) .and. th .lt. xb+a(4) ) vx=-1.0
        if( th .ge. xb+a(5) .and. th .lt. xb+a(6) ) vx=-1.0
        if( th .ge. xc-a(6) .and. th .lt. xc-a(5) ) vx=-1.0
        if( th .ge. xc-a(4) .and. th .lt. xc-a(3) ) vx=-1.0
        if( th .ge. xc-a(2) .and. th .lt. xc-a(1) ) vx=-1.0
        return
        end
```