This is to certify that the

dissertation entitled

Probabilistic Models to Represent
Loads Due To Human Activities

presented by

Bassem K. Khafagi

has been accepted towards fulfillment
of the requirements for

Doctor of Philosophy degree in Civil Engineering

William E. Saul
Major professor

Date March 30, 1993

PROBABILISTIC MODELS TO REPRESENT LOADS DUE TO HUMAN ACTIVITIES

By

Bassem K. Khafagi

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Civil and Environmental Engineering

April 1993

ABSTRACT

PROBABILISTIC MODELS TO REPRESENT LOADS DUE TO HUMAN ACTIVITIES

BY

Bassem K. Khafagi

Human loads are random and produce dynamic force components which cannot be predicted with certainty. Current codes account for human loads by assuming a static load equivalence that included the maximum dynamic effect of human movements. The primary objective of this research is to generate probabilistic functions which accurately represent the dynamic force caused by several human actions.

Measured data were obtained from three sets of experiments, two designed to measure forces due to the action of an individual on a force platform; and the third designed to measure the forces due to the action of a group on a floor system. The study included six different in-place human movements grouped into two categories; transient motions (single jump, standing up suddenly, and dropping into a seat), and periodic motions (jumping, jouncing, and swaying side to side).

Transient actions were modeled by passing straight or curved lines through control points that define the force-time history. Best-fit distributions for the control points were determined. Model error was developed as a Gaussian random process and was incorporated into the model to better represent the modeled actions. Periodic loadings were analyzed as a series of cycles where each cycle is a full period of the motion considered. The period and peak force ratio of each cycle were modeled as first order autoregressive processes. Model verification was applied to the different motions considered in the study. Stochastic models were compared with experimental data and feedback from the process was used to refine the models.

A computer program; *H*uman *L*oad *S*imulation, ***HLS***, was developed to simplify generating load distributions for any combination of human loading conditions. Output of the program includes the statistical parameters of the control points associated with the time histories, amplitudes of peaks and their arrival time, spectral energy distributions for periodic motions, and force-time history for each simulation. A modified version of ***HLS*** was developed to simulate group loading conditions.

Several potential applications of ***HLS*** were presented. Code values were examined and found to be acceptable except for the transverse force component where a value of 45 lb/ft was recommended to replace the current 10 lb/ft value in the code. An example showing the advantage of using probabilistic loads was presented. Other potential applications of the research outcome were discussed. It is anticipated that research findings will have an impact on standards for structural design and serviceability control.

To Ekram,

Yasmeen, Louai, and ....

ACKNOWLEDGMENTS

In the course of this project, I have received assistance in a variety of forms from many people. I am indebted to them, and I wish to acknowledge this debt. My heartfelt thanks to Professor W. Saul and Professor R. Harichandran for their cooperation, enlightening advice and devoting part of their time to guide and help me to overcome the difficulties I have faced during my research program; Professor T. Wolf, Professor F. Hatfiled, and Professor G. Ludden, other members of my Graduate Committee, for their recommendations and review of the dissertation; Special thanks go to Ahmad Al-Ghamedi and Abdulrahman Al-Hadlag for their friendship and incessant help.

I am grateful to my enlightened parents who supported and encouraged me throughout my education. Finally, I believe that God is behind any success I have reached. So, my thanks to Him. .....

TABLE OF CONTENTS

LIST OF TABLES

Table		Page

LIST OF FIGURES

Figure		Page

1.0 INTRODUCTION

1.1 General

Human loads are random and therefore cannot be predicted with certainty. In addition to the forces due to static weights, the movement of persons produces dynamic force components. Human loads vary from event to event in an apparent pattern and therefore it was postulated they could be shown to obey the laws of probability and hence would be predictable. In addition to many other possible applications, these loads are the primary concern in the design and analysis of assembly structures. A challenge a structural engineer faces is how to quantify the uncertainty associated with loads due to human movements and provide corresponding measures to assure safety and serviceability of the structure at a reasonable cost. This was the initial motivation for attempting to provide procedures for generating functions to represent forces due to human movements.

When a person on a horizontal surface is absolutely motionless a vertical force is brought to bear on the support due to gravitational acceleration acting on the mass of the person's body, the weight of the person, as shown in Figure 1.1.a. The value of the static force could be predicted by a probability distribution model of the individual weights of a population sample of people. When activity takes place, such as jumping, sitting down, or swaying, there are resultant vertical and horizontal forces which fluctuate with time due to the changing acceleration of the person's mass, as illustrated in Figure

1.1.b. The vertical component of this force varies from zero to several times the person's weight. Horizontal components do not exist in the static situation. The time variation for any one of various human actions, for instance a single jump in place, provides a pattern characteristic of that action. Similar to the distribution model for

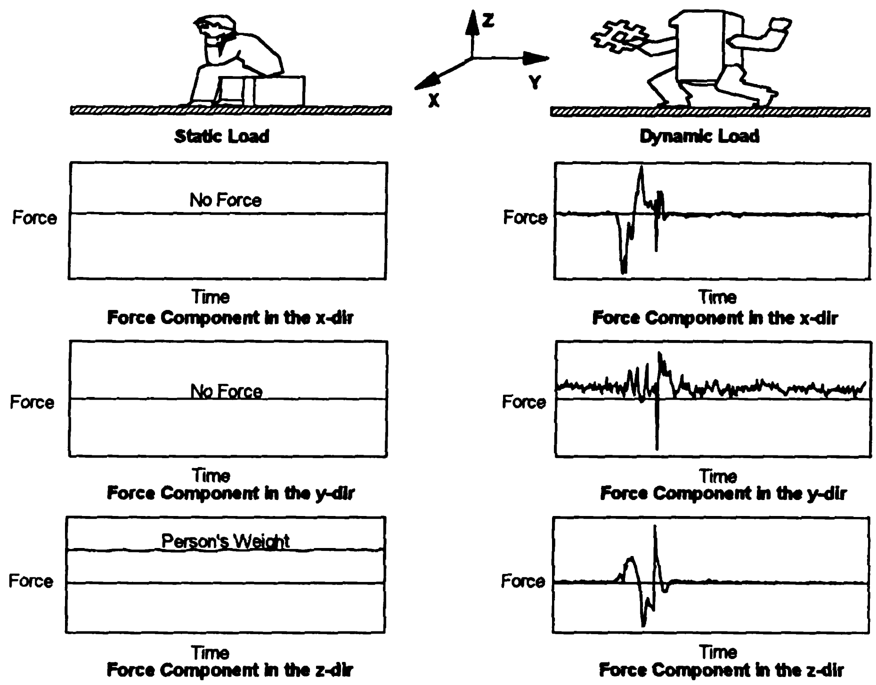

Figure 1.1 Human Loads As Static and Dynamic Forces

predicting the weight of an individual in a group of people, the parameters defining the characteristic pattern of a force due to a human action vary randomly within the bounds of distribution models.

1.2 Problem Statement

For the design of structures to avoid catastrophic failure, the uncertainty of human loads was Traditionally accounted for by assuming a static load equivalence that included the maximum dynamic effect of human movements. Recent incidents of occupant discomfort, and strong vibration of assembly structures, Bachmann (1992), have reinforced the need for quantification of dynamic human loadings. In recent years, researchers have been utilizing probabilistic methods to model loads and calculate the probabilities associated with combined load effects. A number of investigations have been conducted on human loads variabilities which have resulted in several important achievements, Allen (1990), Bachmann (1992), and Murray (1991). However, up until this study, no satisfactory method existed for determining the statistics of the dynamic load one person in a group would generate when performing various types of human movements.

The focus of this study is the modeling and analysis of human loadings to include and quantify their random nature. The impact of the research on the evaluation of safety of assembly structures and determination of design loads and load combinations in building codes is summarized.

1.3 Objectives of the Study

The primary objective of the research is to generate probabilistically-based functions which will accurately represent the dynamic force pattern and values caused by several types of human actions. The force due to each person in a group of n persons is of the form $\{F(t)\}$ where $F_i(t)$ represents a single force and $i = 1, 2, 3, \ldots, n$. The parameters needed to generate these probability-based-functions will be determined utilizing measured values. Data from tests conducted by the author are supported with data from earlier stages of the research. The force functions produced will represent the dynamic load of an individual in a crowd of people who are also active. With accurate correlations, the entire group action will then be modeled with a matrix of discrete forces.

In addition to the interest in having these forces derived and available, there are a number of ancillary objectives which will have benefits and future possibilities:

- The models will be especially important in cases where sustaining human loading is the primary purpose of the structure such as stadium or grandstand seating. In these cases the models would be valuable for design and analysis.
- The models may be used to determine limits or maxima; for example, for the verification of specific building code values which are based on avoidance of catastrophic failure. The code values used in the design of grandstand seating areas, such as for stadium design, will be derived and compared with published code values for verification.
- The models provide for the ability to conduct simulations to study any of a number of possible events. An example is the post-elastic response of a grandstand element subjected to a person jumping on it.
- The models are valuable for reliability-based design methods where the design input needs to be probabilistically presented.
- The load models may eventually have a major impact on standards for structural design such as those produced by ANSI, the American National Standards Institute, from which many building codes are derived.
- The models will be necessary for the design of active controls to minimize vibration of structures subjected to human movements. Accurate definition of

time-varying forces where humans are the primary source of the forces, such as for the design of stadiums, are essential for providing active damping of the structure.

- The models may be used to define parameters, such as acceleration, based on serviceability of the structure. Serviceability criteria should be a requirement for acceptable design.
- It is expected that the process evolved in developing the load models will provide a methodology for extending the list of models to additional human actions such as for aerobics, climbing or descending stairs, and the "heel drop" test.
- The models and the methodology should provide the insight needed to extend definitions of load functions to forces from actions due to persons moving such as dancing, walking or marching.

1.4 Research Description

The load modeling of human movements was accomplished through four tasks:

a) **Experimental measurements**: Force histories from tests performed on a small force platform and on a large floor system were used to statistically characterize dynamic loading due to human activities.

b) **Model identification**: Digitally recorded responses due to specific motions were reviewed to obtain all available information on how the force histories are generated and what factors affect the shape and magnitude of the generated force .

c) Load Modeling: The experimental data were analyzed and inferences were made regarding parameters needed to develop the load models. Modeling methodologies are presented along with detailed procedures for simulating the different human actions considered in the study.

d) Model verification : The proposed models were used to generate extensive load histories which were compared statistically to the measured test data. The intent of the process was to reveal inadequacies of the model and therefore, through iteration, improve the proposed model.

This dissertation is organized in the following parts:

Introduction (Chapter 1.0): It is a condensed outline of the topics herein, defined and more fully explained in the main body of the dissertation. It is intended as a broad mapping of areas covered by the research.

Current State of Knowledge (Chapter 2.0): A detailed review of the current state of knowledge in the field of modeling human loads is presented. Earlier development of the project of which this research is part of is outlined. The chapter also includes an overview of current practice in the design process and how the present U.S. codes characterize human loads on assembly structures.

Experimental Measurements (Chapter 3.0): An overview of procedures of data collection, the experimental setup, data manipulation, and data representations is presented. Computer programs were developed to automate the process of converting raw voltage data into time-force histories. Data available from earlier measurements were utilized as well as data from new tests.

Measurement Analysis (Chapter 4.0): For each human movement type chosen for the study, the forcing function model is hypothesized. The parameters needed to define each motion are evaluated and analyzed statistically.

Probabilistic Load Modeling (Chapter 5.0): The chapter describes an iterative model-building methodology whereby the stochastic models introduced in Chapter 4.0 are related to actual time series data. The process of model verification was applied to the different motions considered in the study. A model to simulate group loading is provided. The developed forcing functions are compared with experimental data. Feedback from this process is used to refine the forcing functions.

Model Applications (Chapter 6.0): The impact of the research findings on the current state of knowledge is described. Several Applications showing the use of the Human Load Simulation program, *HLS*, are developed from the study. Recommendations on handling dynamic human loads in the design stage are presented.

Conclusions (Chapter 7.0): A summary of the research along with the major findings and conclusions is presented. The chapter also includes recommendations for designers and code agencies for incorporating research findings in future reliability-based-codes. A list of tasks for future research is outlined.

2.0 CURRENT STATE OF KNOWLEDGE

2.1 Introduction:

In this chapter, a detailed review of the current state of knowledge in the field of modeling human loads is presented. First, an overview of the history of measuring force parameters of human loads is demonstrated. Then earlier development of the project of which this research is part of is outlined. Finally, the chapter outlines the current practice in both the design and analysis phases that relates to human load modeling.

2.2 Literature Review

Studies of human live loads, from before the turn of the century, have been reported in depth by Saul and Tuan (1986). Tilden (1913) is the first person reported to have measured the parameters of forces due to human movements; his work formed the basis for the parameters used in present day building codes for grandstands and similar places of assembly. Much of the intervening work focused on finding the statically equivalent code values used for design today (ASA 1941). A series of incidents involving the collapse of portable grandstands in the early 1920's led to tests conducted under the supervision of the American Standards Association (now American National Standards Institute) to measure horizontal forces exerted by spectator movements. Other code associations adopted the design values recommended by the American Standards Association Committee in 1941, and the values remain the same today. These tests focused on determining the upper bounds forces and neglected dynamic content.

Bachmann (1984) examined the hypothesis that resonance behavior in a structure can be avoided if its natural frequency is higher than that of the forcing frequency and discovered that the assumption is not always correct. He found that lightly damped structures may be forced into significant resonance if their natural frequency is an integer multiple of the forcing frequency. A case study of a gymnasium where vibrations with large amplitudes due to resonance occurred at twice the forcing frequency is discussed in detail in his paper. He proved that the second and third harmonic of the forcing frequency may produce significant dynamic loading.

Allen, et al (1975, 1982,1987) found that although structures in most cases were built to code, serviceability problems such as unacceptable vibrations due to occupants' movements occur. They related structural resonance to thirteen cases of serviceability problems caused by dancing, spectator activity at rock concerts, aerobics, exercising, and walking. Their studies were concerned with response of the structure and gave negligible treatment to the input function. Greimann and Klaiber (1978) generated a continuous forcing function to successfully model a resonance condition caused at Iowa State University stadium by spectators moving together on a grandstand.

Tuan and Saul (1985) focused upon measuring the loads produced by an individual performing prescribed maneuvers on a small force platform. The loads produced by an individual performing prescribed maneuvers measured on a small force platform were categorized and some statistical analyses were done on the data to produce tentative load functions. The research was continued by McDonald (1984) who designed and constructed a 4 by 8 foot force platform for measuring loads produced by up to five people. The platform was used to acquire data for actions due to one or more persons.

Ebrahimpour (1989), proposed group forcing functions to produce an equivalent vertical dynamic load to be compared to code presented values. A sinusoidal Fourier series was used to describe the total force developed by a symmetrically placed group of people jumping in unison at a fixed rate. VanKleek (1988), constructed and instrumented a 12 foot by 15 foot floor system to accommodate up to 40 people. The floor system was used to obtain experimental measurements and test the accuracy of the group load model proposed by Ebrahimpour. Statistical modeling was performed for periodic jumping conducted at frequencies of 2 and 3 Hz and also for a single jump. The results showed that for the specific conditions of the test, the equivalent maximum uniform floor load approached an asymptotic value comparable to the code value.

Rainer, et al (1988), investigated the dynamic loading and response of footbridges using a force platform and asking individuals to walk or run along the span at a predetermined pulse rate. They concluded that for walking, running, and jumping, the excitation force can be represented as F(t):

$$F(t) = P \left[1 + \sum_{n=1}^{N} \alpha_n \sin(2n\pi f t + \phi_n) \right] \qquad [2.1]$$

where:

P = static weight of a person; α = Fourier amplitude or coefficient;

n = harmonic order of footstep rate; = footstep rate in steps per second;

t = time in seconds; ϕ = relative phase angle; and

N = total number of harmonics.

Allen (1990) introduced some revisions to the Canadian Building Code design criteria

Table 2.1 Periodic actions*

Rhythmical motions	Affected structures
▪ Walking	▪ Footbridges
▪ Running	▪ Office buildings
▪ Jumping	▪ Gymnasia and sports halls
▪ Dancing	▪ Dance and concert halls with no fixed seating
▪ Hand clapping with body bouncing while standing ▪ Hand clapping only	▪ Concert halls with fixed seating as well as galleries
▪ Lateral body swaying while seated or while standing	▪ High-diving platforms in swimming pools

* After Bachmann (1992).

for floor structures to reduce building vibrations due to human activities. He studied a number of recent vibration problems in concrete building structures due to dancing and aerobics exercises and suggested the following design criteria for minimum natural frequency:

$$f_o \geq f \sqrt{1 + \frac{1.3}{a_o/g} \cdot \frac{\alpha \omega_p}{\omega}} \quad \ldots\ldots\ldots\ldots\ldots \text{[2.2]}$$

where:

= forcing frequency; $_o$ = natural frequency of the structures;

a_o/g = acceleration ratio; ϖ = mass weight; and

ϖ_p = Weight of the person. α = dynamic load factor

He recommended increasing the factor (1.3) to (2.0) in the case of aerobics activities.

Considering the problem of floor vibrations, Murray (1991) provided a methodology for controlling annoying floor movement in residential, office, commercial, and gymnasium type environments. His criteria states that the motion of floor systems would not be objectionable to occupants of residential and office environments if the following inequality is met:

$$D > 35\, A_o f + 2.5 \qquad [2.3]$$

where D = damping in percent of critical, A_0 = maximum initial amplitude of the floor system due to heel-drop excitation (in.) defined as the force by a person standing on the toes and dropping suddenly from that position, and = first natural frequency of the floor system (Hz). For commercial environments (such as shopping centers), the criteria is based on an acceleration tolerance limit of 0.005 g due to walking excitation so that maximum deflection under 450 lbs. force applied anywhere on the floor system does not exceed 0.02 in. Murray (1991) confirmed the usefulness of Eq. [2.2] proposed by Alan (1990), to be used for gymnasium environments.

Bachmann (1992) suggested the following function for the dynamic forces due to rhythmical body motions shown in Table 2.1:

$$F_p(t)=G+\Delta G_1\sin(2\pi f_p t)+\Delta G_2\sin(4\pi f_p t-\varphi_2)+\Delta G_3\sin(6\pi f_p t-\varphi_3)+........ \quad [2.4]$$

with:

$_p$ = fundamental frequency of excitation (frequency of the motion, i.e. frequency of walking, running, jumping, dancing, etc.), G = weight of person in motion, ΔG_i = part

Table 2.2 Representative activity and their applicability to actual structures*

Representative types of activity	activity rate	Hz	Fourier coeff. $\alpha_i = \Delta G_i / G$ and phase lag φ_i					Design density [person s/m²]
			α_1	α_2	φ_2	α_3	φ_3	
Walking	vertical	2.0	0.4	0.1	$\pi/2$	0.1	$\pi/2$	~1
		2.4	0.5					
	forward	2.0	0.5($\alpha_{1/2}$=0.1)	0.2				
	lateral	2.0	$\alpha_{1/2}$=0.1	$\alpha_{3/2}$=0.1				
Running		2.0-3.0	1.6	0.7		0.2		-
Jumping	normal	2.0	1.8	1.3	A	0.7	A	in fitness training ~0.25 (in extreme cases up to 0.5)
		3.0	1.7	1.1	A	0.5	A	A: $\varphi 2 = \varphi_3 = \pi(1 - f_p t_p)$
	high	2.0	1.9	1.6	A	1.1	A	
		3.0	1.8	1.3	A	0.8	A	
Dancing		2.0-3.0	0.5	0.15		0.1		~4(in extreme cases up to 6)
Hand clapping with body bouncing while standing		1.6	0.17	0.10		0.04		no fixed seating ~4 (in extreme cases up to 6) with fixed seating ~2 to 3
		2.4	0.38	0.12		0.02		
Hand clapping	normal	1.6	0.024	0.010		0.009		~ 2 to 3
		2.4	0.047	0.024		0.015		
	intensive	2.0	0.170	0.047		0.037		
Lateral body swaying	seated	0.6	$\alpha_{1/2}$=0.4	-		-		~ 3 to 4
	standing	0.6	$\alpha_{1/2}$=0.5	-		-		

*After Bachmann, (1992).

of force of the i-th harmonic with i=2,3,.... (Fourier amplitudes), φ_i = phase lag of the i-th harmonic relative to the first harmonic. Table 2.2 shows the characteristics of representative human load dynamic forces as published by a working group of the Comité Euro-International du Béton (CEB).

2.3 Current Project

The current field of research started in 1983 with a study to determine the sources and accuracy of building code prescribed load values for the design of stadiums. The research focused on live loads due to human movements on assembly structures such as grandstands, stadiums and bleachers. It was found that present codes are based on experience and tests conducted around 1930 to satisfy safety criteria, i.e., the avoidance of catastrophic failure, due to live loads. These loads are presently specified in various building codes as statically equivalent forces having a uniform vertical load over the surface plus three components along the seats (one vertical and two in the horizontal plane, one along the seats and the other perpendicular to the seats). Ignoring the dynamic effect of human loads on assembly structures has led to vibration and serviceability problems in some modern assembly structures. A fairly complete history of U.S. practice, code sources, and background on the issue has been compiled and published by Saul and Tuan (1986).

The loads produced by an individual performing prescribed maneuvers was measured on a small force platform, Tuan (1983) and Tuan and Saul (1985). The maneuvers were categorized and some statistical analyses were done on the resulting data to produce trial model load functions. The research was continued by McDonald (1984) who designed and constructed a 4 foot by 8 foot force platform for measuring loads produced by up to five people. The platform was used to acquire additional data for actions due to one or more persons. This data was used to develop a preliminary model to describe loads due to a crowd of people. Descriptive random parameters were defined for periodic jumping, periodic jouncing, and single jumps. A sinusoidal series was used to describe the total vertical component force developed by a symmetrically placed group of people jumping in unison at a fixed rate, Ebrahimpour (1987).

The objectives of the second phase of the study were to measure dynamic loads generated by large groups of people performing coherent movements and to test the accuracy of the analytical load model derived earlier in the project.

VanKleek (1988) constructed and instrumented a 12 foot by 15 foot floor system to accommodate up to 40 people. The stiffness, mass, and damping matrices of the floor system were determined. The floor system was used to obtain experimental measurements and test the accuracy of the group load model proposed by Ebrahimpour (1987). Load tests were conducted with groups of 10, 20, 30, and 40 participants. Statistical modeling was performed for periodic jumping conducted at frequencies of 2 and 3 Hz and also for a single jump. The load-time histories were determined and comparisons were made between computed and measured results.

The results showed that for the specific conditions of the test the equivalent maximum uniform floor load approached an asymptotic value comparable to the code value. Also, it was found that the average deviation between measured and simulated values was less than 8%, Ebrahimpour and Sack (1989).

The preceding research projects provided extensive data and led to the hypothesis that forces due to the various human actions may be accurately modeled as probabilistic functions. These functions would be very valuable analytical tools for further and extensive research on projects involving such forces.

2.4 Current Practice:

The 1982 Edition of the Uniform Building Code defines **Assembly Structures** as: "A building or a portion of a building used for the gathering together of 50 or more persons for such purpose as deliberation, education, instruction, worship, entertainment, amusement, drinking or dining or awaiting transportation", UBC (1982). The code suggests using a uniform static vertical live load of 120 pounds per linear foot of seating,

Table 2.3 Bounds for Natural Frequencies of Structures subject to Human Loads

Type of Structure	Construction material			
	Reinforced concrete	Prestressed Concrete	Composite steel-concrete	Steel
Footbridges	1.6 to 2.4 Hz and 3.5 to 4.5 Hz to be avoided			
Office buildings	> 4.8 Hz (ζ > 5%) and > 7.2 Hz (ζ < 5%)resp.			
Gymnasia and sports halls	> 7.5 Hz	> 8.0 Hz	> 8.5 Hz	> 9.0 Hz
Dance and concert halls without fixed seating	> 6.5 Hz	> 7.0 Hz	> 7.5 Hz	> 8.0 Hz
Concert halls and theaters with fixed seating (incl. pop concerts)	Vertical: > 6.5 Hz Horizontal: > 2.5 Hz			

lateral sway bracing loads of 24 pounds per foot parallel to the seating and 10 pounds per foot perpendicular to seat and footboards for grandstands, reviewing stands, and bleachers. It also recommends using a uniform vertical live load of 100 pounds per square foot for assembly structures with moveable seating.

A design criteria for assessing the acceptability of floor structures subjected to human periodic movements was introduced into the Commentary to the 1985 National Building Code of Canada.

Bachmann (1992) showed that rhythmical human body motions can induce considerable dynamic forces on assembly structures. Table 2.3 presents bounds for the natural frequency of structures subject to man-induced vibrations to reduce the probability of resonance when the forcing frequency approaches the natural frequency of a structure.

In general, recent studies to include the dynamic effect of human loads represents an advance in representing the effect of human loads. However, work was needed to derive load models that simulate different human movements as sole individuals or as individuals in a group. This research takes a fundamental look at the problem of modeling individual forcing functions and characterizing the correlation between individual forcing functions.

3.0 EXPERIMENTAL MEASUREMENTS

3.1 Experimental Devices

The goal of the experimental measurements was to accurately measure dynamic force components due to specified human activities. Data taken from tests performed on a force platform and a floor system were obtained from three sets of experiments, two

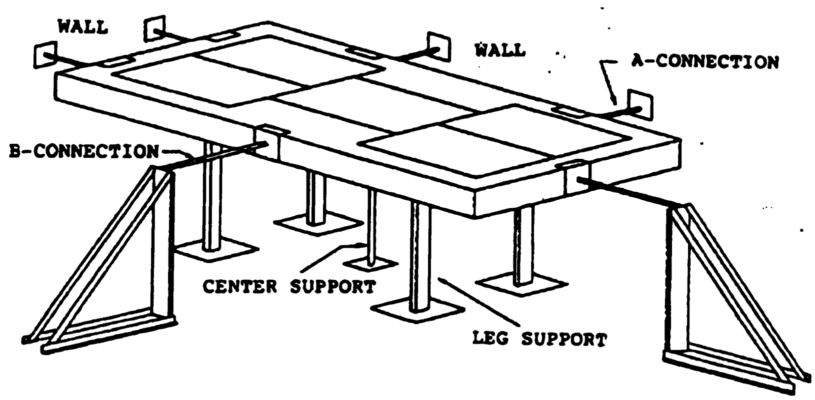

Figure 3.1 View of the Force Platform, Saul et. al. (1990)

designed to measure forces due to the action of an individual; and the third designed to measure the force due to the action of a group. Two different test devices were utilized. One was a 4 ft by 8 ft force platform designed and instrumented as described by McDonald (1984) and Ebrahimpour and Sack (1988) used for measuring the vertical and horizontal components of loads produced by individuals and small groups up to five participants, Figure 3.1. The platform has an approximate fundamental natural frequency

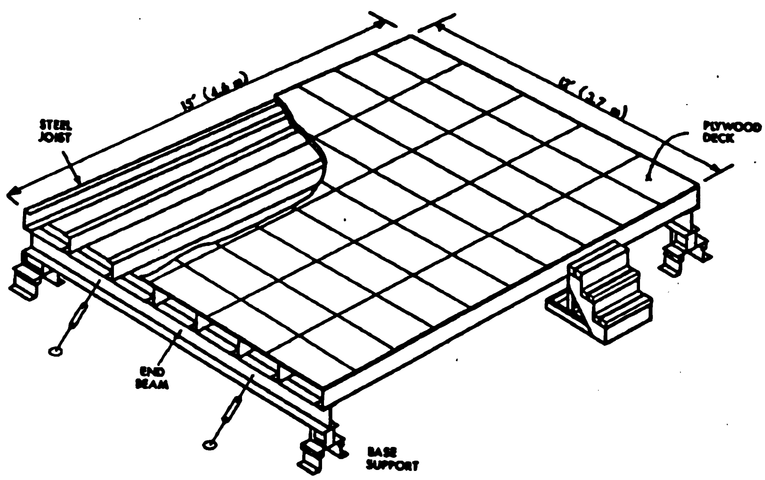

Figure 3.2 The Floor System Lab, Saul et.al. (1990)

of 20 Hz. which provides a flat transfer function so that imposed loads are transmitted to the sensors without distortion in the frequency range of the load; this is essentially the definition of a force platform. The platform is supported vertically by five transducer assemblies (load cells), and horizontally by three in each direction.

The other device, as shown in Figure 3.2, is a 12 ft by 15 ft floor system designed to measure total vertical loads generated by a group of people, up to forty participants, Ebrahimpour (1989). The floor system has a fundamental frequency of about 26.4 Hz and was instrumented with load cells and linear variable differential transformers (LVDTs) to measure the vertical component of human loads,Vankleek (1988). More detailed information on the data acquisition hardware is provided by Sack and Ebrahimpour (1988), and Burke et. al. (1985).

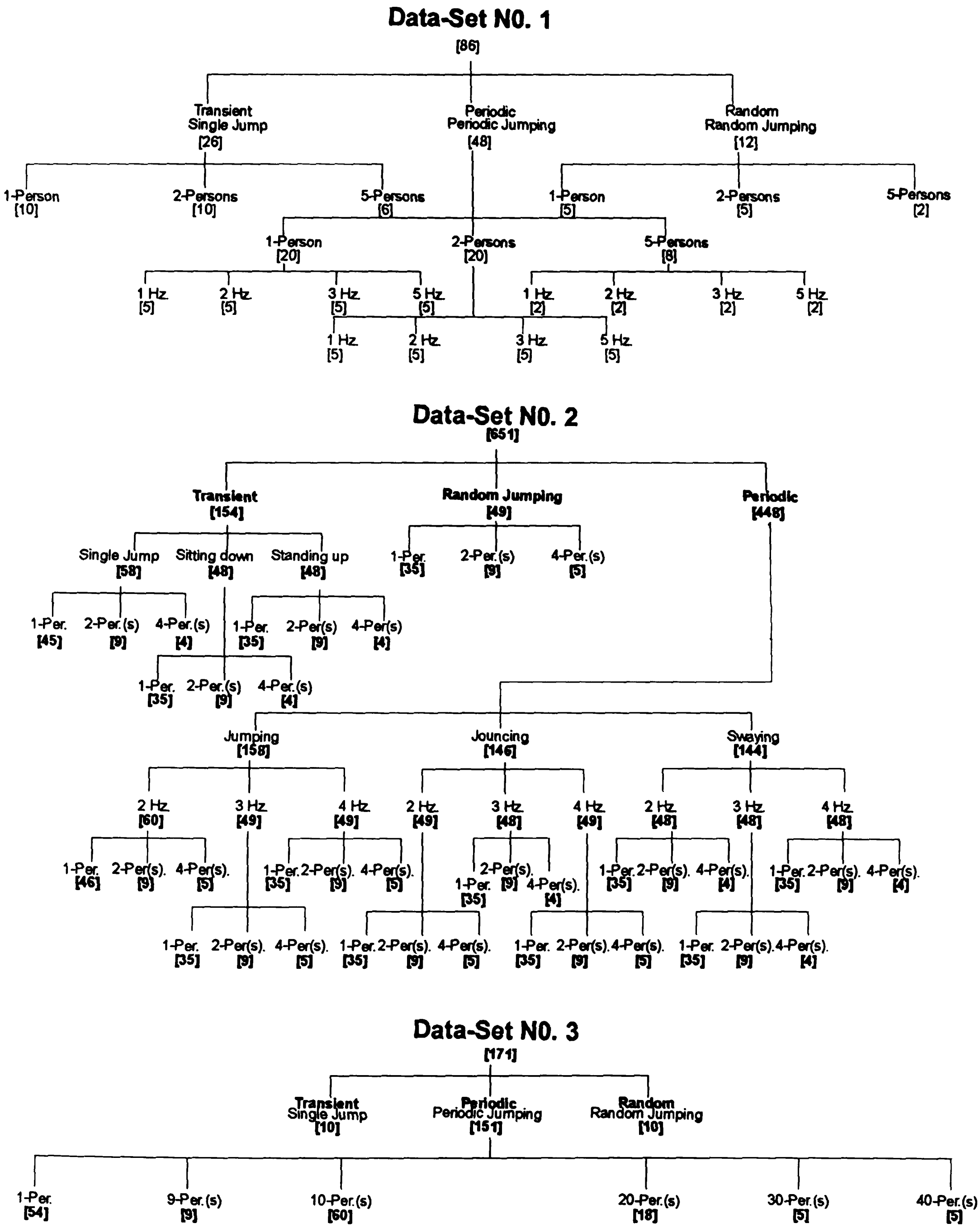

Figure 3.3 Human Load Experimental Data-base

In contrast to the force platform which by definition is designed so that input equals output, the floor system is modeled as a nine-degree of freedom vertical dynamic system where the output is read as vertical displacement of the floor.

3.2 Data Collection

Dynamic human loading can be categorized as transient, periodic, and random. Transient loads are temporary non-repetitive loads from which the structural response is damped out before the next transient action is expected to occur, McDonald (1984). Types of transient actions could include a single jump, suddenly standing up, or falling into a seat. Human load modeling in this study is limited to the transient and periodic motions.

Table 3.1 Transient Data-base

Action	Persons	Tests	Total
Single Jump	1	45	74
	2	19	
	4	4	
	5	6	
Sitting down	1	35	48
	2	9	
	4	4	
Standing up	1	35	48
	2	9	
	4	4	

As illustrated in Figure 3.3, large number of tests, were conducted on both the force platform and the floor system for individuals and groups of up to 40 participants. Prerecorded pulses (prompts) were played at the desired rate through loud speakers, and the participants were requested to perform a specific action at the pulse rate. The output from these tests were utilized to derive and verify the individual and group effect of human movements.. Table 3.1 summarizes the collected

transient actions by the type of action (single jump, sitting down, and standing up). In Table 3.2, the periodic actions are categorized by the stimulus frequency for the motions in the study (jumping, jouncing, and swaying).

Table 3.2 Periodic Actions Data-base

Frequency (Hz.)	No. of Persons	No. of tests			Total
		Jumping	Jouncing	Swaying	
1	1	5	0	0	5
	2	5	0	0	5
	5	2	0	0	2
	Total	**12**	**0**	**0**	**12**
2	1	114	35	35	184
	2	14	9	9	32
	4	5	5	4	14
	5	2	0	0	2
	9	9	0	0	9
	10	60	0	0	60
	20	18	0	0	18
	30	5	0	0	5
	40	5	0	0	5
	Total	**232**	**49**	**48**	**329**
3	1	40	35	35	110
	2	14	9	9	32
	4	5	4	4	13
	5	2	0	0	2
	Total	**61**	**48**	**48**	**157**
4	1	35	35	35	105
	2	9	9	9	27
	4	5	5	4	14
	Total	**49**	**49**	**48**	**146**
5	1	5	0	0	5
	2	5	0	0	5
	5	2	0	0	2
	Total	**12**	**0**	**0**	**12**

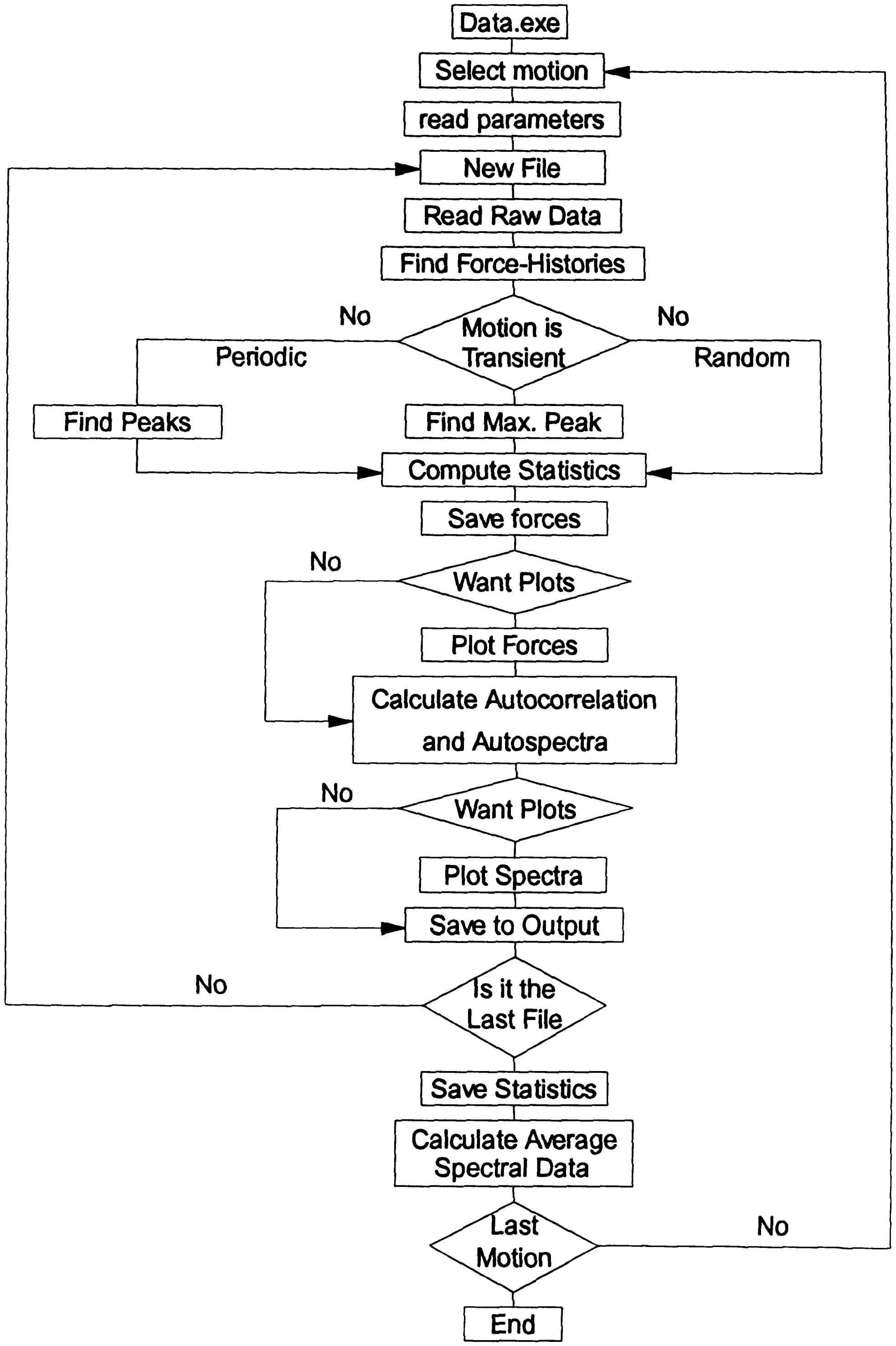

Figure 3.4 DATA.FOR Flow Chart

3.3 Data Processing

The raw data was collected in digital format as voltage readings. To use this data for the development of load models, a FORTRAN program, "DATA.FOR" was developed. For each motion type, the program reduces the transducer data into force-time series in the three principal axis of motion; vertical, and two perpendicular horizontal components Then the program calculates the statistical parameters of each force history, plots the forces in the three directions, computes and plots autocorrelation and autospectra for periodic motions, and calculates the average autocorrelation and average autospectra for force-time histories considered in the analysis. A listing of the program is included in Appendix A. Figure 3.4 is a flow chart to illustrate the components of the program "DATA.FOR".

Input to the data processing program consists of three sets of files; a) raw data files, b) processing files, and c) calibration files. The raw files contain the voltage readings and information about the type of test, weight of person or persons performing the experiment, and time and date of the test. The processing files include "PARAM.DAT" which is a collection of parameters needed to process the raw data and to compute the statistical and spectral information. Font files which were needed to plot graphs on the monitor are also a part of the processing files set. Calibration files hold the calibration factors needed to convert voltage into forces. These calibration factors were computed in an earlier stage of the research project and the process of obtaining these factors are documented in Ebrahimpour (1988), and VanKleek (1989).

The program saves the computed statistical information and spectral analyses in separate files which have the same file names as the raw data, but with different

Table 3.3 Typical Output Files from "DATA.FOR"

File	File Name	Extension	Reference
1	SJT1A010.FRC	Force	Forces in the x,y, and z directions
2	SJT1A010.ACN	Autocorrelation	Autocorrelation for x,y, and z directions
3	SJT1A010.ASP	Autospectra	Autospectra for the three directions
4	SJ-SUMM.OUT	Final output	Statistical information
5	SJ-AVG.ACN	Correlation	When more than one file processed
6	SJ-AVG.ASP	Spectra	When more than one file processed

extensions. For example, a raw data file name "SJT1A010.RAW" which contains data from a single jump (SJ) performed as a transient action by one person (T1), from data set (A), and was saved as test number (010) of the raw data collection (.RAW) would produce the files shown in Table 3.3 when processed by the program "DATA.FOR":

Output to the monitor is composed of force-time plots and spectral analysis plots (for periodic actions). Figures 3.5 and 3.6 are views of typical output from DATA.FOR. Maximum peak values in the three perpendicular axis and the weights of the participants are among the statistical information reported on the figures.

The vertical forces measured using the floor system were computed from transducers voltages using the Fortran program **"FLOOR.FOR"**. The listing of the program is provided in Appendix A. The program provided an effective and accurate method to process the test data gathered from the floor system. The procedure employed in the program to convert LVDTs voltage readings to vertical forces is explained in detail by

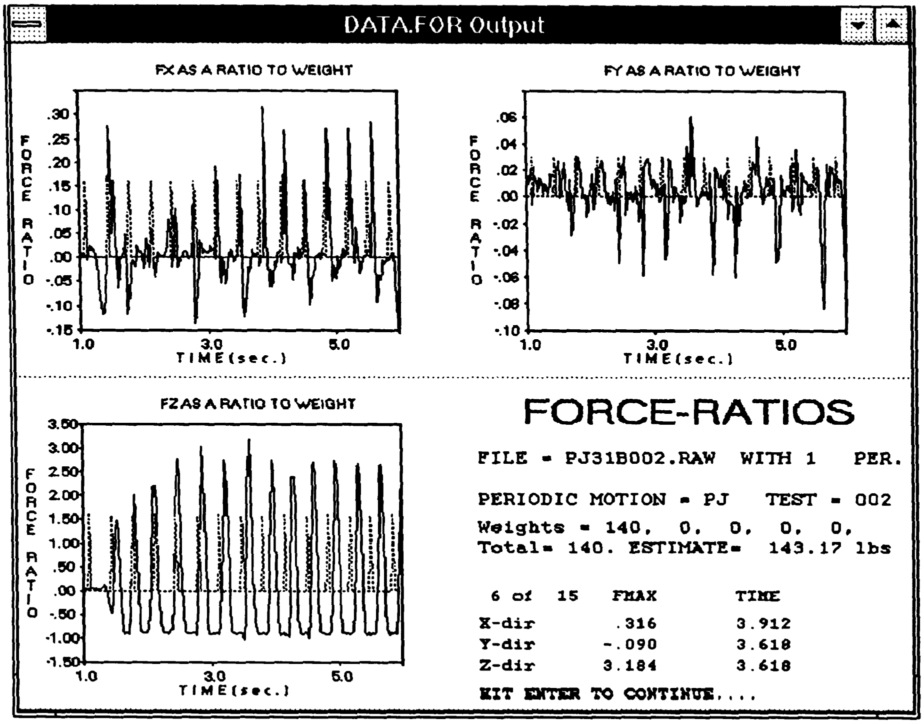

Figure 3.5 Force Type History (DATA.FOR Screen Output)

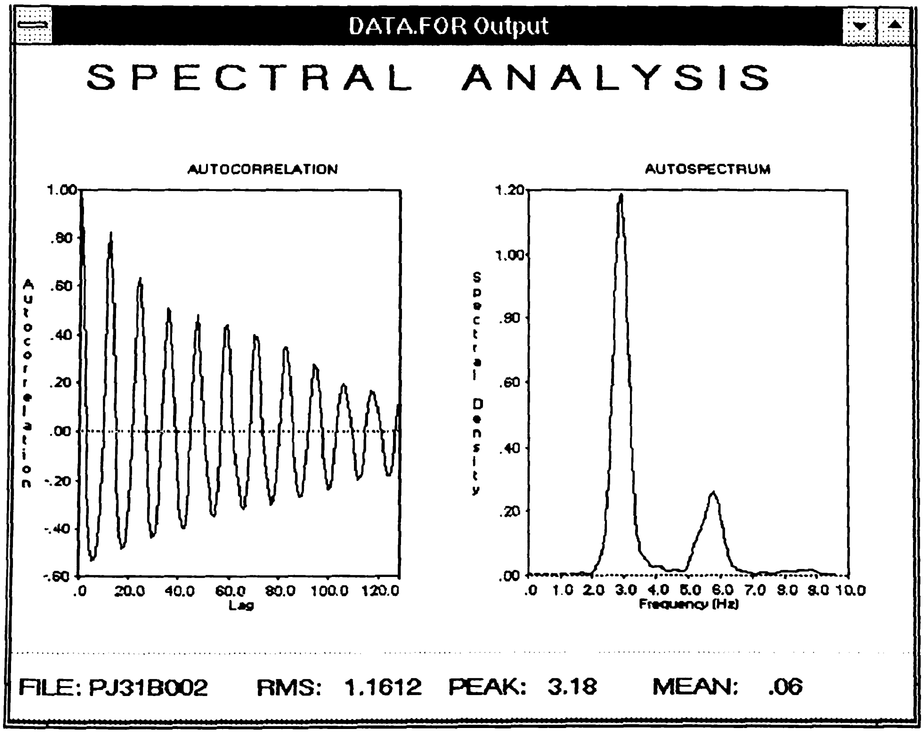

Figure 3.6 Spectral Analysis (DATA.FOR Screen Output)

VanKleek, (1988). The floor system was modeled as a nine degree-of-freedom system and the program FLOOR.FOR was used to solve the following equation of motion:

$$[\,F(\,t\,)\,] \;=\; [M]\,[A] \;+\; [C]\,[V] \;+\; [K]\,[x] \cdots\cdots\cdots \quad [3.1]$$

where :

F_{9x1} : applied nodal force vector

M_{9x9} : mass matrix A_{9x1} : acceleration vector

C_{9x9} : damping matrix V_{9x1} : velocity vector

K_{9x9} : stiffness matrix x_{9x1} : displacement vector

LVDTs Readings were converted from voltages to displacements using the proper calibration factors. Displacement histories were differentiated numerically to obtain velocity and acceleration. The vectors were used in Eq. [3.1] to calculate the force vector F_t. The total force-time history on the floor were computed by summing the force vector at every time t.

Input to FLOOR.FOR consisted of the raw data files, calibration matrix, and the predetermined mass, damping, and stiffness matrices. Output files included force-time history generated at each node location (LVDT location) on the floor, total force-time history, and a summary of statistics of force peaks. Figure 3.7 shows a typical plot of force-time output from FlOOR.FOR

3.4 Statistical Output Summary:

The force history, or shape of the force function with time, for each of the actions measured, proved to be consistent. Since each action appeared to result in a reproducible function, it was hypothesized that each function could be accurately modeled. Although each force history for the same action is similar in shape, each is statistically different; that is, the force functions vary within narrow bounds. As a result an accurate model would be probabilistic.

Statistical Results from the data processing program DATA.FOR output are stored on magnetic media for later use in the analysis. They are arranged by motion type (Transient, Periodic, and Random). For each motion type, output forces are normalized with respect to the participants' weights and referred to as ***"Dynamic Force Ratio"***. Absolute maximum dynamic force ratios and actual and estimated static weight are reported for an individual and for a group. Following are abbreviations that apply to values in these tables:

Fxmax : absolute maximum force ratio (to individual's weight) in the x- direction.

Var : sample variance

STD : sample standard deviation

C.O.V. : coefficient of variation

Table 3.4, a typical statistical table, shows the absolute maximum dynamic force ratios by one individual performing a single jump in-place. It can be seen from the table that force ratios in the vertical direction are significantly more than the other two components in the horizontal direction. The table shows also that an individual would generate an average vertical force of about ***3.19*** times his weight when performing a single jump. The minimum vertical dynamic force ratio observed in this collection of tests was ***1.54***. Thirty five males and fourteen females participated in this experiment.

Their average weight was ***161.36*** lbs. with a standard deviation of ***30.80*** lbs. and weight range from ***103*** to ***242*** lbs.

Table 3.4 Output summary of Single Jump (One-Individual)

Test	Weight	Fxmax	Fymax	Fzmax
1	165	1.823	0.274	2.173
2	172	1.096	0.483	2.552
3	162	0.713	0.227	1.538
4	181	0.594	0.497	2.378
5	175	1.166	0.728	4.241
6	165	0.499	0.156	2.685
7	172	1.448	0.379	4.357
8	162	0.816	0.437	2.038
9	181	1.129	0.286	3.242
10	176	1.615	0.634	3.858
11	196	0.446	0.131	3.491
12	140	0.403	0.069	2.622
13	186	0.426	0.296	3.43
14	150	0.507	0.196	4.602
15	189	0.63	0.383	5.035
16	163	0.998	1.492	1.946
17	125	0.729	0.238	5.037
18	234	0.505	0.032	2.117
19	132	0.809	0.406	5.941
20	130	1.05	0.101	2.428
21	146	0.222	0.086	2.473
22	242	0.822	0.12	3.487
23	184	0.519	0.243	4.047
24	155	1.792	0.157	3.117
25	200	0.415	0.094	4.472
26	191	0.254	0.095	3.823
27	166	1.098	0.085	2.13
28	215	0.615	0.242	2.841
29	154	0.524	0.101	4.236
30	153	0.604	0.166	2.969
31	120	0.31	0.117	1.851
32	103	0.553	0.124	2.796
33	126	0.277	0.045	1.563
34	116	1.649	0.285	1.936
35	185	1.823	0.127	4.269
36	145	0.522	0.069	2.025
37	140	0.531	0.232	2.571
38	170	0.299	0.054	3.056
39	120	0.657	0.255	3.539
40	171	1.047	0.101	1.952
41	177	0.284	0.09	2.722
42	124	0.629	0.08	2.131
43	128	0.618	0.109	1.889
44	110	0.398	0.059	3.038
Mean	161.36	0.77	0.24	3.19
Min	103.00	0.22	0.03	1.54
Max	242.00	1.82	1.49	5.94
Var	948.85	0.19	0.06	1.80
STD	30.80	0.44	0.25	1.34
C.O.V.	19.09%	57.44%	102.30%	42.09%

For each motion type, a table summarizing the statistical information of individual and group force ratios is prepared for later analysis. Table 3.5, for single jump motion,

Table 3.5 Maximum Force Ratios (Single Jump)

No. of Per.	Fxmax				Fymax				Fzmax			
	1	2	4	5	1	2	4	5	1	2	4.	5
Mean	0.77	0.60	0.37	0.42	0.24	0.28	0.21	0.29	3.19	2.30	1.66	1.32
Min	0.22	0.10	0.21	0.32	0.03	0.04	0.09	0.25	1.54	1.32	1.27	0.88
Max	1.82	1.24	0.65	0.52	1.49	0.73	0.31	0.36	8.85	3.62	1.89	1.89
Var	0.19	0.11	0.03	0.01	0.06	0.03	0.01	0.00	1.80	0.42	0.06	0.10
STD	0.44	0.33	0.17	0.07	0.25	0.18	0.09	0.04	1.34	0.65	0.24	0.32
COV	57.44	54.81	45.79	17.35	102.3	63.78	44.41	13.35	42.09	28.21	14.52	24.42

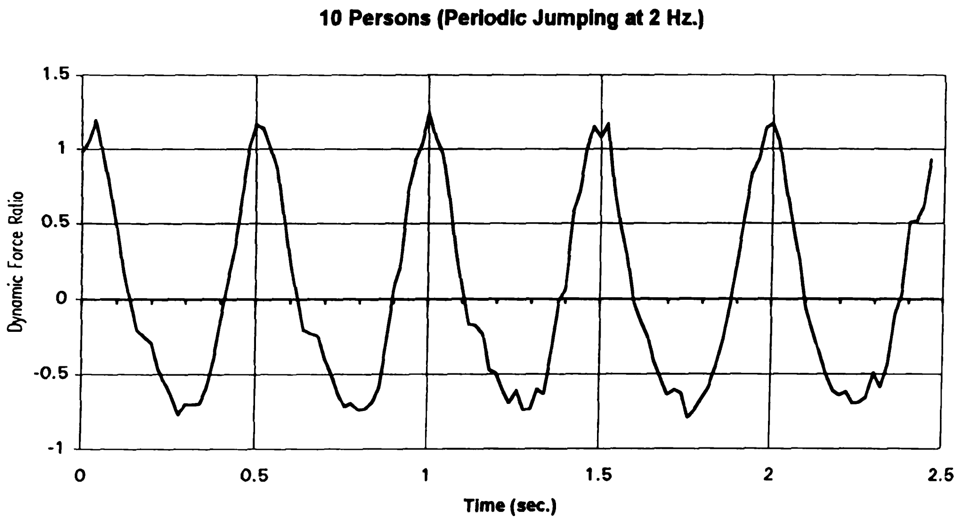

Figure 3.7 A Typical Force-Time History (Floor.For Output)

presents a typical sample of such tables. It shows that an increase in the number of people performing the same action at the same prompt tends to decrease the overall dynamic force. A single jump exemplifies this phenomena since it is difficult for typical individuals to maintain coherency in a pulse action like a single jump.

Tests performed on the floor system were analyzed for peak statistics using FLOOR.FOR. Some of the statistics are tabulated in Table 3.6. Data collected from the floor system are limited in two aspects; only the vertical component of the dynamic force

Table 3.6 Peak Statistics for Floor System Data

Group Size	Group No.	Weight	Avg. weight	Mean	Min.	Max.
10	1	1908	190.80	1.71	1.12	2.41
	2	1711	171.10	2.02	0.99	2.83
	3	1763	176.30	2.45	1.89	4.01
	4	1726	172.60	2.22	1.89	2.50
	5	1812	181.20	2.44	1.65	3.73
20	1	3578	178.90	1.52	1.20	1.88
	2	3404	170.20	1.46	0.61	1.97
30	1	5012	167.07	0.76	0.65	0.86
40	2	6505	162.63	0.68	0.46	1.04

was recorded; and data collection is limited to only periodic jumping at 2Hz. It was observed that individuals could maintain better coherency at this frequency range. Data collected from the floor system were used in this study to verify a load model hypothesized to simulate a large crowd.

Table 3.7 shows that the larger the group that performs the same action at the same time, the harder it is for individuals to maintain coherency. The maximum dynamic force ratio decreased from 4.54 for one persons to 1.14 for forty persons performing periodic jumping at 2 Hz. Figure 3.8 shows the group dynamic ratios for groups of individuals up to forty. The sample size of the four participants was the smallest among all other groups. Therefore, the standard error of the mean for that particular group shows a wider range than all other groups. The figure also suggests that the relationship between the number of participants and the dynamic force ratios may not be linear.

Table 3.7 2 Hz.-Periodic Jumping Peak Analysis

Force Ratio	Group Size *n*						
	1	2	4	10	20	30	40
Avg. Weight	159	162	170	178	175	167	163
Mean	2.86	2.75	2.71	2.17	1.49	1.44	0.79
Min.	1.51	1.57	2.45	1.51	0.91	1.18	0.53
Max.	4.54	3.89	3.05	3.10	1.93	1.64	1.14

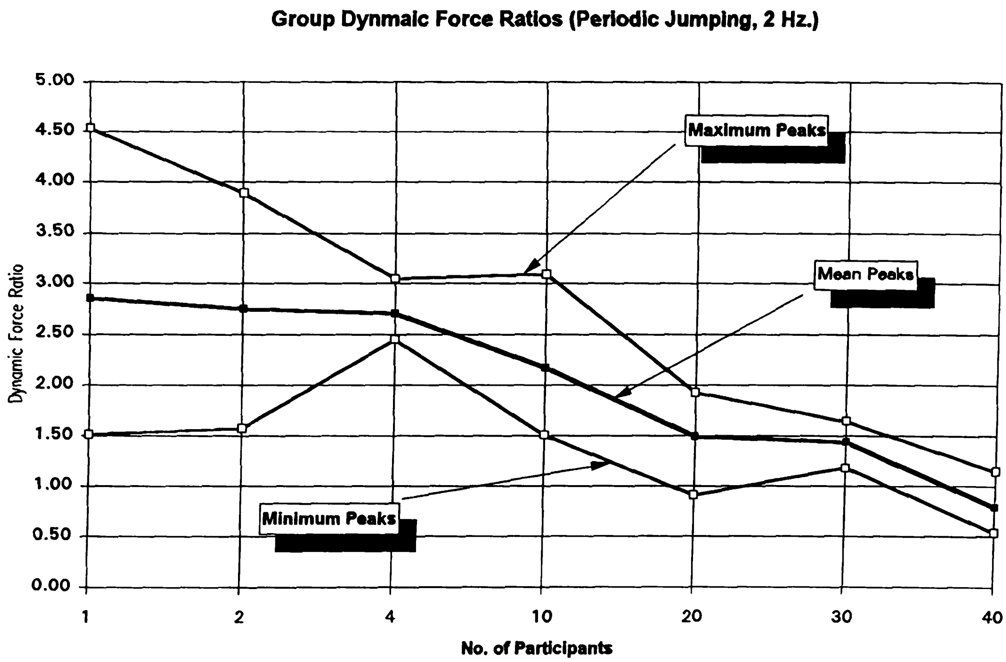

Figure 3.8 Mean, Min, Max Peaks For 2 Hz. Periodic Jumping

Since periodic jumping at 2 Hz. was found to produce the maximum coherence among participants, it would constitute an upper limit of dynamic load ratios of one

individual in group. A non-linear regression model is proposed to estimate "Dynamic Group Factor" ***DF***; an empirical factor to generate dynamic force ratios as function of group size n . The model is in the form :

where regression coefficients ***b, m*** are estimated by transforming Eq. [3.2] to a linear form and solve the linear regression, Chapra (1988):

$$Log\ (\ DF\ )\ =\ n\ Log\ (\ m\ )\ +\ Log\ (\ b\ ) \cdots\cdots\cdots\cdots \quad [3.3]$$

$$DF\ =\ b * m^{n} \cdots\cdots\cdots\cdots\cdots\cdots \quad [3.2]$$

The regression model was solved to estimate the regression coefficient for mean, minimum, and maximum dynamic factors. Figures 3.9 presents the regression model in comparison to the measured values. The figure also shows that the range between the

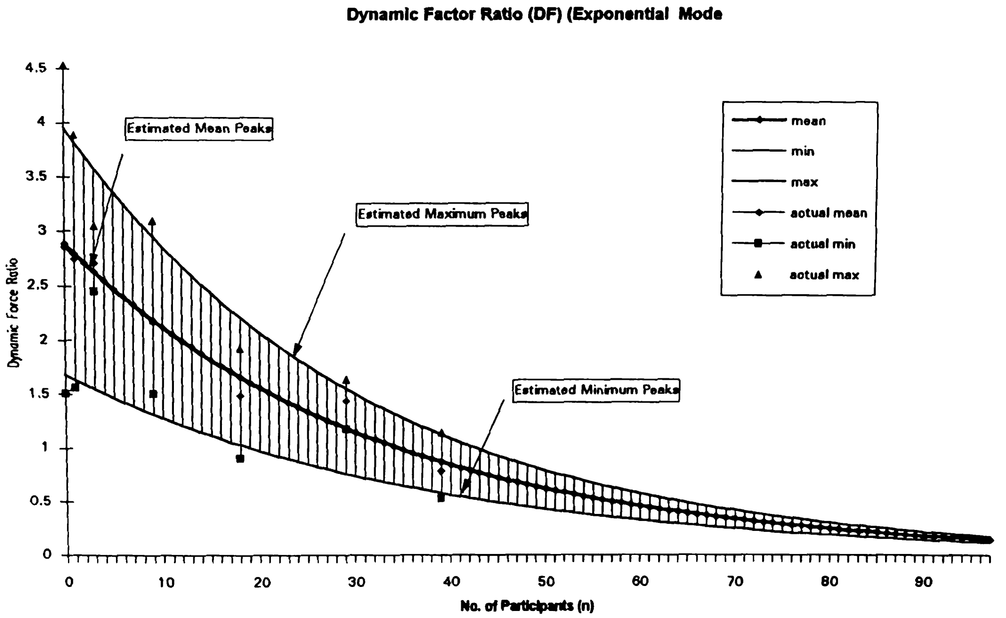

Figure 3.9 Log-Linear Regression Model

maximum and minimum dynamic group factors decreases as the crowd size increases. This formulation could be used as a tool to estimate an upper bound for the mean and range of the dynamic force ratios of any group size ***n***. The proposed Log-linear

Table 3.8 Human Movements Considered in The Study

Action	Motion	Description
Transient	Single Jump	A single jump in place from a standing position
	Standing Up	Standing up from a sitting position
	Sitting Down	Dropping into a seat from a standing position
Periodic	Jumping	Periodic jumping in place
	Jouncing	moving up and down without leaving the floor
	Swaying	Swaying side to side from a sitting position

regression can be simplified to take the form:

$$DF = \alpha * B^{n} \qquad [3.4]$$

where :

α is 2.98, 1.73, and 4.08 for the mean, minimum, and maximum Dynamic Group factors and B is a constant factor equals to 0.97.

Although several other regression forms were examined, the proposed regression in Eq. [3.4] proved to be the best-fit model. Nevertheless, it is important to emphasize the preliminary nature of the proposed regression. It is beyond the scope of this research to further investigate the theoretical aspects of the regression models in spite of their practical importance. Further statistical analysis could provide a better and more representative form to describe ***DF***.

4.0 MODEL ANALYSIS

4.1 General

In this part of the research, the focus is analyzing human loadings data to include and quantify their random nature. For each transient action in the study, a method based on passing straight lines through control points that define the force-time history is used to simulate human loading. Best-fit distributions for the control points were determined. Periodic loadings were analyzed as a series of impulses where the forcing function is divided into cycles, each of which is a full period of the motion considered. Statistics of each impulse in the force-time history were computed and analyzed for correlation and peak distribution.

4.2 Methodology

The dynamic force resulting from a single activity, such as a jump, is a time varying function $F_i(t)$ graphically represented as a single pulse with force changing with time over a relatively short interval, Fig. 4.1.a. This dynamic load will vary somewhat from individual to individual and from one time to the next for the same individual. In addition, when two or more individuals attempt the same action at the same time, the resulting loads will vary somewhat from one another but may be influenced by each other, Figure 4.1.b. The function $F_i(t)$ due to the action of a single person i may be a pulse or a continuing response. Table 4.1 summarizes motions considered in the study:

Table 4.1 Human Movements Considered in The Study

Action	Motion	Description
Transient	Single Jump	A single jump in place from a standing position
	Standing Up	Standing up from a sitting position
	Sitting Down	Dropping into a seat from a standing position
Periodic	Jumping	Periodic jumping in place
	Jouncing	moving up and down without leaving the floor
	Swaying	Swaying side to side from a sitting position

For each movement type, the forcing function is assumed to be a function of several independent random variables such as time, response lag, and group effect. The parameters of the function due to an individual performing a predefined human movement can be represented as $F_i(t)$:

$$F_i(t)_z = w\,[\,1 + f_z\,(\,t,\ \phi, f\,)] \qquad [4.1]$$

$$F_i(t)_x = w\,f_x\,(\,t,\ \phi, f\,) \qquad [4.2]$$

$$F_i(t)_y = w\,f_y\,(\,t,\ \phi, f\,) \qquad [4.3]$$

where the random variables are:

w : Weight of individual

t : Time in seconds

ϕ : Response Lag (Time lag between the prompt and the individual reaction)

f : Frequency of motion (not a factor in transient actions)

The dynamic components of the forces due to one person is approximated by passing line segments through control points defined as shown in Figure 4.2 for the case of a single jump, Tuan (1985). Periodic motion is modeled as a series of impulses, where the shape of each impulse is defined by passing line segments through control points as shown in Figure 4.3. The mean and the variance of arrival time and amplitude is determined for each control point. The probability density functions for the best-fit distribution is evaluated for arrival times and amplitudes of control points.

To simplify the modeling process, several assumptions were made. They are:

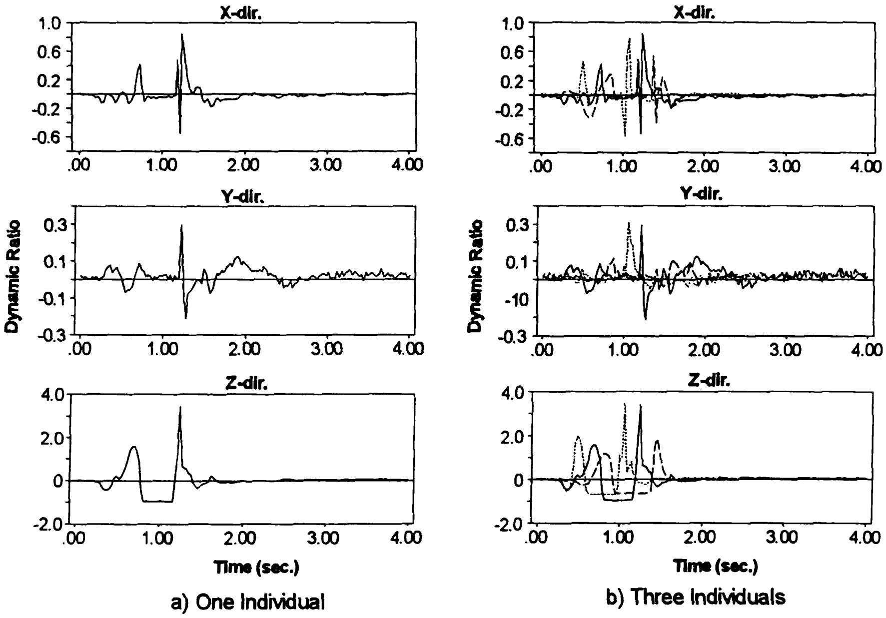

Figure 4.1 A Single Jump

- Initial computations are confined to the vertical component of the dynamic force.

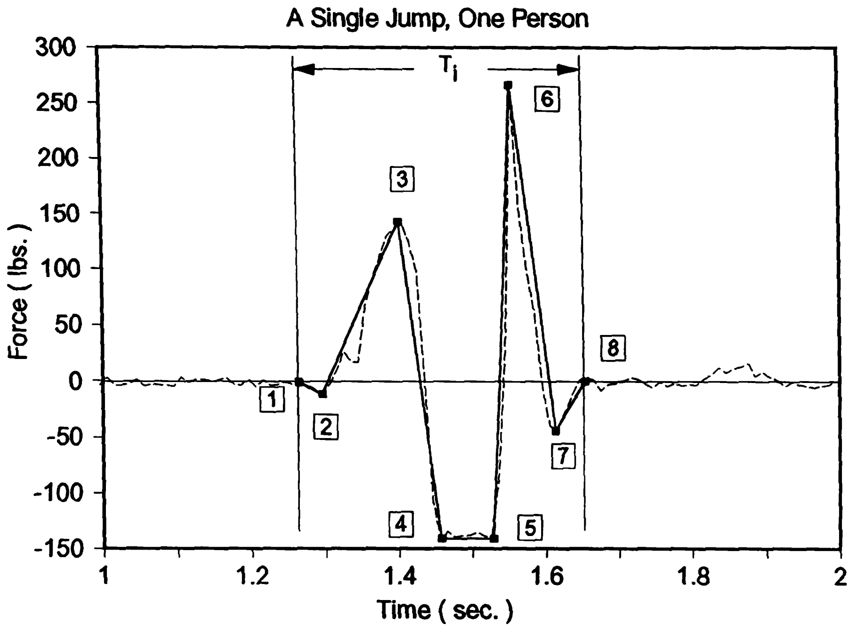

Figure 4.2 Control Points for A Single Jump

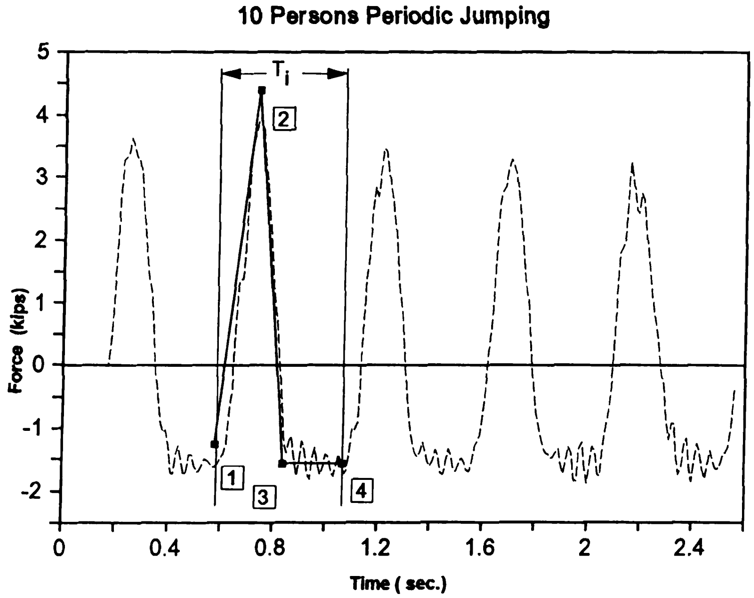

Figure 4.3 Control Points for a Periodic Motion

- Random processes used in the analyses were assumed to be adequately characterized by second-order models (models that can be sufficiently defined with the mean and the standard deviation).
- To account for participants weight variabilities, a general model based on the statistics of the weight distribution of the U.S. population were assumed for the study.
- For periodic motions, A model assuming that the person is always trying to maintain coherency with the prompt was assumed.

4.3 Transient Loading

The study includes single jumps, standing up from a sitting position, and sitting down. A typical transient forcing function $F_i(t)$ has the following random variables:

Response Lag ϕ **:** A random variable that represents the time between the prompt and the response of an individual. Available data were utilized to find the probability distribution function of the Response lag, and to determine whether it depends on the type of the transient motion.

Amplitudes $a_{1i}, a_{2i}, \ldots, a_{ni}$**:** Amplitudes of the control points that best-represent the transient function considered were determined. Covariance matrix and correlation with individual's weight were computed and analyzed.

Arrival Times $t_{1i}, t_{2i}, \ldots, t_{ni}$**:** Similar to the amplitude coefficients, statistics on arrival times were computed. The first arrival time t_{1i} for an individual i, is always assumed to be equal to the response lag ϕ assumed for the individual.

4.3.1 Single Jump

The single jump motion is used to explain in details the process of analyzing the measured data. Other types of transient actions considered in this study (Standing up, and Sitting down) are analyzed similarly. Differences in the analyses process are explained later in this Chapter.

For each measured force-time history, as a stochastic process, the weight of the person performing the test was removed from the process by subtracting it from all

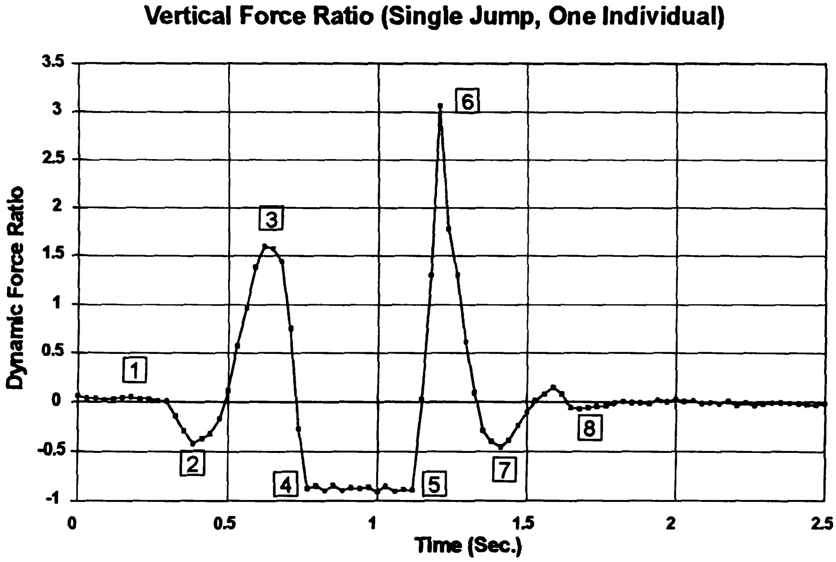

Figure 4.4 A Typical Single Jump

force-time series points. The process was normalized by dividing each reading by the weight of the participant. Figure 4.4 shows a typical vertical component of a single jump.

Eight control points were assumed sufficient to describe the vertical component of the forcing function, as shown in Figure 4.4. The prompt marks the beginning of the process. The first control point denotes the start of the individual's motion. The time lag from the beginning of the process to the first control point is defined as the Response Lag ϕ. The individual is in a static position at this portion. The process consists of three segments; first is the preparation for the jump, from control point 1 to 4; then airborne time between control point 4 and 5; and finally the landing portion where control point 6 marks the peak landing. The process ends at control point 8 when the person returns to the static position, Figure 4.4.

A program, CONTROL.FOR was developed to automate the process of analyzing the measured data. Input to the program consists of a parameter file and the collection samples of force-time histories for the chosen action. CONTROL.FOR proceeds as follows.

- Reduce the force-time history to 2.5 seconds after the prompt. It has been observed that 2.5 seconds is well beyond the duration of any action period considered herein. Force-time histories were aligned such that the prompt marks the beginning of the file.
- Differentiate the reduced function with respect to time to compute the slope function. The slope function is used to find the control points.
- Find control points using two criteria. First, the change in the slope direction will define peak control points 2,3,6, and 7. Control points 1,4,5, and 8 are defined as points where the slope values changes significantly. The program compares the

slope before and after each point and determines whether it exceeds a preset range value. The range value establishes the second criteria.

- The response lag ϕ is determined as the time from the prompt to the first control point. The response lag is then removed from the process and treated as an independent random variable. This assumption is verified later in the analysis. Arrival times are then adjusted so that actual dynamic motion starts at time $t_1=0.0$.
- Reduced force-time history data along with amplitudes and arrival times of the control points are saved in a file which has the same name as the input force-time history with an extension ".CRL",
- The Subroutine PLOTXY, is then called to plot the reduced function and control points on the same axes to visually ensure that control points were properly captured. Arrival times and amplitudes of control points are also shown on the screen.
- The program repeats the previous steps for all samples of force-time histories in the selected human motion (a single jump in this case).
- An output file (SJ-CRL.OUT) saves arrival times, amplitudes of control points, and peak values in the three principal axes from each sample in the collection of files processed by the program, Table 4.2.
- Subroutine MYSTAT is called to compute the important statistics of control points amplitudes and arrival times, Table 4.3. The computed statistics are saved in the output file.

Table 4.2 Control Points for Single Jump

No	WT	ϕ	T1	T3	T4	T5	T6	T7	T8	A1	A2	A3	A4	A5	A6	A7	A8
1	165	0.18	0.12	0.32	0.44	0.85	1.00	1.29	1.62	-0.10	-0.64	2.17	-1.00	-1.00	1.41	-0.45	0.00
2	172	0.15	0.15	0.41	0.56	0.97	1.03	1.41	1.53	-0.04	-0.68	1.92	-0.97	-0.98	2.55	-0.57	0.09
3	162	0.21	0.09	0.41	0.59	1.06	1.15	1.79	2.00	-0.18	-0.54	1.54	-1.03	-1.01	1.44	-0.45	0.02
4	181	0.29	0.09	0.35	0.44	0.88	0.97	1.27	1.38	-0.15	-0.45	2.24	-0.95	-0.92	2.38	-0.51	0.21
5	175	0.15	0.00	0.21	0.35	0.82	0.94	1.38	1.62	-0.05	-0.05	2.07	-0.95	-0.97	4.24	-0.20	0.03
6	165	0.21	0.12	0.27	0.38	0.74	0.79	1.18	1.35	-0.06	-0.74	2.42	-1.01	-1.00	2.69	-0.53	0.02
7	172	0.15	0.12	0.32	0.44	0.85	0.91	1.24	1.59	-0.04	-0.98	2.25	-1.01	-1.01	4.36	-0.49	-0.03
8	162	0.18	0.15	0.41	0.68	1.15	1.24	1.91	2.09	-0.08	-0.52	1.29	-1.00	-1.01	2.04	-0.51	0.10
9	181	0.32	0.09	0.32	0.44	0.79	0.88	1.09	1.24	-0.07	-0.62	1.86	-1.00	-1.01	3.24	-0.71	0.07
10	176	0.27	0.00	0.09	0.18	0.53	0.59	0.85	1.53	-0.03	-0.03	2.56	-1.02	-1.07	3.86	-0.79	-0.25
11	196	0.27	0.09	0.47	0.59	0.97	1.06	1.29	1.47	-0.03	-0.47	1.61	-0.93	-0.97	3.49	-0.34	0.20
12	140	0.50	0.06	0.35	0.50	0.94	1.06	1.24	1.27	-0.23	-0.41	1.70	-0.92	-0.94	2.62	-0.11	0.03
13	186	0.53	0.09	0.65	0.77	1.32	1.38	1.82	1.94	-0.06	-0.38	1.41	-1.03	-0.97	3.43	-0.43	0.11
14	150	0.32	0.09	0.24	0.38	0.77	0.85	1.12	1.24	-0.08	-0.72	2.65	-0.89	-0.95	4.60	-0.32	0.03
15	189	0.35	0.12	0.38	0.53	1.00	1.09	1.27	1.44	-0.07	-0.65	2.06	-0.98	-0.99	5.04	-0.74	0.07
16	163	0.24	0.18	0.47	0.65	1.09	1.18	1.50	1.62	-0.01	-0.56	1.60	-0.91	-0.92	1.95	-0.45	-0.14
17	125	0.38	0.09	0.38	0.53	0.94	1.00	1.47	1.68	-0.07	-0.49	1.56	-0.89	-0.90	5.04	-0.18	0.03
18	234	0.35	0.09	0.29	0.47	0.65	0.77	0.97	1.06	-0.06	-0.29	1.46	-0.85	-0.88	2.12	-0.23	0.06
19	132	0.29	0.12	0.27	0.38	0.85	0.94	1.21	1.50	0.02	-0.72	2.93	-0.88	-0.87	5.94	-0.58	-0.12
20	130	0.24	0.12	0.27	0.41	0.71	0.82	1.00	1.18	-0.04	-0.60	2.13	-0.95	-0.94	2.43	-0.34	0.00
21	146	0.27	0.12	0.24	0.35	0.71	0.82	1.00	1.18	-0.01	-0.85	2.47	-0.90	-0.96	2.31	-0.28	0.19
22	242	0.18	0.12	0.44	0.56	1.03	1.09	1.56	1.71	-0.07	-0.43	1.31	-0.98	-0.98	3.49	-0.42	0.01
23	184	0.35	0.12	0.38	0.50	0.94	1.03	1.27	1.47	-0.06	-0.31	2.44	-0.98	-0.97	4.05	-0.53	-0.03
24	155	0.35	0.18	0.44	0.59	1.09	1.15	1.44	1.62	-0.05	-0.41	1.84	-0.96	-0.96	3.12	-0.69	0.06
25	200	0.21	0.18	0.47	0.59	1.00	1.09	1.29	1.50	-0.01	-0.36	1.86	-0.97	-1.02	4.47	-0.36	-0.04
26	191	0.21	0.09	0.21	0.35	0.56	0.62	0.82	1.03	-0.03	-0.70	1.86	-0.93	-0.93	3.82	-0.54	-0.09
27	166	0.35	0.09	0.27	0.41	0.71	0.79	1.06	1.18	-0.11	-0.49	2.08	-0.90	-0.95	2.13	-0.56	0.08
28	215	0.38	0.09	0.27	0.38	0.79	0.85	1.12	1.24	-0.01	-0.54	2.27	-0.90	-0.87	2.84	-0.63	0.13
29	154	0.32	0.12	0.27	0.41	0.77	0.85	1.15	1.29	-0.04	-0.82	2.20	-0.98	-0.97	4.24	-0.53	0.07
30	153	0.21	0.12	0.35	0.50	0.97	1.06	1.29	1.44	-0.03	-0.62	1.99	-0.90	-0.90	2.97	-0.63	0.43
31	164	0.18	0.06	0.27	0.38	0.79	0.85	1.12	1.59	-0.10	-0.59	2.79	-1.01	-1.01	8.85	-0.40	0.01
32	120	0.29	0.09	0.24	0.38	0.56	0.62	0.77	0.79	-0.05	-0.42	1.29	-0.92	-0.99	1.85	-0.13	0.04
33	103	0.27	0.09	0.21	0.32	0.50	0.59	0.77	0.79	-0.05	-0.72	2.23	-0.88	-0.90	2.80	-0.15	0.01
34	126	0.27	0.00	0.06	0.18	0.32	0.47	0.74	0.85	-0.04	-0.04	1.34	-1.01	-1.10	1.56	-0.53	0.01
35	116	0.21	0.18	0.38	0.50	0.88	0.91	1.21	1.38	0.04	-0.41	1.94	-0.85	-0.86	1.77	-0.59	0.49
36	185	0.24	0.12	0.38	0.47	0.85	0.91	1.21	1.56	0.05	-0.38	2.07	-0.94	-0.96	4.27	-0.34	0.02
37	145	0.35	0.09	0.38	0.53	0.88	0.97	1.32	1.38	-0.11	-0.43	1.38	-0.82	-0.93	2.03	-0.08	0.01
38	140	0.24	0.06	0.18	0.27	0.50	0.59	0.85	1.00	-0.07	-0.57	2.57	-0.69	-0.86	2.23	-0.28	0.04
39	170	0.29	0.09	0.32	0.47	0.82	0.91	1.12	1.24	0.01	-0.42	1.60	-0.87	-0.89	3.06	-0.46	0.01
40	120	0.41	0.06	0.21	0.29	0.65	0.74	0.94	1.12	-0.22	-0.58	2.70	-0.83	-0.93	3.54	-0.69	0.21
41	171	0.27	0.09	0.27	0.38	0.68	0.77	1.06	1.35	0.03	-0.78	1.66	-0.86	-0.87	1.95	-0.32	0.20
42	177	0.24	0.12	0.27	0.41	0.65	0.74	0.94	1.09	-0.03	-0.49	1.65	-0.87	-0.91	2.72	-0.47	0.14
43	124	0.29	0.06	0.32	0.47	0.77	0.88	1.12	1.06	-0.21	-0.59	1.32	-0.93	-0.96	2.13	-0.37	-0.29
44	128	0.32	0.06	0.38	0.53	0.82	0.91	1.12	1.38	-0.16	-0.46	1.06	-0.79	-0.86	1.89	-0.05	0.02
45	110	0.24	0.12	0.29	0.41	0.71	0.79	1.06	1.24	-0.05	-0.41	1.71	-0.89	-0.94	3.04	-0.32	0.02

- All output files are closed and the program report the number of processed files and any errors during the run.

Output from the program is used next to study the correlation among control points and to find the best-fit distribution for arrival times and amplitudes of each control point.

Correlation Analysis

In order to better understand the relationship between the control points arrival times and amplitude ratios, a correlation analysis was conducted. The correlation coefficient *r* was calculated for all control points pairs and the results are tabulated in Table 4.4. The coefficient of correlation is a measure of the degree of linearity in the relationship between any two variables *X, Y*, Pollard, (1979). The correlation coefficient is calculated from the covariance matrix as follows:

$$cov\ (x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - x_m)(y_i - y_m) \cdot \cdots\cdots\cdots \quad [4.4]$$

$$r_{x,y} = \frac{cov\ (x, y)}{\sigma_x\ \sigma_y} \cdots\cdots\cdots\cdots\cdots\cdots \quad [4.5]$$

where ***cov (x , y)*** is the covariance matrix between variable *X* and variable *Y*, *N* is the number of samples, x_i is the value of the *X* variable from sample *i*, and x_m and σ_x are the mean and the standard deviation of *X*. The coefficient of correlation takes values from *-1* to *+1* where the negative values suggest negative correlation (high values of *x* are associated with low values of *y*) and positive values indicate positive correlation (high values of *x* are associated with high values of *y*). The perfect positive linear correlation will result in a value of *1* for the coefficient. An *r* value of zero does not imply

independence between x and y. It only indicates lack of any linear relationship between x and y.

In general, $100r^2$ gives the percentage of the total variation of the variable Y which is explained by, or is due to, the relationship to variable X, Freund (1980). It is however important to emphasize that r measures only the strength of the linear relationship and that high correlation values does not necessarily imply a cause-effect relationship. If the considered variables were drawn from a bivariate normal distribution, the correlation

Table 4.3 Single Jump Control Points Statistics

	WT	PHI	T1	T3	T4	T5	T6	T7	T8	A1	A2	A3	A4	A5	A6	A7	A8
AVG	161	0.28	0.10	0.32	0.45	0.82	0.90	1.19	1.37	-0.0	-0.5	1.93	-0.93	-0.9	3.16	-0.43	0.05
MIN	103	0.15	0.00	0.06	0.18	0.32	0.47	0.74	0.79	-0.2	-0.9	1.06	-1.03	-1.1	1.41	-0.79	-0.29
MAX	242	0.53	0.18	0.65	0.77	1.32	1.38	1.91	2.09	0.05	-0.0	2.93	-0.69	-0.8	8.85	-0.05	0.49
RNG	139	0.38	0.18	0.59	0.59	1.00	0.91	1.18	1.29	0.28	0.95	1.88	0.34	0.24	7.45	0.74	0.78
VAR	949	0.01	0.00	0.01	0.01	0.04	0.04	0.07	0.08	0.00	0.04	0.21	0.01	0.00	1.88	0.03	0.02
STD	31	0.09	0.04	0.11	0.12	0.19	0.19	0.26	0.29	0.06	0.20	0.46	0.07	0.06	1.37	0.18	0.13
COV	19	30.96	41.47	33.22	25.91	23.41	20.70	22.03	20.82	-100	-37.	23.74	-7.50	-5.7	43.4	-42.5	264.2
SER	5	0.01	0.01	0.02	0.02	0.03	0.03	0.04	0.04	0.01	0.03	0.07	0.01	0.01	0.21	0.03	0.02

coefficient can be further examined by using the ***Fisher Z transformation*** which estimates the confidence intervals for calculated coefficient of correlation. The test is based on a change of scale from r to Z which is given by:

$$Z = \frac{1}{2} Ln \frac{1 + r}{1 - r} \quad \text{[4.6]}$$

where the distribution of Z is approximately normal and a direct function of the true population coefficient of correlation ρ. The confidence interval for ρ is constructed from known statistical tables and the following $(1\text{-}\alpha)$ confidence interval:

$$Z - \frac{z_{\alpha/2}}{\sqrt{n-3}} < \mu_z < Z + \frac{z_{\alpha/2}}{\sqrt{n-3}} \qquad [4.7]$$

where μ_z is the mean of the transformation variable Z.

Table 4.4 Correlation Matrix (Single Jump)

VA	WT	PHI	T2	T3	T4	T5	T6	T7	T8	A1	A2	A3	A4	A5	A6	A7	A8	FXm	Fxm	FYm	Fym	FZm
WT	1.00	-0.0	0.11	0.31	0.28	0.31	0.30	0.28	0.33	0.23	0.13	-0.0	-0.3	-0.1	0.16	-0.2	-0.0	0.22	0.04	0.02	-0.1	0.15
PHI	-0.0	1.00	-0.2	0.17	0.14	0.12	0.13	-0.0	-0.1	-0.3	0.14	-0.0	0.17	0.25	-0.0	0.07	0.03	-0.3	0.09	-0.2	0.16	-0.0
T1	0.11	-0.2	1.00	0.58	0.59	0.52	0.50	0.39	0.29	0.36	-0.4	-0.0	-0.0	0.16	-0.0	-0.2	0.25	0.01	-0.1	-0.0	-0.0	-0.0
T3	0.31	0.17	0.58	1.00	0.97	0.92	0.90	0.80	0.65	-0.0	-0.0	-0.4	-0.2	0.07	-0.0	0.05	0.17	0.06	-0.0	-0.0	-0.1	-0.0
T4	0.28	0.14	0.59	0.97	1.00	0.92	0.91	0.84	0.66	-0.0	-0.0	-0.5	-0.2	0.06	-0.0	0.08	0.14	0.04	0.02	0.02	-0.1	-0.0
T5	0.31	0.12	0.52	0.92	0.92	1.00	0.99	0.93	0.83	-0.0	-0.0	-0.2	-0.3	-0.0	0.12	-0.1	0.15	0.27	-0.0	0.19	-0.2	0.12
T6	0.30	0.13	0.50	0.90	0.91	0.99	1.00	0.93	0.82	-0.1	-0.0	-0.2	-0.3	-0.0	0.08	-0.0	0.13	0.23	-0.0	0.18	-0.2	0.08
T7	0.28	-0.0	0.39	0.80	0.84	0.93	0.93	1.00	0.91	-0.1	0.01	-0.2	-0.3	-0.1	0.03	-0.0	0.10	0.33	-0.0	0.28	-0.2	0.03
T8	0.33	-0.1	0.29	0.65	0.66	0.83	0.82	0.91	1.00	0.00	0.05	-0.0	-0.4	-0.2	0.22	-0.1	0.01	0.52	-0.2	0.42	-0.3	0.23
A1	0.23	-0.3	0.36	-0.0	-0.0	-0.0	-0.1	-0.1	0.00	1.00	-0.0	0.12	0.00	0.11	0.14	-0.1	0.16	0.16	-0.1	0.03	-0.1	0.14
A2	0.13	0.14	-0.4	-0.0	-0.0	-0.0	-0.0	0.01	0.05	-0.0	1.00	-0.3	-0.0	-0.2	-0.1	0.06	-0.1	0.04	-0.1	0.09	-0.1	-0.1
A3	-0.0	-0.0	-0.0	-0.4	-0.5	-0.2	-0.2	-0.2	-0.0	0.12	-0.3	1.00	0.00	0.01	0.51	-0.3	0.01	0.28	-0.0	0.25	-0.1	0.54
A4	-0.3	0.17	-0.0	-0.2	-0.2	-0.3	-0.3	-0.3	-0.4	0.00	-0.0	0.00	1.00	0.81	-0.2	0.42	0.24	-0.3	0.07	-0.2	0.15	-0.2
A5	-0.1	0.25	0.16	0.07	0.06	-0.0	-0.0	-0.1	-0.2	0.11	-0.2	0.01	0.81	1.00	-0.1	0.29	0.34	-0.3	0.03	-0.2	0.06	-0.1
A6	0.16	-0.0	-0.0	-0.0	-0.0	0.12	0.08	0.03	0.22	0.14	-0.1	0.51	-0.2	-0.1	1.00	-0.1	-0.1	0.41	0.22	0.09	-0.0	1.00
A7	-0.2	0.07	-0.2	0.05	0.08	-0.1	-0.0	-0.0	-0.1	-0.1	0.06	-0.3	0.42	0.29	-0.1	1.00	-0.1	-0.3	0.22	-0.3	0.25	-0.1
A8	-0.0	0.03	0.25	0.17	0.14	0.15	0.13	0.10	0.01	0.16	-0.1	0.01	0.24	0.34	-0.1	-0.1	1.00	-0.1	0.01	-0.2	0.17	-0.1
FX	0.22	-0.3	0.01	0.06	0.04	0.27	0.23	0.33	0.52	0.16	0.04	0.28	-0.3	-0.3	0.41	-0.3	-0.1	1.00	-0.2	0.64	-0.4	0.40
FX	0.04	0.09	-0.1	-0.0	0.02	-0.0	-0.0	-0.0	-0.2	-0.1	-0.1	-0.0	0.07	0.03	0.22	0.22	0.01	-0.2	1.00	-0.2	0.29	0.19
FY	0.02	-0.2	-0.0	-0.0	0.02	0.19	0.18	0.28	0.42	0.03	0.09	0.25	-0.2	-0.2	0.09	-0.3	-0.2	0.64	-0.2	1.00	-0.7	0.10
FY	-0.1	0.16	-0.0	-0.1	-0.1	-0.2	-0.2	-0.2	-0.3	-0.1	-0.1	-0.1	0.15	0.06	-0.0	0.25	0.17	-0.4	0.29	-0.7	1.00	-0.0
FZ	0.15	-0.0	-0.0	-0.0	-0.0	0.12	0.08	0.03	0.23	0.14	-0.1	0.54	-0.2	-0.1	1.00	-0.1	-0.1	0.40	0.19	0.10	-0.0	1.00

The process could be explained by considering the correlation between the amplitude ratios of maximum force in the X-direction and maximum force in the Y-direction. The value of r from Table 4.4 suggests a positive correlation of ***0.64***. Assuming that both of the variables are normally distributed, the confidence interval for the true strength of the linear relationship between the two variables is estimated by first calculating the Z transformation of ***0.64*** from Eq. [4.6] which is be ***0.767***. Substituting the values, $n=45$,

and $z_{\alpha/2}$=***1.96*** (from the standard normal distribution tables at ***n=45*** and α=***0.05***) in Eq. [4.7], the ***0.95*** confidence interval is

$$Z - \frac{z_{0.05/2}}{\sqrt{45-3}} < \mu_z < Z + \frac{z_{0.05/2}}{\sqrt{45-3}} \qquad [4.8]$$

$$0.465 < \mu_z < 1.069 \qquad [4.9]$$

These two values are then interpolated from the Z Tables or from using Eq. [4.6] to get

$$0.434 < \rho < 0.789 \qquad [4.10]$$

So, it can be said that with a ***0.95*** confidence, the true linear relationship between the maximum force in the X-direction and the maximum force in the Y-direction lies between ***0.434*** and ***0.789*** with an estimate of ***0.64*** provided from ***45*** independent samples. This process has been implemented in this study to compute correlation coefficients and provide confidence intervals for each significant correlation.

Best-Fit Distribution

In the use of statistical distributions, observed data is used to estimate any parameter. The process should be the best possible to estimate the desired value. Maximum likelihood is one of the most widely used methods for estimating the parameters of probability distributions. The estimators derived by this method usually have the desirable properties of being consistent, unbiased, and efficient. The principle of this method is to select as an estimate of Θ the value for which the observed sample would have been most "likely" to occur. That is, if the likelihood of observing a given set of

observations is much higher when $\Theta = \Theta_1$ than when $\Theta = \Theta_2$, then it would seem reasonable to choose Θ_1 as an estimate of Θ rather than Θ_2.

A good estimator must have three properties, which are summarized using the following notation :

Θ = parameter being estimated

Θ' = best estimator of Θ

- Θ' must be a consistent estimator of Θ. This means that it is possible to take a sample size n large enough so that the absolute difference $|\Theta - \Theta'|$ is smaller than some small value ε .
- Θ' must be an unbiased estimator of Θ. To be unbiased, the expected value of Θ' must equal Θ.
- Θ' must be an efficient estimator of Θ. The variance of an efficient estimator Θ' must be smaller than the variance of any other estimator, Blank (1980).

The theoretical probability distributions considered for fitting the results of the control program are the normal, inverse Gaussian, and log-normal distributions. The difference between each probability distribution and the actual sample distribution is obtained at each sample point. The maximum difference (for a given theoretical distribution and over the entire sample) D_n, is the Kolomogorov-Smirnov statistic, denoted as the K-S statistic:

$$D_n = \max \mid F(x) - S_n(x) \mid \qquad \textbf{[4.11]}$$

where $S_n(x)$ is the empirical cumulative distribution function of a sample of a size n drawn from a population in which random variable X has a continuous cumulative distribution function $F(x)$, Gibra (1973).

The K-S statistic determines the level of probability at which it is possible to reject the null hypothesis that a random variable has a particular type of probability distribution, Eliashakoff (1983). The hypothesis is rejected if the observed value of the Statistic falls in the critical region at the desired level of significance which can be calculated as:

$$\left[\frac{1}{2n} Ln \left(\frac{2}{\alpha} \right) \right]^{1/2} \qquad [4.12]$$

The statistical model underlaying the test assumes the sample observations have zero probability of being equal (i.e. continuous distribution), Pollard (1979).

Table 4.5 shows the means of the control point coefficient and their corresponding standard deviations, and the K-S statistic values. By visual inspection and examining the K-S statistic values in table 4.5, probability distribution functions were selected as best-fit distribution for each control point random variable. Figure 4.5 shows a typical plot

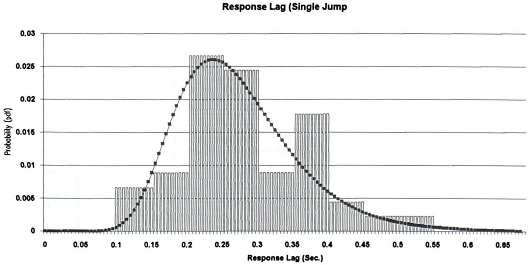

Figure 4.5 Best-Fit Distribution for Response Lag

of a data histogram and the best-fit distribution. The same process explained earlier was applied to all other random variables.

4.3.2 Sitting down

Seven control points were assumed to describe the vertical component of the forcing function as shown in Figure 4.6. The same process, explained earlier in the case of single jump motion, was used to find the control points and their statistics. The program

Table 4.5 Best-Fit Distributions (Single Jump)

VARIABLE	MEAN	STD	K-S NOR	K-S LOG	PDF
WT	161.356	31.152	0.071	0.1017	NORMAL
PHI	0.277	0.087	0.111	0.0812	LOG-NOR
T2	0.098	0.041	0.181	0.4438	NORMAL
T3	0.319	0.107	0.115	0.1448	NORMAL
T4	0.452	0.119	0.077	0.1269	NORMAL
T5	0.818	0.194	0.060	0.0876	NORMAL
T6	0.903	0.189	0.068	0.0825	NORMAL
T7	1.192	0.265	0.128	0.0907	LOG-NOR
T8	1.373	0.289	0.087	0.0826	LOG-NOR
A1	-0.068	0.057	0.194	0.0984	NORMAL
A2	-0.519	0.199	0.085	0.2500	NORMAL
A3	1.934	0.464	0.082	0.0676	LOG-NOR
A4	0.927	0.070	0.069	0.0805	NORMAL
A5	0.951	0.055	0.067	0.0687	NORMAL
A6	3.155	1.388	0.111	0.0761	LOG-NOR
A7	-0.427	0.184	0.060	0.1437	NORMAL
A8	0.094	0.108	0.194	0.0485	NORMAL
FXmax	0.530	0.287	0.160	0.0836	LOG-NOR
FXmin	0.618	0.493	0.202	0.0805	LOG-NOR
FYmax	0.170	0.145	0.193	0.1106	LOG-NOR
FYmin	0.208	0.251	0.230	0.0834	LOG-NOR
FZmax	3.189	1.357	0.121	0.0873	LOG-NOR

CONTROL.FOR was modified to analyze the sitting down motion. Arrival times, amplitudes of control points, and peak values were saved and analyzed for important statistics as shown in Table 4.6. The table shows that the average peak force in the

Table 4.6 Sitting Down Control Points Statistics

No	WT	PHI	T2	T3	T4	T5	T6	T7	A1	A2	A3	A4	A5	A6	A7	FXma	FXmi	FYma	FYmi	FZma
AVG	159	0.25	0.24	0.46	0.53	0.60	0.74	0.86	0.02	-0.7	0.83	0.86	2.11	-0.3	0.02	0.25	-0.2	0.07	-0.0	2.1
MIN	103	0	0.11	0.23	0.26	0.38	0.5	0.58	-0.0	-3.3	0.00	0.02	0.3	-1.2	-0.2	0.06	-1.0	0.00	-0.3	0.3
MAX	242	0.52	0.55	0.97	1	1.14	1.23	1.32	0.23	-0.1	4.89	4.89	8.11	-0.0	0.28	0.70	-0.0	0.30	-0.0	8.1
RNG	139	0.52	0.44	0.73	0.73	0.76	0.73	0.73	0.29	3.15	4.89	4.87	7.81	1.14	0.54	0.64	0.98	0.29	0.32	7.8
VAR	171	0.01	0.01	0.01	0.02	0.02	0.02	0.03	0.00	0.35	1.70	1.66	3.01	0.05	0.01	0.02	0.03	0.00	0.00	3.0
STD	34	0.10	0.10	0.13	0.15	0.15	0.16	0.17	0.06	0.59	1.30	1.29	1.73	0.22	0.13	0.15	0.18	0.07	0.06	1.7
COV	22	42.5	41.7	29.2	28.1	25.6	22.5	19.9	273.	-78.	157.	149.	82.0	-69.	489.	60.8	-69.	94.8	-81.	82.
SER	6	0.01	0.01	0.02	0.02	0.02	0.02	0.02	0.01	0.1	0.22	0.21	0.29	0.03	0.02	0.02	0.03	0.01	0.01	0.2

vertical direction for a person dropping to a seat was about ***2.11*** times his weight. Force ratio components in the orthogonal horizontal directions were ***0.26*** for the average force perpendicular to the seating and ***0.07*** for the average parallel force ratio. The table also reveals that the average time needed to perform the motion was ***0.86*** seconds with a delay after the prompt (response lag) of ***0.25*** seconds.

Linear correlation coefficients were calculated for all points pairs with the results tabulated in Table 4.7. Finding the best-fit probability distribution to represent each variable in the motion was accomplished by examining the K-S statistic values shown in Table 4.8. The best-fit distribution for the response lag was found to be the Log-Normal distribution. This is consistent with the finding in Table 4.5 for the case of a single jump. The table reveals that amplitudes can be best-represented by a Log-Normal Distribution.

Table 4.7 Correlation Matrix (Sitting Down)

VA	WT	PHI	T2	T3	T4	T5	T6	T7	A1	A2	A3	A4	A5	A6	A7	FXM	FXM	FYM	FYM	FZ
W	1	0.22	-0.05	-0.01	0.05	0.07	-0.02	-0.0	0.00	0.34	-0.36	-0.37	-0.17	0.27	0.28	-0.08	0.26	-0.33	0.40	-0.1
PH	0.22	1	-0.06	-0.07	-0.03	-0.01	-0.03	-0.0	-0.07	0.32	-0.35	-0.34	-0.23	-0.01	0.16	-0.10	0.26	-0.29	0.24	-0.2
T2	-0.05	-0.0	1	0.69	0.75	0.79	0.69	0.70	-0.20	0.16	0.10	0.10	0.25	0.02	-0.1	-0.05	-0.19	0.26	-0.24	0.25
T3	-0.01	-0.0	0.69	1	0.84	0.75	0.62	0.72	-0.01	0.04	0.48	0.48	0.55	-0.23	-0.1	0.31	-0.41	0.54	-0.32	0.55
T4	0.05	-0.0	0.75	0.84	1	0.96	0.80	0.83	-0.11	0.19	0.18	0.18	0.26	-0.08	-0.0	0.09	-0.21	0.29	-0.16	0.26
T5	0.07	-0.0	0.79	0.75	0.96	1	0.83	0.83	-0.14	0.25	0.04	0.05	0.13	0.00	-0.0	-0.00	-0.10	0.18	-0.1	0.13
T6	-0.02	-0.0	0.69	0.62	0.80	0.83	1	0.96	-0.24	0.21	-0.04	-0.05	-0.01	0.04	-0.1	-0.06	-0.01	0.19	-0.18	-0.0
T7	-0.06	-0.0	0.70	0.72	0.83	0.83	0.96	1	-0.05	0.07	0.16	0.16	0.19	-0.13	-0.2	0.13	-0.14	0.4	-0.34	0.19
A1	0.00	-0.0	-0.20	-0.01	-0.11	-0.14	-0.24	-0.0	1	-0.48	0.49	0.49	0.39	-0.36	-0.2	0.59	-0.17	0.35	-0.35	0.39
A2	0.34	0.32	0.16	0.04	0.19	0.25	0.21	0.07	-0.48	1	-0.71	-0.71	-0.55	0.67	0.45	-0.71	0.72	-0.62	0.39	-0.5
A3	-0.36	-0.3	0.10	0.48	0.18	0.04	-0.04	0.16	0.49	-0.71	1	0.99	0.89	-0.66	-0.3	0.76	-0.76	0.82	-0.56	0.89
A4	-0.37	-0.3	0.10	0.48	0.18	0.05	-0.05	0.16	0.49	-0.71	0.99	1	0.89	-0.66	-0.3	0.76	-0.75	0.82	-0.56	0.89
A5	-0.17	-0.2	0.25	0.55	0.26	0.13	-0.01	0.19	0.39	-0.55	0.89	0.89	1	-0.66	-0.2	0.74	-0.74	0.83	-0.55	1
A6	0.27	-0.0	0.02	-0.23	-0.08	0.00	0.04	-0.1	-0.36	0.67	-0.66	-0.66	-0.66	1	0.41	-0.76	0.58	-0.67	0.26	-0.6
A7	0.28	0.16	-0.14	-0.10	-0.02	-0.01	-0.16	-0.2	-0.22	0.45	-0.33	-0.32	-0.27	0.41	1	-0.23	0.31	-0.56	0.39	-0.2
FX	-0.08	-0.1	-0.05	0.31	0.09	-0.00	-0.06	0.13	0.59	-0.71	0.76	0.76	0.74	-0.76	-0.2	1	-0.65	0.63	-0.40	0.74
FX	0.26	0.26	-0.19	-0.41	-0.21	-0.10	-0.01	-0.1	-0.17	0.72	-0.76	-0.75	-0.74	0.58	0.31	-0.65	1	-0.72	0.40	-0.7
FY	-0.33	-0.2	0.26	0.54	0.29	0.18	0.19	0.4	0.35	-0.62	0.82	0.82	0.83	-0.67	-0.5	0.63	-0.72	1	-0.56	0.83
FY	0.40	0.24	-0.24	-0.32	-0.16	-0.1	-0.18	-0.3	-0.35	0.39	-0.56	-0.56	-0.55	0.26	0.39	-0.40	0.40	-0.56	1	-0.5
FZ	-0.17	-0.2	0.25	0.55	0.26	0.13	-0.01	0.19	0.39	-0.55	0.89	0.89	1	-0.66	-0.2	0.74	-0.74	0.83	-0.55	1

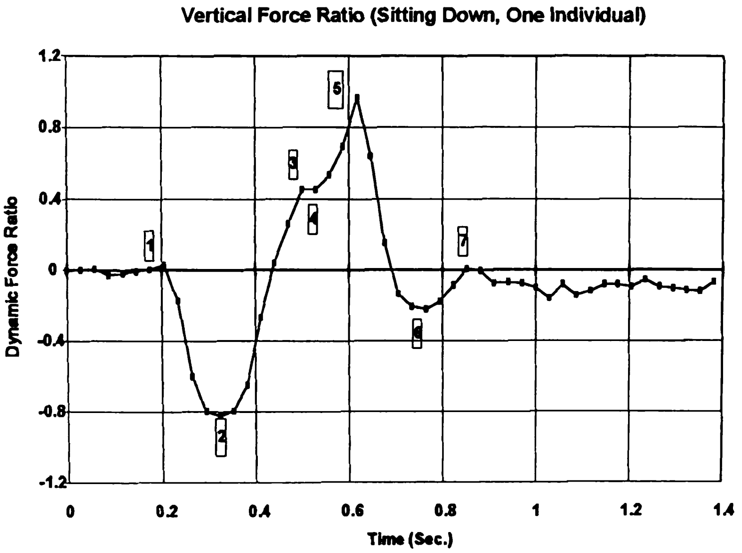

Figure 4.6 A Typical Sitting Down Motion

Table 4.8 Best-Fit Distribution (Sitting Down)

VARIABLE	MEAN	STD	K-SNOR	K-SLOG	PDF
WT	158.5714	34.7261	0.0922	0.0895	LOG-NOR
PHI	0.2521	0.1088	0.1672	0.3588	NORMAL
T2	0.2445	0.1035	0.1448	0.087	LOG-NOR
T3	0.4631	0.1372	0.1354	0.0895	LOG-NOR
T4	0.5328	0.1523	0.1737	0.128	LOG-NOR
T5	0.6025	0.1568	0.217	0.1614	LOG-NOR
T6	0.7479	0.1713	0.146	0.1001	LOG-NOR
T7	0.8622	0.1746	0.158	0.1272	LOG-NOR
A1	0.0417	0.0544	0.2608	0.2315	LOG-NOR
A2	-0.7595	0.6027	0.2467	0.1394	NORMAL
A3	0.8298	1.3254	0.3551	0.0847	LOG-NOR
A4	0.8609	1.3101	0.342	0.1299	LOG-NOR
A5	2.1178	1.7622	0.224	0.0905	LOG-NOR
A6	-0.3263	0.2298	0.2434	0.1171	NORMAL
A7	0.1072	0.0816	0.1857	0.0817	LOG-NOR
FXMIN	0.2497	0.1541	0.1555	0.1033	LOG-NOR
FXMAX	0.2608	0.184	0.2095	0.0899	LOG-NOR
FYMIN	0.0766	0.0738	0.3124	0.1689	LOG-NOR
FYMAX	0.0808	0.0672	0.2444	0.164	LOG-NOR
FZMAX	2.1178	1.7622	0.224	0.0905	LOG-NOR

4.3.3 Standing up

The vertical component of the forcing function for the standing up motion was found to be sufficiently approximated by six control points, Figure 4.7. The program CONTROL.FOR was modified to analyze the motion. Statistics for control points and peak values are shown in Table 4.9. The table shows that the average peak force in the

Table 4.9 Standing Up Control Points Statistics

	WT	PHI	T2	T3	T4	T5	T6	A1	A2	A3	A4	A5	A6	FXM	FXM	FYM	FYM	FZ
AV	159	0.291	0.229	0.528	0.608	0.857	1.003	0.005	0.981	-0.96	-0.93	1.044	0.029	0.371	-0.50	0.107	-0.14	1.47
MI	103	0	0	0	0.059	0.118	0.176	-0.02	-0.57	-4.63	-4.60	0.021	-0.10	0.104	-2.11	0.021	-0.48	0.30
MA	242	0.912	0.5	0.912	0.941	2.235	2.294	0.03	4.131	-0.17	-0.14	4.598	0.317	1.569	-0.11	0.855	-0.01	4.59
RN	139	0.912	0.5	0.912	0.882	2.118	2.118	0.059	4.702	4.466	4.461	4.577	0.42	1.465	2	0.834	0.471	4.29
VA	1171	0.029	0.014	0.033	0.038	0.137	0.144	0	0.865	0.698	0.704	1.609	0.008	0.103	0.162	0.023	0.017	1.52
ST	34	0.17	0.12	0.181	0.195	0.37	0.379	0.016	0.93	0.835	0.839	1.268	0.088	0.321	0.403	0.15	0.13	1.23
CO	22	58.37	52.57	34.30	32.17	43.12	37.81	315.9	94.74	-86.8	-89.7	121.5	305.0	86.53	-79.3	140.1	-89.9	83.7
SE	6	0.029	0.02	0.031	0.033	0.062	0.064	0.003	0.157	0.141	0.142	0.214	0.015	0.054	0.068	0.025	0.022	0.20

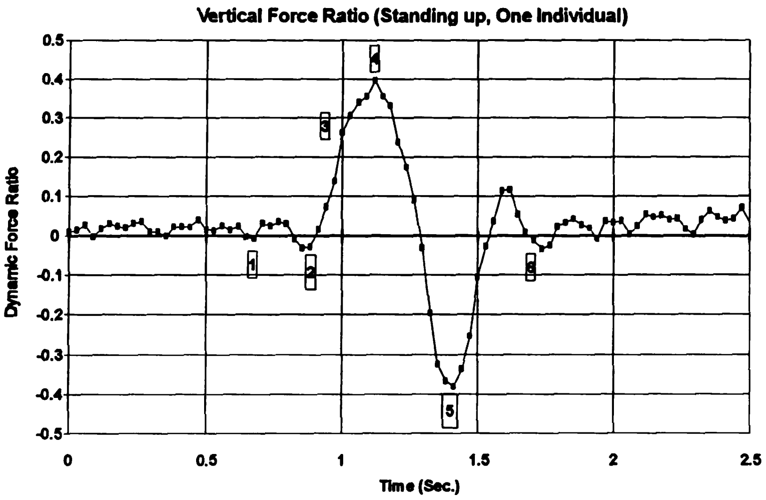

Figure 4.7 A Typical Standing up

Table 4.10 Best-Fit Distribution (Standing Up)

VARIABLE	MEAN	STD	K-SNOR	K-SLOG	PDF
WT	158.5714	34.7261	0.0922	0.0895	LOG-NOR
PHI	0.3135	0.1576	0.1319	0.1196	LOG-NOR
T2	0.2344	0.1113	0.115	0.1298	NORMAL
T3	0.5285	0.1232	0.0985	0.0884	LOG-NOR
T4	0.621	0.1568	0.1199	0.1335	NORMAL
T5	0.7941	0.2276	0.1621	0.1096	LOG-NOR
T6	0.9471	0.2416	0.1205	0.0799	LOG-NOR
A1	0.0159	0.0088	0.1104	0.1623	NORMAL
A2	1.0104	0.8999	0.2032	0.1074	LOG-NOR
A3	0.9604	0.8472	0.308	0.1619	LOG-NOR
A4	0.9419	0.846	0.305	0.1574	LOG-NOR
A5	1.0809	1.2655	0.2624	0.1917	LOG-NOR
A6	0.0805	0.0893	0.2452	0.1862	LOG-NOR
FXMIN	0.3708	0.3255	0.2508	0.136	LOG-NOR
FXMAX	0.5074	0.4085	0.3124	0.1742	LOG-NOR
FYMIN	0.1071	0.1523	0.3053	0.1304	LOG-NOR
FYMAX	0.145	0.1323	0.1946	0.1015	LOG-NOR
FZMAX	1.4718	1.2511	0.229	0.1477	LOG-NOR

vertical direction for a person standing up from a sitting position was about ***1.47*** times his or her weight. Force ratio components in the orthogonal horizontal directions were ***0.5*** for the average force perpendicular to the seating and ***0.14*** for the average parallel force ratio. Comparison with same statistics for the sitting down motion reveals that the later produced larger forces in the vertical direction. This observation is expected since the person standing up is moving against gravity while moving with it in sitting down. The same argument could also be used to explain the finding that the average time needed to perform the motion was ***1.003*** seconds with a delay after the prompt (response lag) of ***0.291*** seconds in comparison with only ***0.86*** seconds to perform the sitting down motion and a response lag of ***0.25*** second.

Linear correlation coefficients were calculated for all control points pairs with the results tabulated in Table 4.10. Finding the best-fit probability distribution to represent each variable in the motion was accomplished by examining the K-S values shown in Table 4.11. The best-fit distribution for the response lag was found to be the Log-Normal distribution. This confirms the finding in Table 4.5 for the case of a single jump. The table reaffirms again that amplitudes can be best-represented by a Log-Normal Distribution.

4.4 Periodic Loading

Periodic Loading includes: periodic jumping, jouncing, and swaying. The dynamic component of the vertical force due to one person performing a periodic motion is approximated by a series of impulses, where the shape of each impulse was defined by

Table 4.11 Correlation Matrix (Standing up)

VA	WT	PHI	T2	T3	T4	T5	T6	A1	A2	A3	A4	A5	A6	FXMIN	FXMAX	FYMIN	FYMAX	FZMAX
WT	1	0.078	-0.17	-0.25	-0.19	-0.23	-0.20	0.339	-0.29	0.253	0.246	-0.00	-0.20	-0.33	0.359	-0.36	0.275	-0.11
PHI	0.078	1	-0.18	-0.15	-0.10	-0.17	-0.25	0.008	-0.38	0.341	0.344	-0.12	0.114	-0.32	0.298	-0.32	0.158	-0.27
T2	-0.17	-0.18	1	0.631	0.704	0.479	0.389	-0.28	0.441	-0.47	-0.48	0.285	-0.06	0.526	-0.46	0.488	-0.37	0.43
T3	-0.25	-0.15	0.631	1	0.736	0.629	0.566	-0.18	0.551	-0.52	-0.52	0.225	-0.10	0.388	-0.54	0.482	-0.27	0.40
T4	-0.19	-0.10	0.704	0.736	1	0.618	0.66	-0.27	0.439	-0.42	-0.43	0.545	0.067	0.481	-0.42	0.348	-0.60	0.59
T5	-0.23	-0.17	0.479	0.629	0.618	1	0.874	-0.05	0.291	-0.19	-0.19	0.142	-0.05	0.243	-0.22	0.198	-0.31	0.25
T6	-0.20	-0.25	0.389	0.566	0.66	0.874	1	-0.01	0.456	-0.34	-0.35	0.501	-0.08	0.409	-0.37	0.248	-0.55	0.56
A1	0.339	0.008	-0.28	-0.18	-0.27	-0.05	-0.01	1	-0.29	0.231	0.209	0.058	-0.10	-0.26	0.235	-0.32	0.319	-0.09
A2	-0.29	-0.38	0.441	0.551	0.439	0.291	0.456	-0.29	1	-0.83	-0.83	0.441	-0.01	0.804	-0.79	0.617	-0.36	0.70
A3	0.253	0.341	-0.47	-0.52	-0.42	-0.19	-0.34	0.231	-0.83	1	0.998	-0.44	0.113	-0.89	0.947	-0.77	0.429	-0.64
A4	0.246	0.344	-0.48	-0.52	-0.43	-0.19	-0.35	0.209	-0.83	0.998	1	-0.46	0.12	-0.89	0.949	-0.77	0.445	-0.66
A5	-0.00	-0.12	0.285	0.225	0.545	0.142	0.501	0.058	0.441	-0.44	-0.46	1	0.12	0.5	-0.39	0.193	-0.62	0.92
A6	-0.20	0.114	-0.06	-0.10	0.067	-0.05	-0.08	-0.10	-0.01	0.113	0.12	0.12	1	0.026	0.148	-0.07	0.064	0.06
FXMIN	-0.33	-0.32	0.526	0.388	0.481	0.243	0.409	-0.26	0.804	-0.89	-0.89	0.5	0.026	1	-0.87	0.701	-0.50	0.67
FXMA	0.359	0.298	-0.46	-0.54	-0.42	-0.22	-0.37	0.235	-0.79	0.947	0.949	-0.39	0.148	-0.87	1	-0.74	0.537	-0.59
FYMIN	-0.36	-0.32	0.488	0.482	0.348	0.198	0.248	-0.32	0.617	-0.77	-0.77	0.193	-0.07	0.701	-0.74	1	-0.40	0.45
FYMA	0.275	0.158	-0.37	-0.27	-0.60	-0.31	-0.55	0.319	-0.36	0.429	0.445	-0.62	0.064	-0.50	0.537	-0.40	1	-0.63
FZMA	-0.11	-0.27	0.431	0.406	0.592	0.254	0.566	-0.09	0.707	-0.64	-0.66	0.92	0.069	0.678	-0.59	0.457	-0.63	1

passing line segments through predetermined control points. Impulses were saved as series of transient actions with the following assumptions:

- *If the prompt frequency is within the physical limits of an individual to perform the prescribed motion, the individual in general attempts to synchronize the motion to follow it. It has been observed that when an individual is out of phase with the prompt in one cycle, an effort is made in the next cycle to adjust to it. Findings from the measured data asserted this observation.*

- *When a person, due to physical limitations, is unable to maintain coherency with a high prompt rate (frequency higher than the range of 2-3 Hz.), Reiner (1987), The person would perform the action at the highest frequency rate possible with no attempt to adjust to the prompt. Although insufficient data were available to verify this assumption, visual observation of tests conducted at frequencies of 4*

and 5 Hz. demonstrated the leaning of individuals to perform actions at their highest possible rate.

- The dominant forces on the platform due to the side swaying motion is the horizontal longitudinal component (parallel to the seating).
- It is assumed that individuals maintain the same level of activity throughout the length of the studied motion (approximately 10 seconds).
- The studied motions are treated as weakly stationary processes. The validity of this assumption is discussed later in the chapter.
- The analyzed motion is confined to the periodical part of the motion. Initial stages of the motion would be simulated separately as transient actions. This is necessary to preserve the assumption made earlier about the stationarity of the studied process.

4.4.1 Periodic Jumping

Force-time histories were divided into cycles, where each cycle is a full period of the action considered. Three control points were sufficient to describe each cycle. One is the peak point in the cycle and the other two define the airborne time. Figure 4.8 presents a typical vertical force ratio of one individual jumping at 2 Hz.

For each measured force-time history, the analysis was conducted using the program CONTROL.FOR which was described in Section 4.3.1 for the case of single jump motion. The program was modified, however, to accomplish the following additional steps for analyzing periodic motions in this study:

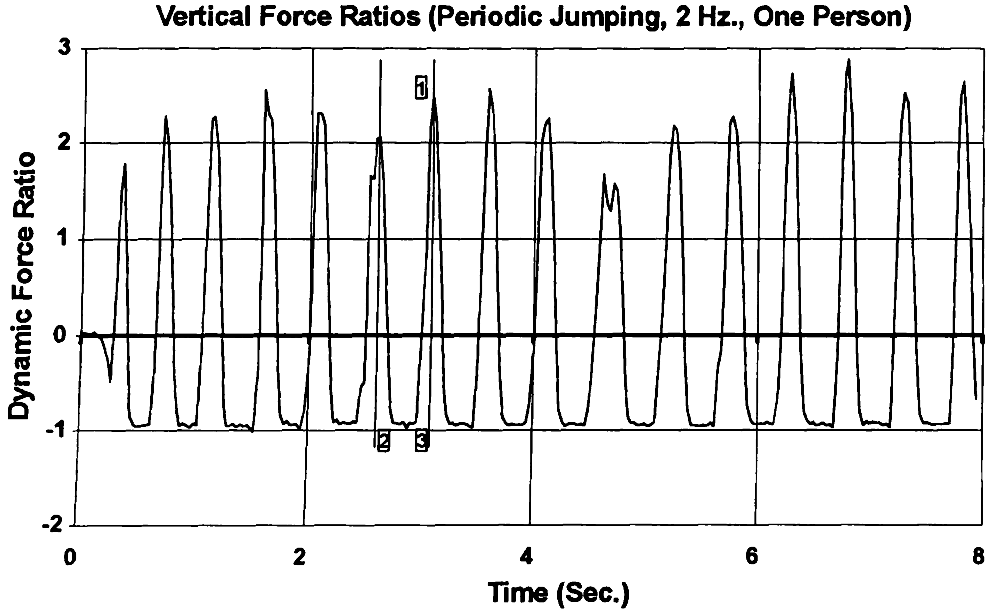

Figure 4.8 A Typical Periodic Jumping

- The number of cycles was computed for each measured load-time history. The starting time of each cycle was recorded along with the peak force in the cycle.
- For each cycle, arrival time and amplitudes of control points were defined. Figure 4.9 shows the control points for the periodic motion in Figure 4.8.
- An output file is created in which the periods of each cycle and control points data are saved. Table 4.12 is a sample of information reported in the output files.
- *Subroutine SPECTRA.FOR is called to compute the autocorrelation, autospectrum function and the root mean square of the process considered. Typical screen output is shown in Figure 4.10 and 4.11 for the forces in vertical and parallel axes of motion.*

Output from CONTROL.FOR shows that the average peak force ratio in the vertical direction for a person jumping periodically in place at a 2 Hz. prompted frequency was about 2.273 times the person's weight, Table 4.13. This ratio decreased to 2.154 at 3 Hz.

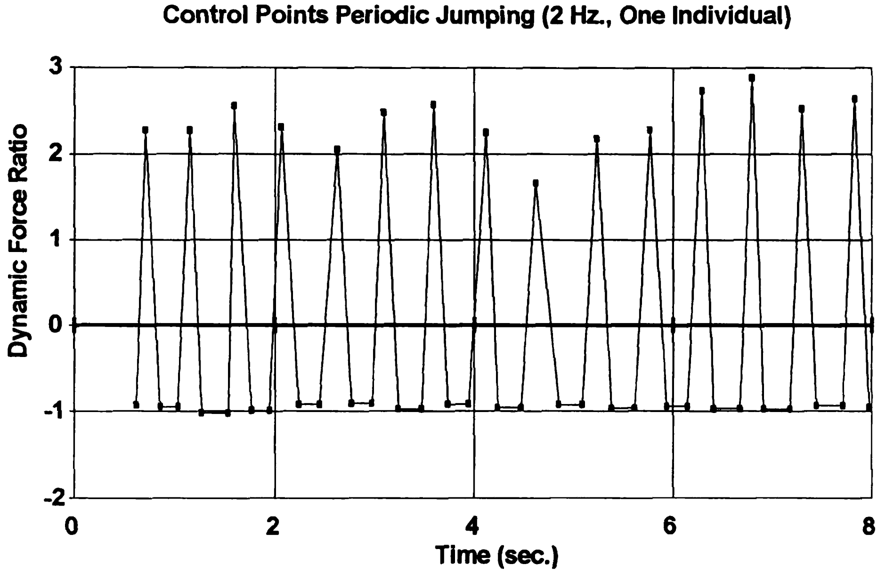

Figure 4.9 Control Points for a Typical Periodic Jumping

and 2.002 at 4 Hz. This finding shows that the higher the jumping frequency the lower the average peak force ratio a person generates. Maximum peaks also followed the same pattern and dropped from an average of 2.852 for 2 Hz. jumping to 2.750 and 2.448 for 3 and 4 Hz. respectively.

At a prompt of 2 Hz. (0.5 sec. period), the period of the jumping ranged from 0.472 to 0.59 seconds with an overall average of 0.509 seconds. This means that on the average, individuals were jumping at a frequency of 1.95 Hz. while the prompt frequency was 2 Hz. This difference between prompt and response increased at 3 and 4 Hz. as shown in Table 4.14. The average response period for persons jumping at a prompt of 0.333 seconds (3 Hz.) was 0.36 seconds (2.77 Hz.) and for the case of a prompt period of 0.25 seconds (4 Hz.), it was 0.285 seconds (3.51 Hz.). This observation verifies the assumption made earlier about the inability of a person to maintain coherency with the

Table 4.12 2-Hz. Periodic Jumping

No	WT	AVG T	STD T	AVG P	STD P	MAX P
1	165	0.476	0.054	2.863	0.582	4.28
2	173	0.478	0.045	2.22	0.416	3.331
3	126	0.477	0.033	3.148	0.39	3.94
4	160	0.497	0.021	3.627	0.468	4.219
5	205	0.476	0.032	2.539	0.404	3.175
6	196	0.509	0.042	2.397	0.252	2.883
7	140	0.501	0.039	2.685	0.246	3.186
8	186	0.492	0.104	3.222	1.395	4.536
9	150	0.507	0.023	2.513	0.136	2.824
10	189	0.524	0.035	3.126	0.485	3.736
11	163	0.521	0.013	2.746	0.132	3.006
12	125	0.55	0.173	2.669	0.352	3.517
13	234	0.488	0.027	1.524	0.166	2.002
14	132	0.587	0.025	3.825	0.326	4.227
15	130	0.498	0.037	1.784	0.218	2.462
16	146	0.526	0.023	1.93	0.197	2.372
17	242	0.505	0.026	1.542	0.136	1.774
18	184	0.472	0.028	2.451	0.469	3.107
19	155	0.547	0.076	1.094	0.24	1.579
20	200	0.504	0.034	2.537	0.32	3.069
21	191	0.505	0.05	1.066	0.252	1.861
22	166	0.493	0.035	1.894	0.261	2.522
23	215	0.517	0.024	3.459	0.435	3.935
24	154	0.472	0.03	2.608	0.193	3.084
25	153	0.533	0.03	2.454	0.47	3.198
26	164	0.504	0.027	3.213	0.171	3.473
27	120	0.51	0.1	2.188	0.176	2.487
28	103	0.51	0.039	1.399	0.323	2.243
29	126	0.52	0.024	1.337	0.219	1.732
30	116	0.522	0.097	1.129	0.224	1.512
31	185	0.489	0.053	1.684	0.253	2.189
32	145	0.504	0.042	2.444	0.42	3.029
33	140	0.496	0.044	2.013	0.148	2.3
34	170	0.513	0.025	1.758	0.16	2.008
35	120	0.511	0.028	2.044	0.287	2.542
36	177	0.59	0.267	2.142	0.167	2.384
37	124	0.518	0.017	1.924	0.235	2.282
38	128	0.513	0.029	1.334	0.334	1.829
39	110	0.498	0.038	2.107	0.369	3.413
AVG	159.179	0.509	0.048	2.273	0.319	2.852
MIN	103	0.472	0.013	1.066	0.132	1.512
MAX	242	0.59	0.267	3.825	1.395	4.536
RNG	139	0.118	0.254	2.759	1.263	3.024
VAR	1135.89	0.001	0.002	0.498	0.044	0.651
STD	33.703	0.026	0.046	0.706	0.21	0.807
COV	21.173	5.113	95.406	31.053	65.763	28.278
SER	5.397	0.004	0.007	0.113	0.034	0.129

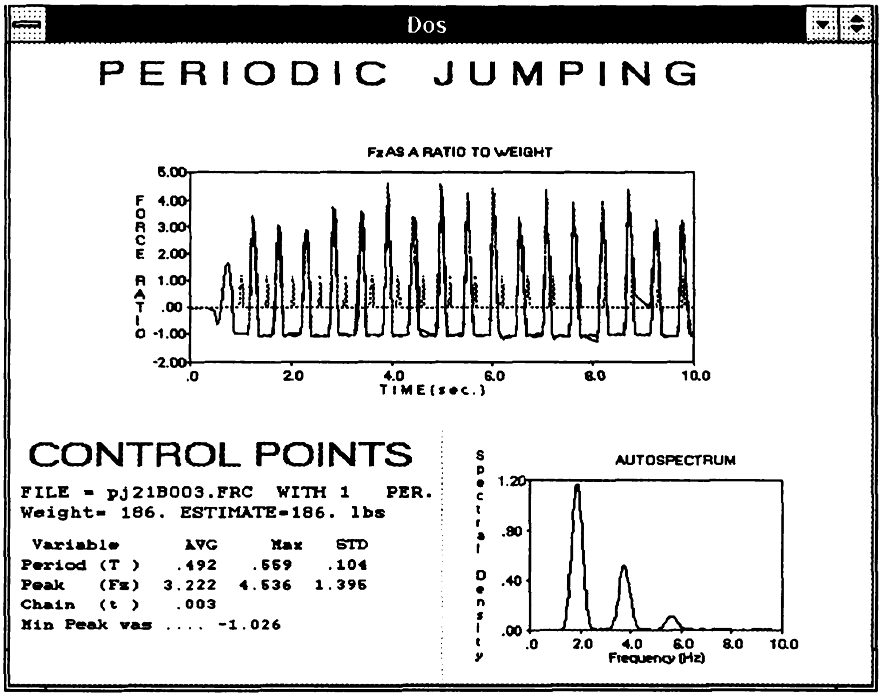

Figure 4.11 CONTROL.FOR Output (Periodic Jumping, 2 Hz., Vertical Ratio)

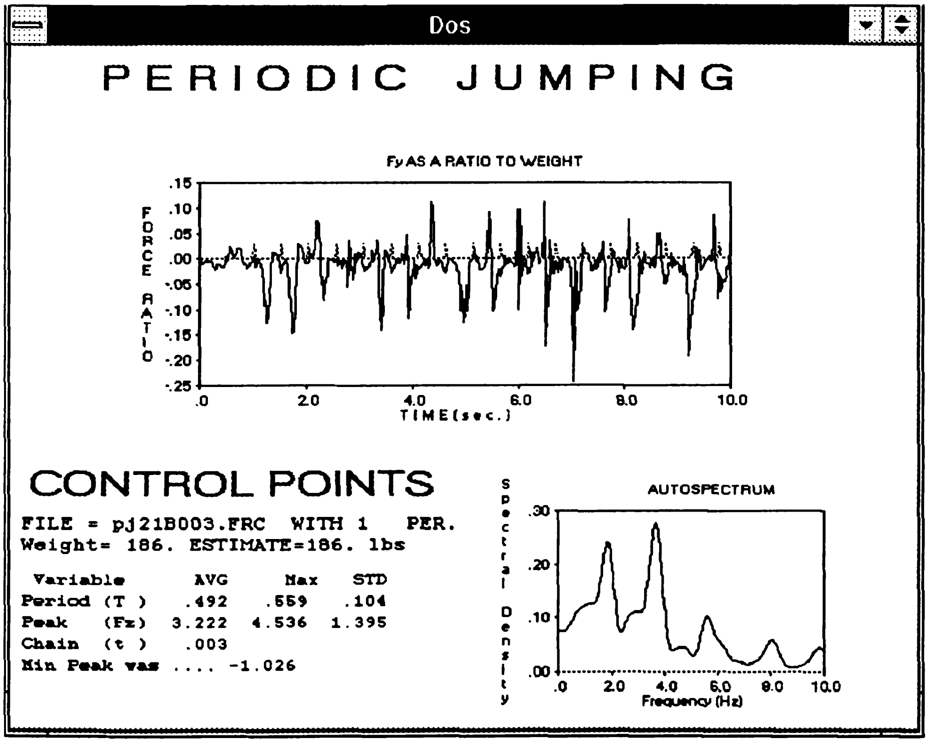

Figure 4.10 CONTROL.FOR Output (Periodic Jumping, 2 Hz., Parallel Ratio)

Table 4.13 Peak Analysis (Periodic Jouncing)

Frequency		AVG	MIN	MAX	RNG	Var	STD
2 Hz.	AVG P	0.839	0.334	1.657	1.323	0.078	0.279
	MAX P	1.061	0.483	2.098	1.615	0.111	0.334
3 Hz.	AVG P	1.08	0.466	1.971	1.504	0.125	0.354
	MAX P	1.369	0.602	2.419	1.817	0.227	0.477
4 Hz.	AVG P	1.004	0.491	2.119	1.628	0.106	0.326
	MAX P	1.327	0.652	2.562	1.91	0.172	0.414

Table 4.14 Control Points Statistics (Periodic Jumping)

Frequency	No	AVG	MIN	MAX	RNG	VAR	STD
2hz	AVG T	0.509	0.472	0.59	0.118	0.001	0.026
	AVG P	2.273	1.066	3.825	2.759	0.498	0.706
	MAX P	2.852	1.512	4.536	3.024	0.651	0.807
3hz	AVG T	0.36	0.33	0.538	0.209	0.001	0.039
	AVG P	2.154	1.618	4.393	2.775	0.223	0.472
	MAX P	2.75	1.867	4.935	3.069	0.284	0.533
4hz.	AVG T	0.285	0.23	0.436	0.206	0.002	0.04
	AVG P	2.002	1.481	3.556	2.075	0.162	0.403
	MAX P	2.448	1.705	4.187	2.482	0.254	0.504

high prompt frequency.

To further study this pattern, the period of each cycle of a typical periodic jumping at 2, 3, and 4 Hz. were plotted against the cycle number as shown in figures 4.12. The figure shows that when the person performs the motion at a period less than the prompt frequency, an attempt is made in the next cycle to increase the period to adjust to the prompt frequency. This assumption proved to be valid for most of the recorded cases for jumping at 2 Hz. The figure also shows that this particular person started at frequency different from the prompt. However, as the test proceeded, the person adjusted to the

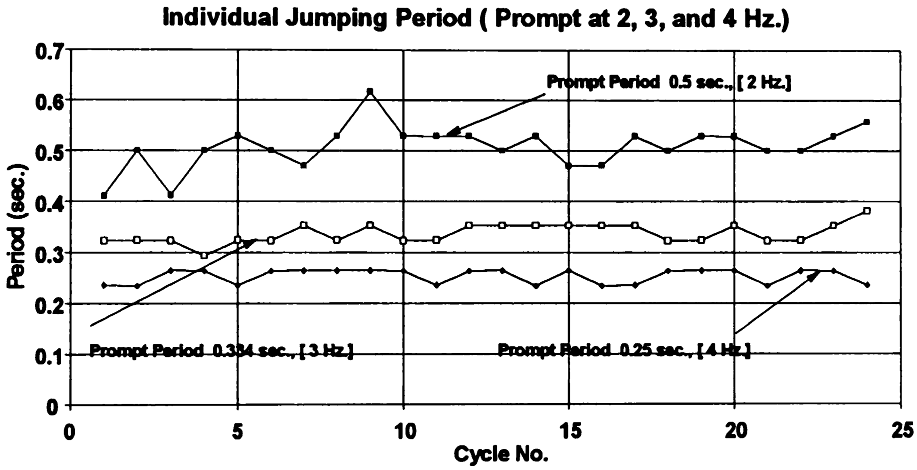

Figure 4.12 A Typical Periodic Jumping Response Period

prompt and the variation around the prompt period decreased. On the contrary, when the prompt period decreased to 0.33 (3 Hz.) and 0.25 (4 Hz.), the two individuals were jumping consistently at lower frequencies than the prompt frequency with very few attempt to adjust to it. This observation will be used in the modeling process as explained in the next chapter.

4.4.2 Periodic Jouncing

Five control points were sufficient to describe vertical force ratios for each cycle in the periodic jouncing motion One is the peak point in the cycle and the other four define the shape of the negative part. Figure 4.13 presents a typical vertical force ratio of one individual performing periodic jouncing at 2 Hz.

Average peak force ratio in the vertical direction for a person jouncing in place at a 2 Hz. prompted frequency was about 0.84 times the person's weight, Table 4.15. This ratio increased to approximately 1.0 at 3 Hz. and 4 Hz. Maximum peaks also followed

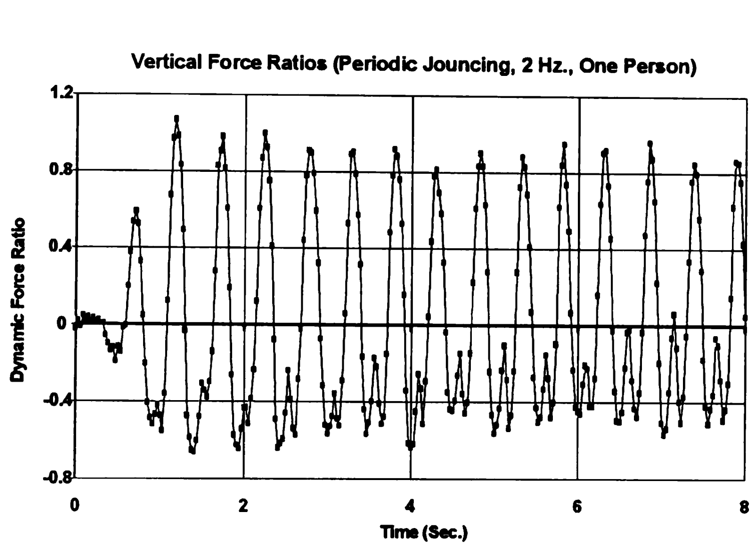

Figure 4.13 A Typical Periodic Jouncing

the same pattern and increased form an average of 1.06 for 2 Hz. jouncing to 1.37 and 1.33 for 3 and 4 Hz. respectively. The average period of individuals jouncing periodically at 2 Hz. (period of 0.5 seconds) was found to be 0.501 seconds. This means that on the average, individuals were jouncing at the same prompted frequency of 2.0 Hz. As in the case of periodic jumping, individuals were unable to maintain coherency with higher frequency (3 and 4 Hz.) and their average jumping period were 0.368 (2.71 Hz.) and 0.28 (3.57 Hz.) respectively.

4.4.3 Side Swaying

Side swaying forces were dominant in the horizontal direction and parallel to the individuals' seating. The recorded motion is shown in figure 4.14 where the presented force ratio is in the horizontal plane. Only peak forces were collected for the analysis. Typical force ratios in the horizontal plane are presented for the case of periodic swaying

Table 4.15 Periodic Jouncing Statistics (One Individual, 2 Hz.)

Test	Wt	Avg Peak	Max Peak
1	196	0.096	0.199
2	140	0.084	0.175
3	186	0.03	0.147
4	150	0.053	0.149
5	189	0.274	0.337
6	125	0.197	0.289
7	234	0.11	0.211
8	132	0.112	0.182
9	146	0.103	0.204
10	191	0.147	0.222
11	166	0.122	0.179
12	215	0.209	0.259
13	331	0.177	0.227
14	153	0.184	0.354
15	307	0.1	0.112
16	103	0.089	0.17
17	126	0.129	0.283
18	282	0.086	0.156
19	145	0.122	0.169
20	340	0.081	0.144

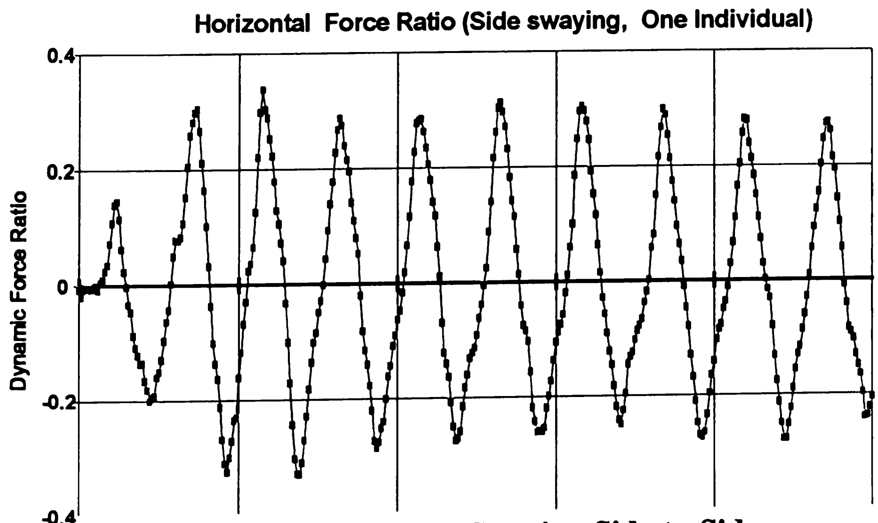

Figure 4.14 A Typical Swaying Side to Side

at 2 Hz, Table 4.16. Maximum force ratios in the horizontal direction were found not to correlate with the individuals weights. The same conclusions that were obtained from the analysis of periodic jumping and periodic jouncing regarding the inability of individuals to maintain coherency at higher frequencies were observed in this motion also. Table 4.17 shows typical statistics for reported average and maximum peaks in the horizontal direction. It shows the average force was about 0.125 times the person's weight with a range from 0.03 to 0.274. The Maximum peak ratios were in the range of 0.112 to 0.354 times the persons' weight with an average of 0.208.

Table 4.16 Max. Peaks (Side Swaying)

Test	WT	Avg Peak	Max
AVG	192.85	0.125	0.208
MIN	103	0.03	0.112
MAX	340	0.274	0.354
RNG	237	0.244	0.242
VAR	4812.328	0.003	0.004
STD	69.371	0.057	0.064

Table 4.17 Typical Side Swaying Data

Test	Wt	Avg Peak	Max
1	196	0.096	0.199
2	140	0.084	0.175
3	186	0.03	0.147
4	150	0.053	0.149
5	189	0.274	0.337
6	125	0.197	0.289
7	234	0.11	0.211
8	132	0.112	0.182
9	146	0.103	0.204
10	191	0.147	0.222
11	166	0.122	0.179
12	215	0.209	0.259
13	331	0.177	0.227
14	153	0.184	0.354
15	307	0.1	0.112
16	103	0.089	0.17
17	126	0.129	0.283
18	282	0.086	0.156
19	145	0.122	0.169
20	340	0.081	0.144

5.0 HUMAN LOAD MODELING

5.1 Introduction

A computer program; *H*uman *L*oad *S*imulation, ***HLS***; utilizing the understanding of the nature of each motion and factors that affect its force history was developed giving researchers an advanced tool to use in design and analysis. The forcing functions produced by the program simulate several types of human movements. The probable levels of instantaneous or peak values of human loads are represented by probability density functions. The program may be used to generate load distributions for any combination of human loading conditions, composed of those presented herein. Input to the program includes information about the type of action to be simulated, group size, and parameters necessary to define the individual loads. The program calls an array of subroutines to calculate forcing functions for different human actions and these forcing functions are combined in any form requested by the analyst. Output of the program is based on the type of action and can be summarized as follows:

Transient loading (pulse loading): Statistical parameters of the control points associated with the time histories of the transient action are reported. The amplitudes of peaks and their arrival time are calculated. The program would also provide the probability density function of the peaks or the impact factors for the input action.

Steady state loads (periodic loading): The program produces force-time histories, estimates the probability densities of peaks, and computes the spectral energy distributions in the frequency domain in the form of power spectral density functions (PSD). Power spectra are determined by taking the Fourier transform of the autocorrelation functions associated with the simulated time histories, Clough and Penzien (1975). Fast Fourier Transform (FFT) method outlined by Newland (1975), and implemented by Harichandran (1984) is adopted in the program.

In order to develop the above mentioned Human Load Simulation program, ***HLS***, several steps were required. First, model hypotheses were examined and summarized in the previous chapter. In this chapter, an attempt to describe an efficient methodology to simulate human actions is presented. The next chapter illustrates several applications to explain different ways of using the developed software.

This chapter explains the random processes application to the development of the program, the generation of correlated random variables to use in estimating control points for each load type considered, and development of the Monte Carlo simulation process. The intention of the next section is to briefly explain the mathematics involved in these steps for the use in modeling human loads.

5.2 Basic Concepts

5.2.1 Random Processes

Since loads due to human movements vary randomly, current methods of handling them are fundamentally deficient because they do not provide quantitative information

regarding load variabilities in a rational way. Random process theory deals directly with uncertainty by characterizing random excitations, human loads in this case, with a statistical approach. In this way, a large portion of the uncertainty in defining the forces due to human actions can be investigated and expressed in a way that can be better understood and used. The application of the concept of random processes to the modeling of the loads due to human movements seems to be very appropriate in this sense.

The function describing the force due to a human action is a "*random process*" because it varies with time. A random process cannot be defined by a single measure or function. The set of all possible "*sample functions*" constitutes the random process, and is called the "*ensemble*". Figure 5.1 displays three typical samples in an ensemble of random process ***F(t)***. The figure shows that, a different sample function is obtained every time the load is recorded. The random process can then be described through its probability properties, which allow estimates or forecasts within some degree of confidence, Augusta (1984). Some of the important definitions in using random processes are described as follows:

Ensemble Average: Is the average of the random process ***F(t)***, measured at time t:

$$E\,[\,F\,(\,t_1\,)\,] \;=\; Lim_{n\to\infty} \sum_{i=1}^{N} \frac{F_i\,(\,t_1\,)}{n} \qquad \text{[5.1]}$$

where N: is the number of samples in the ensemble.

Stationary Random Process: If all joint probability density functions obtained for the ensemble do not depend on time, the random process is said to be stationary. If all the average values (moments) remain constant over a change in time, the process is called

a "strong stationary process". If only the first and the second moments (mean and covariance) of the process remain constant with respect to time, the process is a "weak stationary process", where the first and second moments expressions are

$$\begin{aligned} E[F(t_1)] &= E[F(t_2)] \\ E[F(t_1)F(t_2)] &= E[F(t_1+\tau)F(t_2+\tau)] \end{aligned} \qquad \ldots\ldots [5.2]$$

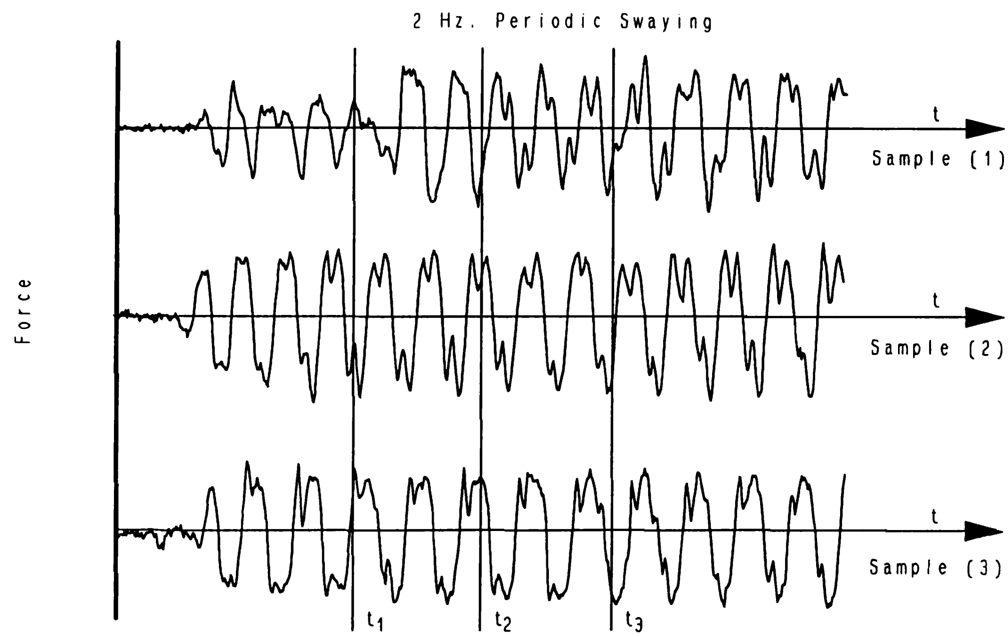

Figure 5.1 Typical Samples in the Random Process F(t)

Ergodic Process: The process is called "ergodic", if in addition to being stationary, the average taken at any sample is the same as the ensemble averages. If the process is ergodic, it is stationary.

$$< F (t_i) > \; = \; Lim_{t \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} F (t_i) \, dt \; = \; E (F(t)) \ldots . \quad [5.3]$$

Autocorrelation: Autocorrelation is the average value of the product of the forcing function at time t and at time $(t+\tau)$ for a random process $F(t)$, sampled at time t and then again at time $t+\tau$. If the process is weakly stationary, the autocorrelation depends only on τ and is equal to the mean square of the process $F(t)$:

$$R_f (\tau) \; = \; E [F (t_1) . F (t_1 + \tau)] \ldots \ldots \ldots . \quad [5.4]$$

$$\textit{for ergodic process} \quad R_f (\tau) \; = \; \frac{1}{T} \int_0^T F (t) F (t + \tau) \, dt \, , \quad T \to \infty \quad [5.5]$$

Power Spectral Density: It defines the frequency composition of the forcing function $F(t)$, and is the Fourier Transformation of the autocorrelation function.

$$S_f (\omega) \; = \; \frac{1}{2 \pi} \int_{-\infty}^{\infty} R_f (\tau) \exp (-i \omega \tau) \, d\tau \ldots \ldots . \quad [5.6]$$

where ω = is the circular frequency.

5.2.2 Random Number Generation

Random number generation is an important component of every stochastic simulation model. An essential property of every random number generator is the ability to generate random variables that are uniformly distributed on the interval (0,1) and stochastically independent. John von Neumann (1951) and others suggested using the computer's arithmetic operations to produce sequences of numbers that, while being entirely deterministic, had the appearance of randomness. The numbers resulting are generally

termed pseudorandom numbers. That is, since the numbers are generated algorithmically, they are not actually random. However, for practical purposes, pseudorandom numbers are considered to behave as random numbers, i.e., to be uniformly distributed and mutually independent provided that the random generator has a long- enough cycle length. The cycle length is the number of pseudorandom numbers generated before the same sequence of numbers starts again.

Several methods may be employed to generate uniformly distributed random numbers (Motooka 1954). The first method uses a recurrence relation to generate successive random numbers by employing any algorithm consisting of arithmetic and logical operations. The second method employs a special computer accessory, namely a physical random-number generator, that transforms the results of a physical random process (like electrical noise generators, pulse detectors of ionizing radiation events, or gas discharge tubes) into a sequence of binary digits in the computer (Golenko and Smiryagin 1960). This method increases the speed of computation, since the generated numbers are written in a fixed standard location of the memory. A third, rarely-used, method consists of inserting tables of uniformly distributed random numbers into the working memory of the computer. The fourth method, **which is used in this study**, is to obtain uniform random numbers by the mixing method. This method is based on utilizing specific features of the electronic computer to generate pseudorandom numbers. The method begins by assigning an initial integer number, which is placed in a certain location in computer memory. Then, the random numbers are calculated by shifting the image of the bits of the previously generated number a certain number of places. This technique of shifting

registers in the computer memory is used by the *UNI* uniform random generator subroutine to generate uniform pseudorandom numbers for this study.

The advantages of pseudorandom numbers over purely random numbers is that the computer can generate them. Moreover, an important statistical advantage is the ability of such subroutines to reproduce the same sequence of pseudorandom numbers if it starts with the same initial value in the generating formula.

5.2.3 Generation of Correlated Random Numbers:

From the analysis of different human load motions, it was found that correlation exists among control point arrival times and in some cases among the control point amplitudes. It is therefore necessary to preserve the same correlation structure if these variables are to be estimated as random variables to simulate the motion of an individual. The procedure explained here to generate correlated random variables was programmed by Harichandran, (1992), and is based on a methodology explained by Johnson, (1987) for the generation of multivariate normal distributions.

Assume that it is desired to generate y_i random variables which maintain a correlation structure preserved in the covariance matrix $cov\ (y_n,y_m)$. The samples of y could be generated from a sample of independent standard normal random variables x_i using the following linear transformation :

$$[\,Y\,] = [\,H\,]\,[\,X\,] + [\,\mu\,] \qquad [5.7]$$

where H is a lower triangular transformation matrix that relates to the covariance matrix of the correlated variables y_i, and μ is a vector of the mean of each variable y_i, Then it can be shown that

$$E[\,Y\,Y^T\,] \;=\; E[\,H\,X\,X^T\,H^T\,] \;=\; H\,E[\,X\,X^T\,]\,H^T \cdots \quad [5.8]$$

$$E[\,Y\,Y^T\,] \;=\; H\,H^T \cdots\cdots\cdots\cdots \quad [5.9]$$

since the product of $E[\,X\,X^T\,]$ is an identity matrix because the random variables x_i's are independent and have a unit variance. Then knowing the left hand side of eq. 5.9 which is the definition of the covariance matrix, the entries of the matrix resulting from the product $H\,H^T$ is found and therefore the matrix H could be obtained. Equation 5.9 represents the Cholesky factorization of the symmetric positive definite covariance matrix of Y.

5.2.4 Monte Carlo simulation

Generally, a simulation model seeks to duplicate the behavior of a phenomena knowing the statistical properties of its random variables and the interactions between its components. Simulation results will reach steady state only after the numerical experiment is repeated a sufficiently large number of times. Thus, in order for the output of a model to be representative of what would be expected in the long run, simulation must produce a large enough sample (or be repeated enough times) to ensure representative results.

Monte Carlo methods are those in which properties of the distributions of random variables are analyzed by use of simulated random numbers. The Monte Carlo method can be defined in a broad sense as "any technique for the solution of a model using random number or pseudorandom numbers" (Hammersly and Handscamb 1964). One application of the Monte Carlo method is to find the distribution or some of the parameters of the distribution of a stochastic variable (probabilistic variable). This variable is a known function of one or more other stochastic input variables that have known distributions. The method involves the determination of the distributions of the parameters, selection of a random sample of each parameter, and combination of these samples to obtain a measure of overall performance or reliability. The process of random selection and determination of response effects is repeated a large number of times, each repetition resulting in another independent estimate of the modeled function. As the sample size increases, the distribution of the sample becomes a better representation of the distribution of the population.

In this study, Monte Carlo simulation is used to produce a 1000-point sample of force-time history values for each load type combination provided by the user of ***HLS***.

5.3 Transient Load Modeling

Transient loads are modeled in ***HLS*** in a process consisting of four stages:

- **a)** Input from the analyst provides the program with the necessary information to determine the type of transient action, number of participants, and type of output media (printer, screen, or files).
- **b)** The program reads the parameter files for the selected motion which contains the control points statistics, phase lag parameters, and the model error (explained next). The seed array for the random number generator is read at this stage.

- c) *HLS* then calls the appropriate subroutines for the selected actions to determine the force-time history in the vertical direction, and computes the maximum positive and negative horizontal components. The model error is generated at this stage. The group force-time history is also calculated, and the maximum and minimum values of the generated forces are prepared for the output stage.
- d) Calculated individual force-time histories along with the group force time history are sent to the requested output media at this stage. The program plots force-time histories to the screen with the important statistics and saves all output files in the same directory that the program is running from.

Figures 5.2 and 5.3 shows typical screen output for the case of a single jump and sitting down by one individual.

In the next section, the mathematical approach used in developing load models for transient motions is explained. The single jump motion is used to demonstrate the methodology and the steps for arriving at a reliable estimate of the load produced by an individual, any number of individuals, or a group.

5.3.1 Mathematical Approach

Vertical forces from transient actions are modeled in *HLS* as a series of lines or curves that closely fits data measured earlier and maintains the same control points statistics and correlation between different arrival times and amplitudes. Horizontal components are modeled as two impulses with magnitudes and arrival times that conforms to the observed measurements.

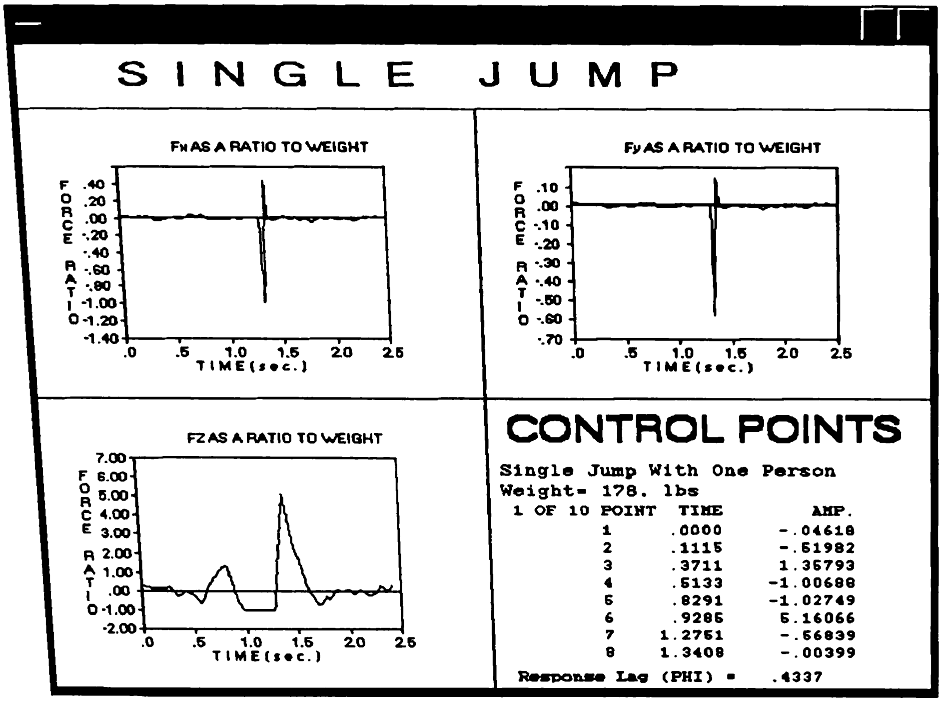

Figure 5.2 *HLS* Screen Output (Single Jump, One Individual)

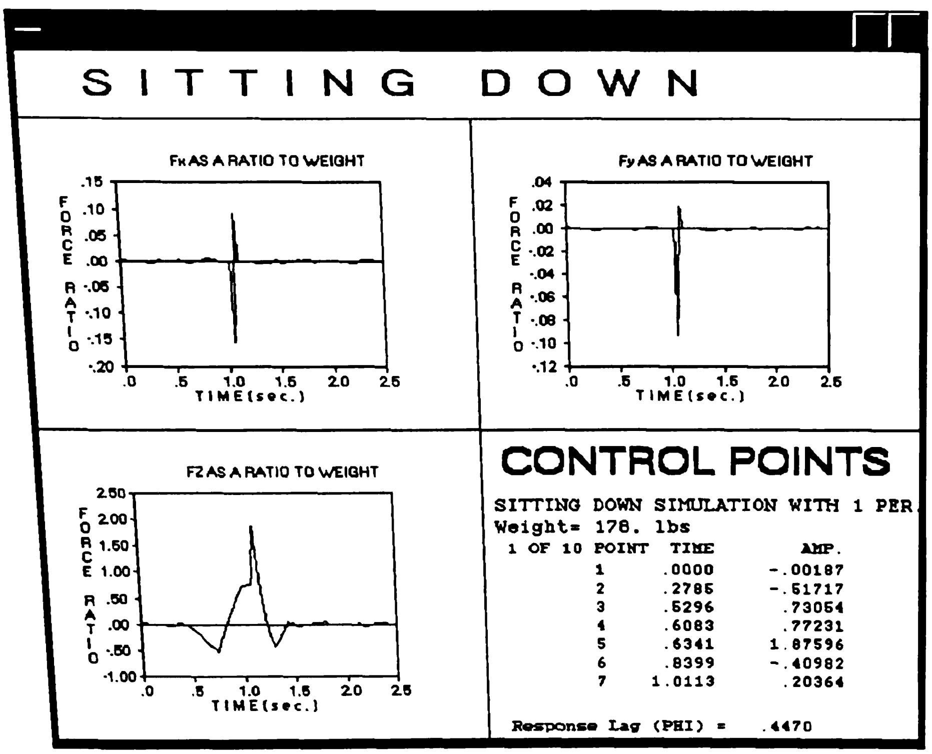

Figure 5.3 *HLS* Screen Output (Sitting Down, One Individual)

Random Variables

The process of transient load modeling starts by determining the weight of the person or persons performing a certain type of transient motion. Two options are available to the user of ***HLS*** ; one is to provide weights, the other to use the random number generator to estimate this variable. Currently, simulated weights are based on a model suggested by Sidney (1974). The model is a log-normal probability density function with a mean of ***163*** lbs, and a standard deviation of ***17*** lbs.

The response lag ϕ, for any sample is predicted as a log-normal random variable with a mean of ***0.277*** seconds and a standard deviation of ***0.087*** seconds. Generating a log-normal random variable is based on the assumption that if a variable ϕ is log-normally distributed with a mean of μ_ϕ and a standard deviation of σ_ϕ, then the natural logarithm of ϕ, denoted as ***Ln (φ)***, is normally distributed.

$$\textit{assuming} \quad y = Ln(\phi) \cdots\cdots\cdots \quad [5.10]$$

coefficient of variation as follows:

$$\textit{with Coefficient of Variation} \quad V_{(\phi)} = \frac{\sigma_\phi}{\mu_\phi} \cdots\cdots \quad [5.11]$$

and mean and standard deviations are calculated as:

$$\mu_y = \mu_{Ln\,(\phi)} = Ln\,(\,\mu_\phi\,) - \frac{\sigma^2_y}{2} \cdots\cdots \quad [5.12]$$

$$\sigma_y = \sigma_{Ln\,(\phi)} = \sqrt{Ln\,(\,1 + V_\phi^2\,)} \cdots\cdots \quad [5.13]$$

Estimating a log-normal random variable is a three-step process. First the transformed variable y is obtained by from Eq. [5-10] and treated as a normal random variable with a mean and standard deviation calculated from Eq. [5.12] and Eq. [5.13]. Then a random variable is generated for the variable y , within the given mean and standard deviation, using random generations techniques discussed in Section 5.2. Finally, the random value ϕ is computed as :

$$\phi = e^{y} \quad \text{[5.14]}$$

It is essential to note that log-normally distributed random variables cannot assume values less than zero. This assumption is suitable for the Response Lag in this case since participants always wait for the prompt to perform the intended motion.

Correlated Random Variables

Arrival times and amplitudes are generated as correlated random variables using the procedure in Section 5.2. Random variables with any probability distribution function other than the normal were transformed to normal equivalence. The covariance matrix of the correlated variables were computed by multiplying the correlation coefficients by the appropriate standard deviations. The subroutine HMATRIX was used to perform linear transformation of standard normal variables to obtain the correlated variables. The mean values of the correlated variables were then added to the generated random variables. Then the new correlated random variables were transformed back to their original assumed distribution. It was found that the best-fit distributions for all random variables in the study were either normal or log-normal. Therefore, the transformation was performed using the process outlined in Eq. [5.10] through Eq. [5.14].

Model Error

The load model is constructed by fitting straight or curved lines to the randomly estimated control points. The determination of using straight lines or curves was based on minimizing the error or the difference between generated and measured load histories. A program MODEL.FOR, was used to fit the control points to the measured data and connect the points first by straight line and then by second degree curves. Estimates of the error associated with each trial were calculated. Figure 5.4 is a typical screen output from the program which shows an average error calculated from 45 samples of a single jump by one individual. The error spectrum was calculated for the average error and it is shown in figure 5.5. It was found that straight lines provided less error in most cases. Figure 5.6 shows a typical single jump where only one curve line was needed to connect control points 5 and 6.

Further analysis was conducted on the error function which was defined as the difference in amplitude between measured and fitted data. Treating the error function as a random process shows that over a large number of samples, the random process approach stationarity and therefore realizations of the error can be predicted from its amplitude spectrum, Newland (1987).

It is essential to note that for this particular case of single jump, a part of the process is known to be deterministic; that is the airborne time where the dynamic force ratio is always *-1*. This deterministic part of the motion disturbs the assumption of stationarity. The problem could be avoided by removing the error terms from the deterministic porion of the motion. Another alternative would be to divide the error process into three

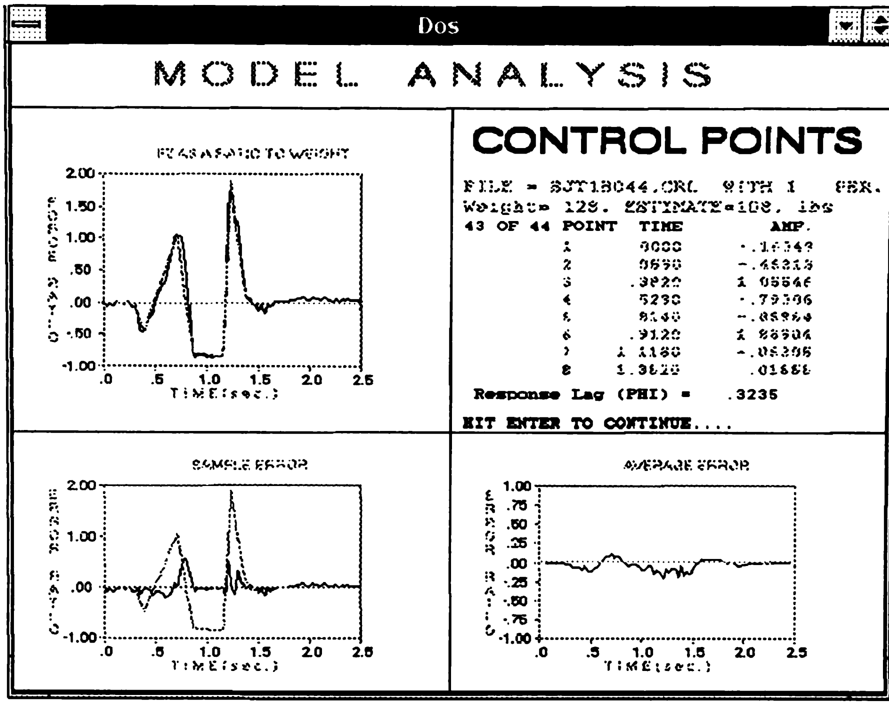

Figure 5.4 Error Function (Single Jump, One Individual)

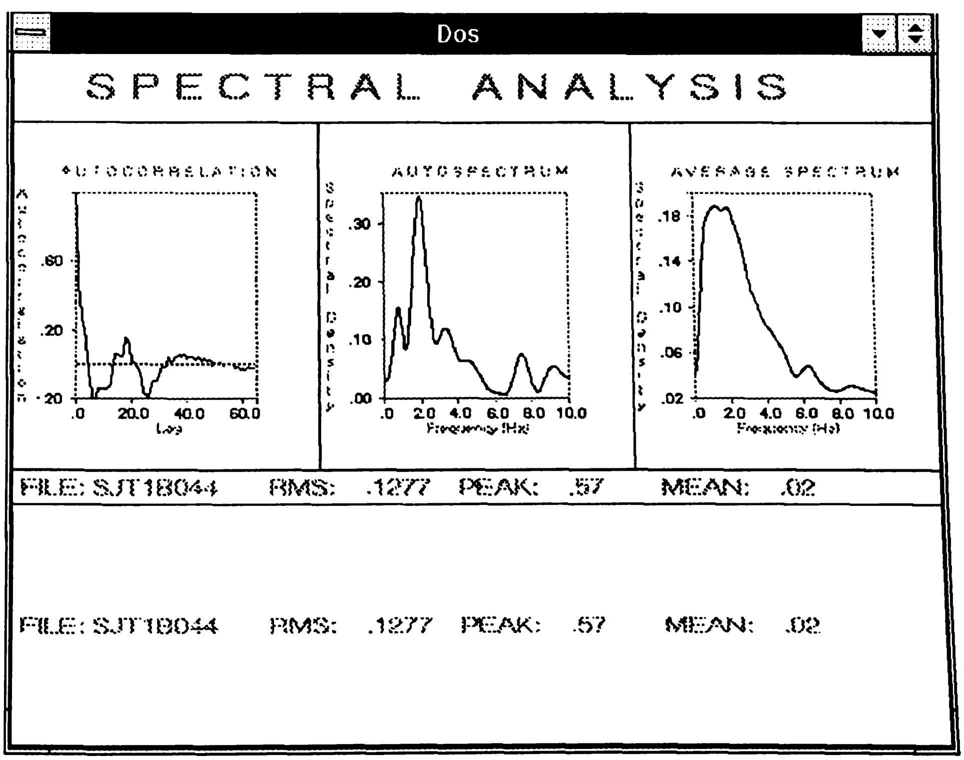

Figure 5.5 Error Spectrum (Sitting Down, One Individual)

segments; one before the airborne time, then the deterministic part followed by the landing part. The first and third segments could then be treated as two separate random processes. Removing the error term from the deterministic part was chosen in this study to simplify the modeling process.

Model Error Simulation

The error functions were simulated from their spectra for each generated sample and

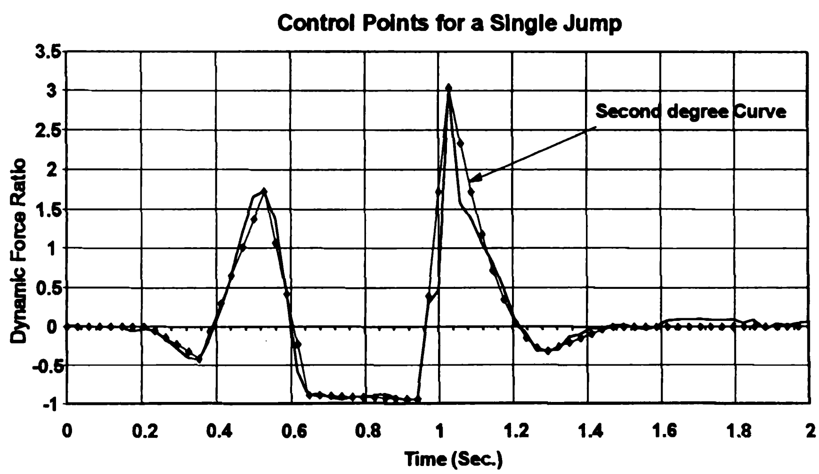

Figure 5.6 Straight and Curved Lines of Single Jump Model

then added to the simulated force ratios in the three principal axes of motion. Taking advantage of the central limit theorem and assuming that the error variable is the result of summing many statistically independent variables, the error function is assumed to approximate a Gaussian random process, Newland (1979). The mean and variance of the error stationary Gaussian process is computed for each motion considered in the study using the program **MODEL.FOR**.

To ensure consistent estimates of the error function realizations using the computed power spectrum, several smoothing band widths were considered. Figures 5.7 and 5.8 are two of the error spectra where the smoothing bandwidth in the first figure was about 0.73 Hz. while it was 0.36 Hz. for the second. It can be concluded from visual inspection of the two figures that the bias introduced into the spectrum when using the higher bandwidth is not significant. The power spectrum in figure 5.7 was used in this study to generate error functions for the case of single jump.

Assuming the process to be a zero-mean, real-valued, stationary Gaussian process ***E(t)***, and using the one-sided power spectral density function ***G(ω)***, the subroutine ***SIMU***, adopted from Harichandran (1992), is used to generate samples (realizations) of ***E(t)*** using the following simulation technique, Soong (1992):

- Minimum and maximum frequencies are determined for the simulation such that the area under the one-sided power spectra could be approximated as:

$$\overline{G}(\omega) = G(\omega), \quad for\ 0 \le \omega \le \overline{\omega} \cdots\cdots\cdots \quad [5.15]$$

- The reduced power spectra is then partioned into non-overlapping intervals n of width $\Delta\omega_n$ and amplitudes S_n determined from the one-sided reduced spectra.
- The variance σ^2_n for each interval is calculated as the area under the spectra, Figure 5.7.
- A series of independent random variables Φ_m are generated to be used in the Fast Fourier Transform algorithm, ***FFT***, to obtain a sample of size m of the estimated error function. The variable Φ_m is distributed uniformly over the range $(0, 2\pi)$.

- The approximate sample (realization) of the error process $E(t)$ is then generated

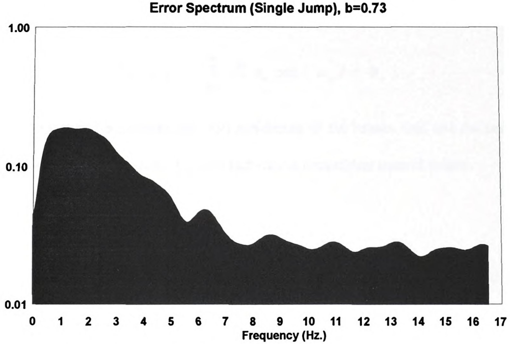

Figure 5.7 Single Jump Error Function, b = 0.73 Hz.

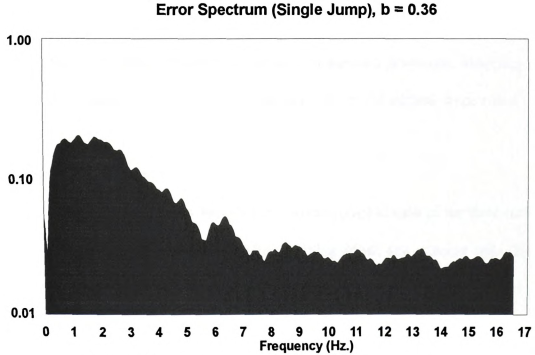

Figure 5.8 Single Jump Error Function, b = 0.36 Hz.

using the ***SIMU*** subroutine which applies a Fast Fourier Transformation on the following stochastic process, Soong (1992) to obtain the error function:

$$Z_m (t) = \sum_{n=1}^{m} \sqrt{2}\, \sigma_n \cos (\omega_n t + \Phi_n) \cdots\cdots\cdots \quad [5.16]$$

- The process is repeated for each simulation of the human load and the resulting error function is added to lines and curves connecting control points.

Horizontal Force Ratios

Horizontal force ratios were estimated as correlated random variables based on the measured data statistics. For each horizontal component (parallel and perpendicular to seating), the maximum and minimum force ratios were computed. Arrival times to these force ratios were determined as correlated random variables to the arrival time of the maximum force in the vertical direction. Amplitudes of the force ratios were found from the respective probability distributions, means, and standard deviations. Modeling error was established using the same process explained earlier for vertical force ratios.

5.3.2 Load Simulation

The procedure explained in Section 5.3.1 were applied to each of the three transient motions considered in this study (single jump, sitting down, and standing up). Typical screen views of ***HLS*** output were shown earlier in figures 5.2 and 5.3. For each simulation, the program provides the dynamic force ratios in the three principal axes of motion along with the control points and the peak amplitudes of the sample.

For each motion type, *HLS* was used to simulate an ensemble of 1000-samples of the motion and mean, standard deviation, and probability distributions of the control points of the ensemble were estimated. These values were then compared to the ensemble of measured data. Tables 5.1 through 5.3 present these statistics and show the absolute difference between the measured and simulated values as percentages of the measured data. The tables demonstrate that the simulated samples converged to the measured data with differences of no more than 5% of the measured data.

5.4 Periodic Load Modeling

Periodic loads are modeled in *HLS,* in a process consists of five stages

- **a)** Input to the program provides information needed to determine the type of periodic action, prompt or stimulus frequency, number of participants, and type

Table 5.1 Measured Versus Simulated Control Points for Single Jump

Single	NO	PHI	T2	T3	T4	T5	T6	T7	T8
Mean Arrival Time	Measured	0.277	0.098	0.319	0.452	0.818	0.903	1.192	1.373
	Simulation	0.277	0.1	0.323	0.459	0.829	0.915	1.208	1.389
	Difference	0.20%	1.60%	1.40%	1.40%	1.30%	1.30%	1.40%	1.20%
St. Dev Arrival Times	Measured	0.086	0.041	0.106	0.117	0.192	0.187	0.262	0.286
	Simulation	0.088	0.04	0.106	0.116	0.188	0.183	0.249	0.273
	Difference	2.00%	2.90%	0.20%	1.30%	2.10%	1.90%	5.00%	4.40%
		A1	A2	A3	A4	A5	A6	A7	A8
Mean Amplit	Measured	-0.062	-0.519	1.134	-0.997	-0.991	3.155	-0.427	0.051
	Simulation	-0.06	-0.516	1.144	-0.997	-0.992	3.22	-0.43	0.051
	Difference	3.70%	0.50%	0.90%	0.00%	0.10%	2.00%	0.70%	0.90%
St. Dev Amplit.	Measured	0.063	0.196	0.459	0.069	0.055	1.372	0.182	0.134
	Simulation	0.063	0.2	0.458	0.067	0.054	1.409	0.186	0.132
	Difference	0.40%	2.00%	0.10%	2.40%	1.40%	2.70%	2.40%	1.30%

Table 5.2 Measured Versus Simulated Control Points For Sitting Down

Mean Arrival Time	Measured	0.252	0.245	0.463	0.533	0.603	0.748	0.862
	Simulatio	0.254	0.243	0.456	0.53	0.6	0.745	0.858
	Difference	0.60%	0.80%	1.40%	0.60%	0.40%	0.30%	0.50%
St. Dev. Arrival Time	Measured	0.107	0.102	0.135	0.15	0.155	0.169	0.172
	Simulatio	0.111	0.101	0.136	0.149	0.154	0.167	0.169
	Difference	3.60%	1.00%	0.80%	0.80%	0.80%	1.30%	1.50%
		A1	A2	A1	A4	A5	A6	A7
Mean Amplit.	Measured	0.023	-0.759	0.83	0.861	2.118	-0.326	0.027
	Simulatio	0.023	-0.769	0.826	0.857	2.087	-0.323	0.024
	Difference	1.40%	1.30%	0.50%	0.40%	1.40%	0.90%	11.10%
St. Dev. Amplit.	Measured	0.064	0.594	0.306	0.291	0.737	0.227	0.131
	Simulatio	0.067	0.596	0.291	0.277	0.697	0.228	0.134
	Difference	4.60%	0.30%	4.80%	4.70%	5.40%	0.40%	1.90%

of output media (printer, screen, or files).

- **b)** The program reads parameter files which contains control points statistics, phase lag parameters, model error spectra, and the seed array for the random number generator.

- **c)** *HLS* then calls the appropriate subroutines for the selected actions to determine the force-time history, and computes peak statistics for the selected motion. The model error function as a random process is generated at this stage and added to the force-time histories.

- **d)** The one-sided power spectra for the simulated dynamic force ratios is computed and plotted to the screen.

- **e)** Calculated individual force-time histories along with the group force time history, if requested, are sent to the requested output media at this stage. The program is set up to plot force-time histories and spectra to the screen. Other important statistics are saved in a summary file.

The stochastic approach to develop load models for periodic motions is explained. in the next section. A first order linear Markov process is used to approximate the arrival time and amplitude peak of control points. The process as an autoregressive function is explained in detail for the periodic jumping motion. Application of the process is then extended to other motion types.

5.4.1 Stochastic Analysis

Statistical analyses of measured periodic loadings revealed a consistent pattern of

Table 5.3 Measured Versus Simulated Control Points For Standing up

			PHI	T2	T3	T4	T5	T6
Arrival Times	**Mean**	Measured	0.291	0.229	0.528	0.608	0.857	1.003
		Simulation	0.298	0.231	0.529	0.611	0.865	1.009
		Difference	2.30%	1.00%	0.30%	0.50%	0.90%	0.60%
	St. Dev.	Measured	0.17	0.12	0.181	0.195	0.37	0.379
		Simulation	0.173	0.118	0.175	0.192	0.377	0.379
		Difference	1.90%	1.90%	3.50%	1.30%	2.00%	0.00%
			A1	A2	A1	A4	A5	A6
Amplitudes	**Mean**	Measured	0.005	0.981	-0.66	-0.63	0.744	0.029
		Simulation	0.005	0.986	-0.672	-0.644	0.73	0.026
		Difference	8.70%	0.50%	1.90%	2.20%	1.90%	11.10%
	St. Dev.	Measured	0.016	0.33	0.35	0.39	0.268	0.088
		Simulation	0.016	0.328	0.347	0.386	0.264	0.092
		Difference	0.30%	0.70%	0.90%	1.10%	1.60%	5.00%

arrival times and amplitude peaks of cycles within each recorded sample.

For arrival times, It was observed that participants always attempted to adjust their motion period to match that of the prompt. This behavior was shown in Section 4.4.1 to be consistent from one individual to another and suggests some dependency between the period of the current cycle and the next one. If the probabilistic behavior at the

present, given all of the past behavior, depends on only the most recent past, the process is called ***A Markov process***. Understanding the phenomena of human jumping would further suggests that the motion in a present cycle may be approximated by considering only the previous cycle.

The process X is considered a ***first order linear Markov process***, or a first order autoregressive process if it satisfies the difference equation:

$$X_t - a\, X_{t-1} = \zeta\,(\,t\,) \qquad [5.17]$$

where a: is a constant and $\zeta(t)$ is a stationary purely random process to represent the "random disturbance" portion of the process, Priestley (1981). If the jumping period of a participant is assumed to be a first order Markov process , then Equation [38] could be rewritten as:

$$T_n - a\, T_{n-1} = \zeta\,(\,n\,) \qquad [5.18]$$

where T_n is the period at cycle n. Equation [5.18] could be interpreted as a linear regression equation to estimate the period T at cycle n knowing the value of the period at cycle $n\text{-}1$.

$$T_n = a\, T_{n-1} + \zeta\,(\,n\,) \qquad [5.19]$$

The process is then simplified by assuming initially conditions the period has a value of zero and Eq. [5.19] will results in :

$$T_1 = \zeta\,(\,1\,) \qquad [5.20]$$

and Eq. [5.19] is then rewritten as:

$$T_n = \zeta_n + a\,(\,\zeta_{n-1} + a\,T_{n-2}\,) \cdots\cdots\cdots\cdots \quad [5.21]$$

$$T_n = \zeta_n + a\,\zeta_{n-1} + a^2\,\zeta_{n-2} + a^3\,\zeta_{n-3} + \ldots\ldots + a^{n-1}\,T_1 \cdots\cdot \quad [5.22]$$

and using Eq. [5.20]:

$$T_n = \zeta_n + a\,\zeta_{n-1} + a^2\,\zeta_{n-2} + \ldots\ldots + a^{n-1}\,\zeta_1 \cdots\cdots \quad [5.23]$$

and if $E(T_n) = \mu_\zeta$, then

$$E\,(\,T_n\,) = \mu_\zeta\,(\,a + a^2 + \ldots\ldots + a^{n-1}\,) \cdots\cdots\cdots \quad [5.24]$$

$$E\,(\,T_n\,) = \mu_\zeta\,(\,\frac{1 - a^n}{1 - a}\,) \qquad for\ \ a \neq 1 \cdots\cdots\cdots \quad [5.25]$$

and $E(T_n) = \mu_\zeta\, n$ for $a = 1$. If the mean of the disturbance function is zero, then the expected value of the period is zero and does not depend on the cycle number; i.e a stationary process up to the first order. If the mean of the disturbance function is not zero, then the expected value of the period will be a function of the cycle number and the process is not stationary. However, if the absolute value of the constant coefficient a is less than zero, it can be proved that for large cycle numbers n, the process T_n will be "asymptotically stationary" to the first order and computed as follows, Priestley, (1981):

$$E\,(\,T_n\,) = \frac{\mu_\zeta}{1 - a} \qquad for\ large\ n\ , \quad |\,a\,| < 1 \cdots\cdots \quad [5.26]$$

A similar process is then repeated for evaluating the variance and covariance of the process $T(n)$, and the following equations are obtained:

$$cov\ (T_n, T_{n+r}) = E\ (T_n\ T_{n+r}) \approx \sigma_\zeta^2 \frac{a^r}{1-a^2} \quad for\ \ |a| < 1. \quad [5.27]$$

and it can be said that the process T_n will be "asymptotically stationary" to the second

$$\sigma^2{}_T \approx \sigma_\zeta{}^2\, n \qquad for\ \ |a| = 1 \ldots\ldots \quad [5.28]$$

$$\sigma^2{}_T \approx \sigma_\zeta \left(\frac{1-a^{2n}}{1-a^2} \right) \qquad for\ \ |a| \neq 1 \ldots\ldots \quad [5.29]$$

order given the absolute value of a is less than one. The autocorrelation function for the process is thus:

$$R(r) = a^{|r|}, \quad r = 0 \pm 1 \pm 2 \pm 3 \pm \ldots\ldots \quad [5.30]$$

which decays to zero exponentially as $|r|$ increases.

The outlined procedure from Eq. [5.17] through [5.30] is also applied to amplitude peaks which was observed to have the same pattern of the ***Markov property***. The process of preparing and analyzing measured data consisted of the following steps:

- The CONTROL.FOR program was used to determine the period and peak ratio of every cycle of each sample in the ensemble of measured data. The program reports the value of the current and last period and peak ratio for each cycle in the periodic process. Table 5.4 shows a typical sample of this process for a periodic jumping test 2 Hz for one individual.
- The constant value a was estimated from each sample in the ensemble of measured data for the considered motion using correlation program CORREL.

Two values of the constant a are estimated from Eq. [5.28]. These are a_T, *and* a_P for the arrival times T and peaks P which are the random processes of each sample.

- The value of the disturbance function $\zeta(n)$ associated with each process was also computed and analyzed for determining its statistical properties.
- The mean and the standard deviation of a_T and a_P were calculated over the ensemble of measured data to determine the probability distribution function that best-fit the variables.

Table 5.5 shows the statistics of the constants a_T and a_P for the case of periodic jumping at frequencies 2, 3, and 4 Hz. It can be seen from the values in the table that the positive correlation between each cycle and the cycle before it was stronger for the 2 Hz. jumping in comparison with the 3 and 4 Hz. jumps. This was found to be valid for both the period and the peak ratios and it also verifies the observation made earlier in chapter 4.0 about the inability of participants to adjust to the prompt at Integer frequencies higher than 2 Hz.

5.4.3 Load Modeling

The periodic motions were modeled in *HLS*, as first order linear autoregressive processes where the period the amplitude peak of each cycle are generated from the procedure explained earlier. For each periodic motion considered in the study (Jumping, jouncing, and side swaying) the program provides the dynamic force ratios as a time series and computes the spectral density function for the simulated motion.

Table 5.4 Periods and Peaks for a Typical Periodic Jumping

Cycle	T(i)	T(i-1)	P(i)	p(i-1)
1	0.471	0.000	3.371	0.000
2	0.558	0.471	3.001	3.371
3	0.559	0.558	2.891	3.006
4	0.530	0.559	3.171	2.891
5	0.559	0.530	3.594	3.171
6	0.500	0.559	4.535	3.595
7	0.558	0.500	3.354	4.536
8	0.530	0.558	4.533	3.544
9	0.529	0.530	4.200	4.532
10	0265	0.529	4.418	4.200
11	0559	0.265	3.331	4.418
12	0.294	0.559	-1.023	3.330
13	0.559	0.294	3.854	-1.027
14	0.530	0.559	3.918	3.854
15	0.500	0.530	4.357	3.199
16	0.558	0.500	2.883	4.357
17	0.265	0.558	3.209	2.883
18	0.265	0.265	-0.935	3.209
19	0.529	0.265	3.615	-0.935
20	0.559	0.529	3.523	3.615
21	0.529	0.559	3.728	3.523
22	0.559	0.529	2.756	3.728
23	0.559	0.559	3.265	2.756

For each periodic motion, three ensembles of 1000-samples of the motion were simulated at the frequencies 2, 3, and 4 Hz. The mean and the standard deviation of peak force ratios and motion period were compared with measured data statistics to verify

Table 5.5 Constants a_p and a_T for the Periodic Motions

	2 Hz.		3 Hz.		4 Hz.	
	Period	Peak	Period	Peak	Period	Peak
Mean	0.251	0.392	0.145	0.281	0.049	0.161
St. Dev.	0.393	0.313	0.279	0.272	0.243	0.184

the proposed model. Table 5.6 presents these statistics and shows the absolute difference between the measured and simulated values as percentages of the measured data. The table demonstrates that the simulated samples converged to the measured data with differences of no more than 1% of the mean measured data. Simulated data from all motions showed smaller standard deviations than the measured data. This is due to the fact that in measured data there were always cases where some individuals chose to end their motions earlier than others in the same group.

Table 5.6 Measured Versus Simulated Data (Periodic Jumping)

	2 Hz.		3 Hz.		4 Hz.	
	Mean	**St. Dev.**	**Mean**	**St. Dev.**	**Mean**	**St. Dev.**
Measured	2.273	0.705	2.054	0.472	2.002	0.403
Simulated	2.287	0.387	2.15	0.151	2.003	0.103
Difference	0.62%	-45.11%	-0.18%	-68.01%	0.05%	-74.44%

6.0 Model Applications

6.1 Introduction

This research provides an accurate model of loads due to several human actions. The model could be used to verify existing code values, provide a tool for use in reliability-based designs, and study the effects of human loads on various special structures. Modeling Transient action would be useful in the design of elements that support mainly transient loads such as spring boards and staircases steps. Periodic loads are more likely to govern in the vibration prevention design stage especially if the loading frequency is close to the natural frequency of the structure. It has been confirmed that periodic loadings are major contributors to most of the documented cases of human discomfort due to vibration of assembly structures, Bachmann (1988). It has also been observed and verified that periodic motions would produce the maximum forces on structures when compared with transient and random motions produced by the same group of persons.

It is expected that the research conclusions would provide a basic tool to be used in changing building codes to provide designs which are not only safe against catastrophic failure but serve their purpose without harmful or annoying vibrations or deflections.

6.2 Group Load Simulation

A modified version of The Human Load Simulation program, *HLS* is used herein to generate loads for a group of participants and compare it with similar measured data.

The periodic loading module is used to illustrate the ability of the program to generate group loadings. The modified model, GROUP.FOR uses the procedure outlined in last chapter to generate periodic jumping for individuals. Force-time series from each individual is then added to a new array denoted here as the group force-time series. This process takes into account the fact that each individual would start the periodic motion at a different time (the response lag). Although the main focus in this process is to obtain the maximum force ratio of the group of individuals, the program does not attempt to align the peaks of different individuals. The assumption here is that every individual would jump independently and the overall force is the result of adding individual forces at each discrete time point in the time series. Statistics from measured data showed that the larger the group that performs the same action at the same time, the harder it is for individuals to maintain coherency and the lower the overall generated dynamic force-ratios, Table 3.7. Using the same weights from measured data, a simulation of 100-samples per group of individuals were conducted using GROUP.FOR.

Output showed that simulated mean peak forces were always lower than the measured data. Moreover, the ratio of the difference to the measured data was found to have a consistent pattern. Table 6.1 summarizes this finding. It shows the measured versus the simulated maximum peak force ratios for groups up to 40 participants.

The table illustrates the possibility of a non-linear correlation between the group size and the dynamic force-ratio generated by an individual within a group. This phenomena has been addressed in earlier stages of this project, Tuan (1981), and Mcdonald (1984). However, insufficient data hindered studying the pattern.

Although other potential factors may have contributed to the pattern shown in table 6.1, the "group factor" may still be the dominant part of the shown trend. It has been observed throughout the data collection process that individuals motions were always influenced by neighbors motion. Having active individuals within a group seemed to

Table 6.1 Measured Versus Simulated Maximum Force Ratios

Group	2	4	10	20	30	40
measured	2.86	2.75	2.17	1.49	1.44	0.79
Simulated	2.44	1.69	1.25	0.95	0.93	0.74
Difference	1.17	1.63	1.74	1.57	1.55	1.07

energize other individuals in the same group. It is important to emphasize however, that this conclusion has not been studied statistically to justify it due to the lack of sufficient measurement. The measuring devices utilized in this study were not designed to measure individual motions of participants in a group. The output force-time histories were always the recording of total motion on the device.

The program GROUP.FOR was modified to include a non-linear regression model to account for the trend explained earlier, and denoted as the "group factor". Screen output from the modified program are shown in Figures 6.1 and 6.2. The figures show that the simulated group of 10 individuals produced a maximum peak ratio of 1.87 while the simulation for 40 individuals produced a dynamic force ratio of 0.511. Since the simulated experiment starts with a response lag that was drawn from a distribution with a small coefficient of variation, high peaks have more probability of occurring at the first portion of the force-time series. The pattern was observed in the simulated data figures

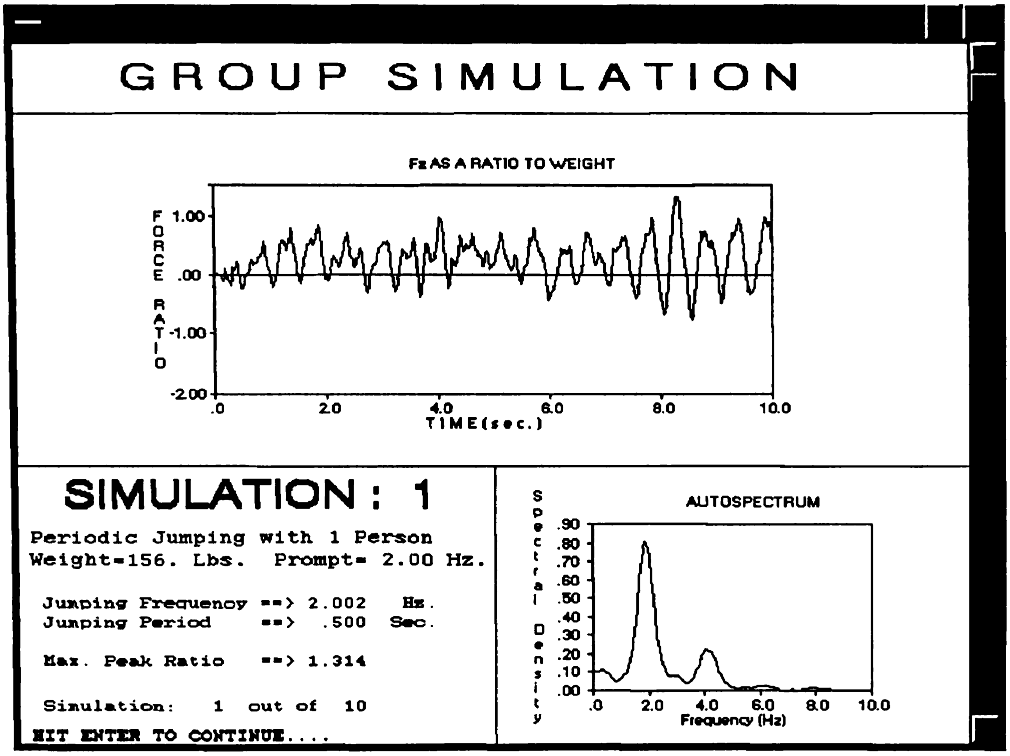

Figure 6.1 A Typical GROUP.FOR Output, 10 Individuals

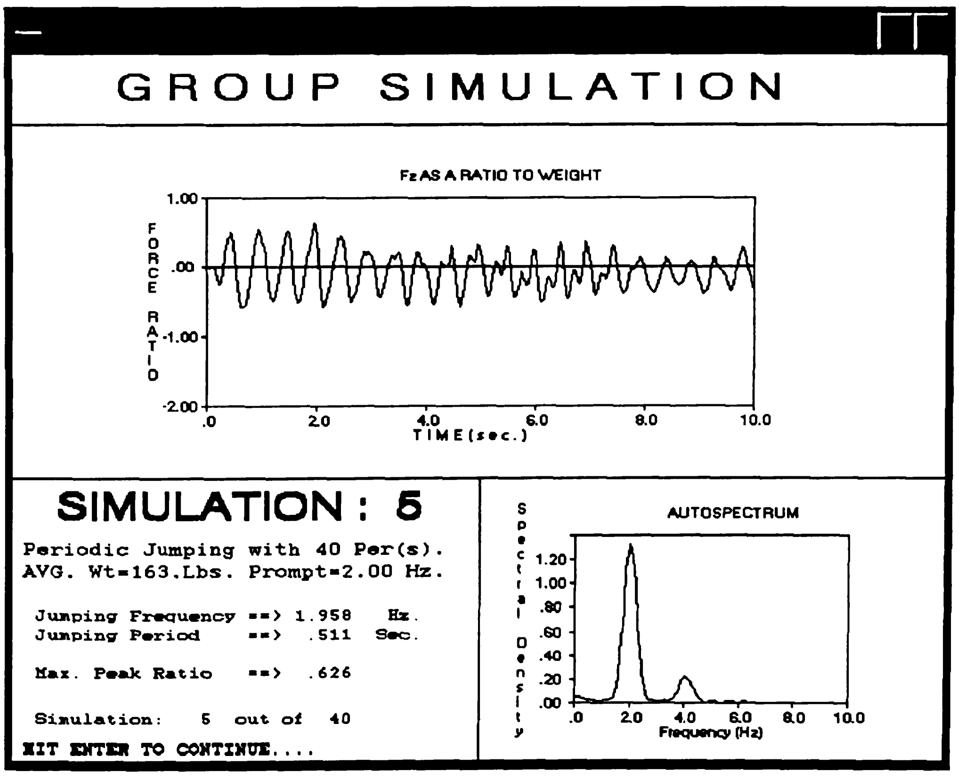

Figure 6.2 A Typical GROUP.FOR Output, 40 Individuals

6.1 and 6.2 reaffirm the observation Table 6.2 summarizes the modified simulated maximum peak ratios. Maximum difference between measured and simulated group loads

Table 6.2 Measured Versus Modified Simulated Maximum Force Ratios

Group	2	4	10	20	30	40
measured	2.86	2.75	2.17	1.49	1.44	0.79
Simulated	2.75	2.69	2.06	1.54	1.37	0.82
Ratio	1.04	1.02	1.05	0.97	1.05	0.96

was found to be less than 5%

6.3 Code Verification

The focus of this study was to define the force-time history produced by one individual or a group of individuals performing a predefined typical motion. Measured and simulated human loads in the research provide a limited sets of data in the much larger spectrum of activities, motions, frequencies, and participants composition. In order to transform these load values into structural design loads, other variabilities such as spatial and temporal variations must be considered. However, knowing the maximum force ranges produced from each motion simulated in the study may help in verifying current code values for assembly structures where human beings are the major source of live loads.

The highest loading effects were simulated using the Human Load Simulation program, *HLS* developed in this study. Statistics from the collected data confirmed that

periodic motions produced the highest vertical load ratios. Swaying side to side in a standing position presented the maximum force ratios in the horizontal direction and parallel to the rows of seats. Maximum transverse horizontal force ratios (perpendicular to the seating) was always associated with motions that produced maximum vertical force ratios. The single jump by an individual was found to produce the highest vertical and transverse force-ratios among all actions considered in the study (Periodic and Transient). Table 6.3 is a summary of the load ratios obtained from the current research. The log-normal distribution was found to be the best-fit distribution for all the dynamic force ratios in the table.

It is proposed here to consider two separate cases of loading conditions for code verifications. One case is for structures subjected mainly to the forces of one person. Typical examples would be spring boards, car seats, and steps of a stair case. The other case is assembly structures where dynamic forces are typically generated by a group of individuals. Dynamic force ratios in Table 6.3 could be utilized when designing structures from the first case. Group loading conditions will form the bases for suggesting dynamic load ratios for structures from the second case.

To illustrate the use of Table 6.3, a single jump by one individual is considered. A typical individual would generate a peak vertical dynamic force ratio of about 3.2. Assuming the individual occupies an area of 3.5 square feet, UBC (1982), and has an average weight of about 163 lb, Sidney (1979), the mean dynamic force would be about 150 lb/ft^2. Similarly, the mean transverse and parallel forces could be obtained from the table considering that the code assumes parallel and transverse forces to be linearly

distributed over a length of 28 inches. The mean dynamic forces would therefore be about 26 and 35 lb/ft for the parallel and transverse horizontal forces. The values in the vertical and the transverse directions differ significantly from the current code values of 100 lb/ft^2 for the vertical force and 10 lb/ft for the transverse force. This finding is affirmed by other studies Tuan (1981), and McDonald (1984). It is emphasized however that the suggested values are based on statistically limited data collection. Experimental data is still needed to arrive at more reliable estimates of load ratios.

Periodic motions would be utilized for load verification of the second type of structures; assembly structures where dynamic loads are typically produced by a group of individuals. It has been found that coherency among individuals performing transient actions is generally low and therefore it significantly reduces the overall group force-ratio.

Periodic jumping at about 2 Hz. was found to produce the largest vertical and transverse force ratios. The inability of individuals to maintain perfect coherency tends to reduce the group force ratios when more than one person attempts the action. It has been observed however, that a group effect exists in periodic motions where individuals may influence the motions of others in the same group. Taking both observations into account and reviewing the measured and simulated data, it is proposed to use the "Dynamic group factor" introduced in chapter 3.0 to modify the mean dynamic force ratios in Table 6.3. The Dynamic Group Factor, denoted as ***DF***, has the following form:

$$DF = 0.97^n * F_r \qquad [6.1]$$

where

F_r: Mean dynamic force ratio of one individual

n : Crowd size

Equation [6.1] should be used to reduce the mean dynamic force ratio for periodic actions in Table 6.3. The corrected force ratios could then be employed to estimate an upper bound to the dynamic force by a group of individuals performing any of the actions simulated in the study. When periodic jumping was used to verify code values, it was found that the code values would provide acceptable maximum ranges for both the vertical and the parallel components of the periodic motions. However, the transverse component was found to be significantly higher than the code value. It is therefore suggested to increase the load value of 10 lb/ft in the transverse direction to 45 lb/ft. Other literature showed the finding to be valid. Tuan (1983), suggested a value of 43 lb/ft for the same force component.

6.4 Reliability-Based Design

Generally, the reliability of any system is defined to be the probability of performing adequately for the period of time intended under the operating conditions encountered. The reliability of a structure is also defined as the probability that the structure will not exceed any of its limit states during a specified time period. In the case of human, one of the limit states loadings on assembly structures could be considered as the probability of not exceeding a certain vibration level. A certain value of acceleration may be another form of a limit state. Defining the statistical parameters of the dynamic human loads is therefore an important and essential ingredient in finding the overall reliability of the

Table 6.3 Dynamic Force Ratios for One Individual

Motions Considered			Transverse		Parallel		Vertical	
			Mean	Std.	Mean	Std.	Mean	Std.
Transient Actions	Single Jump		0.77	0.44	0.24	0.25	3.19	1.34
	Sitting Down		0.31	1.08	0.10	0.08	2.15	1.74
	Standing Up		0.53	0.41	0.17	0.17	1.47	1.23
Periodic Actions	Frequency Range of 2-4 Hz.							
	Jumping	2 Hz.	0.48	0.41	0.19	0.14	2.87	0.81
		3 Hz.	0.43	0.27	0.18	0.12	2.87	0.66
		4 Hz.	0.46	0.20	0.14	0.05	2.42	0.52
	Jouncing	2 Hz.	0.21	0.15	0.02	0.03	1.04	0.34
		3 Hz.	0.23	0.13	0.07	0.03	1.37	0.48
		4 Hz.	0.24	0.12	0.09	0.04	1.34	0.48
	Swaying	2 Hz.	0.08	0.02	0.21	0.06	014	0.42
		3 Hz.	0.10	0.04	0.29	0.10	0.20	0.05
		4 Hz.	0.11	0.05	0.33	0.10	0.25	0.13

structure under analysis.

A currently used measure to describe structural reliability is the reliability index, ß. It represents the distance, in standard deviations, from the set of variable values that is most likely to occur to the set of variable values that is most likely to cause failure. The reliability index ß is directly proportional to the difference between the mean values of the load L and resistance R. where σ_L, σ_R are the standard deviations of load and resistance respectively. It should be noted that this formulation for the reliability index

is valid only if both of the load and resistance are normally distributed. Moreover, the same principles apply for other distributions.

$$\beta = \frac{R_m - L_m}{\sqrt{\sigma_r^2 + \sigma_L^2}} \qquad [6.2]$$

Although the reliability index does not directly indicate a value for the reliability of a structure, it can be used to indicate a lower bound on the reliability and also to serve as a relative measure for the level of safety of different structures. Beyond that, we can only state that, for a given structure, larger values of the reliability index represent greater reliabilities than smaller values.

The reliability index usually varies from 2.0 to 8.0. Analysis of ß values for specifications governing the design of buildings in the United States indicates that ß typically ranges from 3.0 to 4.0. These values are for members designed to resist gravity loads in situations where failures are ductile, Ellingwood (1980). It is however important to emphasize at this point that, whereas the reliability index (ß) is not a direct measure of the reliability, it is a useful comparative measure and can serve to evaluate the relative safety of various structures.

The advantage in simulating dynamic loads produced by human actions may greatly benefit the reliability design concepts. Knowing the load statistics enables a designer to simulate over all probable values for loads and resistance and then determine the acceptable level of safety. The load ratios developed in this study for several human movements could be used in design cases where the knowledge of load variabilities would

enhance the ability to assume the appropriate safety measures and reliability for the intended structure.

6.3 Other Potential Applications

It is anticipated that the results of this research will be very useful for a broad range of building industry. following are some potential examples:

Floor Vibrations

Modern structures which have slender and elegant elements can be very sensitive to vibrations. Increasing numbers of excessive floor vibration incidents caused by human movements have been reported recently, Buchmann (1992), and Sterareh (1992). The coincidence of the natural frequencies of the floor system with human movement frequency is the major cause of this problem. The correspondence of the structure's natural frequency with the frequency of a higher harmonic of the time history of the dynamic excitation has also been realized to lead to resonance. To reduce the effect of annoying vibrations on the serviceability of structures, several solutions have been suggested. Among them, are the modifications of the floor stiffeners or mass, increasing the system damping, and/or limiting the vibration amplitudes. The accurate representation of the dynamic forces applied on the structure is essential to develop a reliable solution to the excessive vibration problem.

Manufacturers of floor systems, timber, pre-stressed concrete, open-web steel joists, and other combinations of geometry and materials may be willing to conduct a computer

simulation to anticipate potential problems rather than doing a full scale test or risk an adverse design for prototype installation.

Safety Studies

Insurance companies concerned with adverse or potentially harmful loading due to human actions or occupancy, such as occurred at the disastrous Kansas City Hyatt sky bridges, could be interested in a tool which has the capability of simulating a situation in a computer prior to construction, or in another scenario, analyzing an occurrence so as to remedy a problem.

Education

The Human Load Simulation program, ***HLS***, could be used as an educational tool for graduate courses dealing with reliability design, random processes, theory of random vibrations, and dynamic of structures. The ability of the program to simulate several loading conditions (transient and Periodic) and to produce pseudorandom processes could help in illustrating many of the concepts and applications of these courses. An instructor could utilize the random number generation features in ***HLS*** to produce random variables, correlated variables, or Markov random processes for any application since the produced variables would be dimensionless.

7.0 CONCLUSIONS

7.1 Summary

In addition to many other possible applications, human loads are the primary concern in the design and analysis of assembly structures. Human loads vary from event to event in an apparent pattern and they can be shown to be predictable statistically. The focus of this research is to quantify the uncertainty associated with the loads due to human movements. Providing accurate models of probabilistic functions to represent forces due to several predefined human movements was the main objective of the study.

A detailed review of current state of knowledge in the field of modeling human loads is presented. The study also provides an overview of current practice in the design process and how the present U.S. codes characterize human loads on assembly structures.

Data taken from tests performed on a force platform and a floor system were obtained from three sets of experiments; two designed to measure the force due to the action of an individual and the third designed to measure the force due to the action of a group. Two different test devices were utilized. A 4 ft by 8 ft force platform was used for measuring the vertical and horizontal components of loads produced by individuals and small groups up to five participants. The other device was a 12 ft by 15 ft floor system designed to measure total vertical loads generated by a group of people of up to forty participants. The study includes six different in-place human movements grouped into two categories; transient motions which includes single jumps, standing up suddenly

from a sitting position, and dropping into a seat from a standing position; and periodic motions which includes periodic jumping, jouncing, and swaying side to side.

For each transient action in the study, a method based on passing straight or curved lines through control points that define the force-time history is used to simulate the forces due to human actions. Best-fit distributions for the control points were determined. An error function was developed and incorporated into the model to better represent the modeled actions.

Periodic loadings were analyzed as a series of impulses where the forcing function is divided into cycles each of which is a full period of motion considered. Statistics of each impulse in the force-time history were computed and analyzed for correlation and peak distribution. The motions were modeled as first order autoregressive processes.

An iterative model-building methodology whereby the stochastic models are related to actual time series data was conducted. Model verification was applied to the different motions considered in the study. A model to simulate group loading is provided. The developed forcing functions are compared with experimental data and feed back from the process is used to refine the forcing functions.

A computer program, Human Load Simulation, *HLS*; was developed giving researchers an advanced tool to use in design and analysis. The forcing functions developed by the program simulate several types of human movements. The program may be used to simplify generating load distributions for any combination of human loading conditions, similar to those presented in this research. Input to the program includes information about the type of action to be simulated, group size, and parameters

necessary to define the individual loads. The program calls an array of subroutines to calculate forcing functions for different human actions and these forcing functions are combined in any form requested by the analyst. Output of the program is based on the type of action and includes the following:

- Statistical parameters of the control points.
- Amplitudes of peaks and their arrival times.
- Spectral energy distributions for periodic motions.
- Force-time history for individuals.
- Group force histories (for group loading).

Several potential applications of the program developed from the study are presented. Code values were examined and recommendations regarding the transverse horizontal force is outlined. Other potential applications of the research outcome are also discussed.

7.2 Conclusions

This study indicates several conclusions with respect to the modeling of human load movements:

- Measured data showed that human movements can be reproduced using probabilistic procedures.
- Periodic motions produced the highest vertical load ratios.
- Swaying side to side in a standing position presented the maximum force ratios in the horizontal direction (parallel to the seatings).

- Maximum transverse horizontal force ratios (perpendicular to the seatings) were always associated with motions that produced maximum vertical force ratios.
- The single jump was found to produce the highest vertical and transverse force ratios among the actions considered in the study (periodic and transient).
- Log-normal distribution was found to be the best-fit distribution for all peak force ratios.
- The developed program ***HLS***, will simplify generating load distributions for any combination of human loading conditions, similar to those presented in this research.
- Code values were examined and found to be acceptable except the transverse force component where a value of 45 lb/ft is recommended to replace the current 10 lb/ft value.

7.3 Future Work

The study leads to the following areas of needed further work:

- The human load program developed in the study needs to be expanded to include other types of human loadings.
- More statistical work is needed to establish the model error for human loads provided by ***HLS***.
- A study is needed to understand the effect of area of occupancy, floor type, and coherency among participants on the overall force ratios of one individual in a

group. A state of the art force platform would enable measuring individual forcing functions for individuals in a group.

- Aerobic motions are interesting class of human motions which deserves to be studied and modeled. Higher jumping frequencies and higher degree of coherency are expected in this class of motions.
- Studies on the effect of the prompt frequencies on the output forcing functions is still needed. Available data are very limited in this perspective.
- Taking advantage of the developed software may raise other potential possibilities of using CAD interface systems to show the performed motions and generated forces in real-time.
- Additional work is needed to translate the results of this research into formats suitable for use in design specifications.

APPENDIX A

DATA PROCESSING PROGRAMS

DATA.FOR

```
$STORAGE:2
$LARGE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                            DATA.FOR                             *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *    APRIL 15, 1992                                               *
* *    MAY   05, 1992                                               *
* *    June  15, 1992                                               *
* *    June  20, 1992                 BASSEM K. KHAFAGI             *
* *    AUG   16, 1992                                               *
* *    Aug   24, 1992                                               *
* *    Aug   30, 1992                                               *
* *    Oct   5,  1992                 MICHIGAN STATE UNIVERSITY     *
* *                                   EAST LANSING, MI 48823        *
* *                                   U.S.A                         *
* *                                                                 *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ************************************************************** *
C                    ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT      ++++++++++++++++
C                    ++++++++++++++++++++++++++++++++++
C
C
C     PROGRAM DATA.FOR
      DIMENSION XVOLT(512,11), FX(512), FY(512), FZ(512)
      DIMENSION BVOLT(1,11), VOLT(512,11), PROMPT(512)
      DIMENSION TIME(512), PR(512), PRP(200)
      DIMENSION XCALIB(11),YCALIB(11),ZCALIB(11)
      DIMENSION XCALIBA(11),YCALIBA(11),ZCALIBA(11)
      DIMENSION XCALIBB(11),YCALIBB(11),ZCALIBB(11)
      DIMENSION FXP(512), FYP(512), FZP(512)
      DIMENSION FXmx(200), FYmx(200), FZmx(200),twgt(200)
      INTEGER WEIGHT(5),NAA(20)
      LOGICAL*1 XABS
      CHARACTER REPORT*12,INPUTS*40,JOB*40,dataset*1,TT*5,EXT*12,DP*1
      CHARACTER FNAME1*3,FNAME2*3,FNAME3*3,FNAME4*3,FNAME*12,OUTPUTS*40
      CHARACTER Fileflag*2,fileflagn*2
      CHARACTER*8 NAME(20,200)
      COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
      COMMON /FILES/ INPUTS,OUTPUTS,LINP,LOUT
C
C       FLAG FOR THE OPEN SCREEN
C
      DP='y'
      OFLAG=1
C
C Read INPUT Data from PARAMETER FILE "PARAM.DAT"
C
      OPEN (1,FILE='PARAM.DAT')
      READ (1,*) BW
      READ (1,4) JOB
      READ (1,7) INPUTS
      READ (1,7) OUTPUTS
      READ (1,*) N
 4    FORMAT(A2)
 7    FORMAT(A40)
 6    FORMAT(A1)
```

```
C
C      DETERMINE THE LOCATION OF INPUT AND OUTPUT FILES
C
C
C        1.0 INPUTS
C
      LINP=LEN TRIM(INPUTS)
      IF(LINP.NE.0) THEN
      INPUTS=INPUTS(1:LINP)//'\'
      LINP=LINP+1
      END IF
C
C        2.0 OUTPUTS
C
      LOUT=LEN TRIM(OUTPUTS)
      IF(LOUT.NE.0) THEN
      OUTPUTS=OUTPUTS(1:LOUT)//'\'
      LOUT=LOUT+1
      END IF
C
C     START FILES' PROCESSING
C
      READ (1,*) NG
      DO 5 LL=1,NG
      READ (1,*) NA
      NAA(LL)=NA
      DO 20 I=1,NA
      READ (1,10) FNAME1,FNAME2,FNAME3,DATASET,FNAME4
 10   FORMAT (A2,A1,A1,A1,A3)
 20   WRITE (NAME(LL,I),30) FNAME1,FNAME2,FNAME3,DATASET,FNAME4
 30   FORMAT (A2,A1,A1,A1,A3)
        IF (LL.EQ.1) THEN
            WRITE (EXT,9)JOB,fname2
9           FORMAT (A2,'-',a1,'-SUM.OUT')
            OPEN (7,FILE=OUTPUTS(1:LOUT)//EXT)
        END IF
  5   CONTINUE
C
C
C Read spectral estimation parameters
C
      READ (1,*) M
      READ (1,*) IWIN
      READ (1,*) L
      READ (1,*) NFFT
      READ (1,*) OF
      CLOSE (1)
C
C
C                 +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++     2. READING CALIBRATION FACTORS    +++++++++++++
C                 +++++++++++++++++++++++++++++++++++++++
C
C
C
C       READ THE CALIBRATION FACTORS FILES IN THE THREE DIRECIONS X,Y,Z
C       USE THESE FACTORS TO CALCULATE THE FORCES IN EACH DIRECTION BY
C       MULTIPLYING THE VOLT(I,J) * CALIB-X FOR THE X FORCES AND THE SAM
C       FOR THE OTHER TWO DIRECTIONS Y,Z
C
C       READ CALIBRATION FACTORS FOR THE X, Y, Z FOR DATA SET A,B
C
C
C     OPEN THE CALIBRATION FILES IN THE X, Y, Z DIRECTIONS.
C
C
      DO 35 NN=1,2
      IF (NN.EQ.1) THEN
      OPEN (3,FILE='X-CALIB.DS1')
      OPEN (4,FILE='Y-CALIB.DS1')
      OPEN (5,FILE='Z-CALIB.DS1')
      DO 40 J=1,11
      READ (3,50) XCALIBA(J)
      READ (4,50) YCALIBA(J)
      READ (5,50) ZCALIBA(J)
 40   CONTINUE
      CLOSE (3)
      CLOSE (4)
```

```
      CLOSE (5)
      ELSEIF (NN.EQ.2) THEN
      OPEN (3,FILE='X-CALIB.DS2')
      OPEN (4,FILE='Y-CALIB.DS2')
      OPEN (5,FILE='Z-CALIB.DS2')
      DO 45 J=1,11
      READ (3,50) XCALIBB(J)
      READ (4,50) YCALIBB(J)
      READ (5,50) ZCALIBB(J)
 45   CONTINUE
      CLOSE (3)
      CLOSE (4)
      CLOSE (5)
      END IF
 35   CONTINUE

 50   FORMAT (11(E18.3))
C
C     START THE FILES-PROCESSING LOOP
C
      DO 480 LL=1,NG
      NA=NAA(LL)
      DO 380 K=1,NA
C
C     E.G. THE LOOP CAN BE CHANGED TO K=1,5 IF WE WANT TO PROCESS 5 FILE
C
      WRITE (FNAME,60) NAME(LL,K)
 60   FORMAT (A8,'.RAW')
      WRITE (REPORT,70) FNAME
 70   FORMAT (A8,'.FRC')
      READ (FNAME,10) FNAME1,FNAME2,FNAME3,DATASET,FNAME4
c
c     determine the calibration files to use and the sampling rate for the test
c
      IF ((DATASET.EQ.'A').OR.(DATASET.EQ.'a')) THEN
      FS=37.
      DO 75 J=1,11
      XCALIB(J)=XCALIBA(J)
      YCALIB(J)=YCALIBA(J)
      ZCALIB(J)=ZCALIBA(J)
 75   CONTINUE
      ELSEIF ((DATASET.EQ.'B').OR.(DATASET.EQ.'b')) THEN
      FS=34.
      DO 85 J=1,11
      XCALIB(J)=XCALIBB(J)
      YCALIB(J)=YCALIBB(J)
      ZCALIB(J)=ZCALIBB(J)
 85   CONTINUE
      ENDIF
C
C     BEGIN BY OPENING FILES TO READ DATA
C
      OPEN (2,FILE=INPUTS(1:LINP)//FNAME)
      OPEN (6,FILE=OUTPUTS(1:LOUT)//REPORT)
      READ (2,80) (WEIGHT(I),I=1,5)
 80   FORMAT (/,5(I4)//)
C
C
C       Now add the weights of participants to get TWEIGHT
C
      TWEIGHT=0.0
      DO 90 I=1,5
      TWEIGHT=TWEIGHT+WEIGHT(I)
 90   CONTINUE
C
      IF((Fname2.eq.'t').or.(Fname2.eq.'T').or.(Fname2.eq.'R').OR.
     1(Fname2.eq.'r')) N=256
C
      DO 100 I=1,N
      READ (2,110) (XVOLT(I,J),J=1,11),PROMPT(I)
 100  CONTINUE
 110  FORMAT (11(F8.0/),F6.0)
C
C
C                 ++++++++++++++++++++++++++++++++++++++
C     +++++++++++++     3. NORMALIZING VOLT. READINGDS    +++++++++++++
C                 ++++++++++++++++++++++++++++++++++++++
```

```
C
C
C          SELECT THE FIRST (OR THIRD) VOLTAGE READING AS THE BASE TO NORMALIZE
C          THE VOLTAGE READING AGAINST THE PART. WEIGHTS AND SAVE IT IN
C          MATRIX "BVOLT"
C
      DO 120 I=1,11
      IF ((DATASET.EQ.'A').OR.(DATASET.EQ.'a')) THEN
      BVOLT(1,I)=XVOLT(1,I)
      ELSE
      BVOLT(1,I)=XVOLT(3,I)
      END IF
 120  CONTINUE
C
C       DETERMINE THE TIME STEP USED IN COLLECTING DATA
C
      TIMESTEP=1.0/FS
C
C     NORMALIZE DATA BY SUBTRACTING EACH RAW OF DATA FROM THE FIRST
C     VLOTAGE READING WHICH REPRESENT THE STATIC WEIGHT OF THE FLOOR
C     AND PARTICIPANTS.
C
      DO 134 I=1,N
      DO 144 J=1,11
      VOLT(I,J)=0.0
      TIME(I)=0.0
      PR(I)=0.0
 144  CONTINUE
 134  CONTINUE
C
C
C
      DO 130 I=1,N
      DO 140 J=1,11
      VOLT(I,J)=XVOLT(I,J)-BVOLT(1,J)
      TIME(I)=TIME(I-1)+TIMESTEP
      PR(I)=ABS(PROMPT(I)-PROMPT(1))
      IF (I.EQ.1) THEN
      TIME(I)=0.0
      END IF
 140  CONTINUE
 130  CONTINUE
C
C                ++++++++++++++++++++++++++++++++++++++
C     +++++++++++  4. FINDING FINAL FORCES X, Y, Z , P  +++++++++++
C                ++++++++++++++++++++++++++++++++++++++
C
C
      DO 150 I=1,N
      FX(I)=0.0
      FY(I)=0.0
      FZ(I)=0.0
      FXP(I)=0.0
      FYP(I)=0.0
      FZP(I)=0.0
 150  CONTINUE
C
      DO 170 I=1,N
      DO 160 J=1,11
      FX(I)=FX(I)+VOLT(I,J)*XCALIB(J)
      FY(I)=FY(I)+VOLT(I,J)*YCALIB(J)
      FZ(I)=FZ(I)+VOLT(I,J)*ZCALIB(J)
 160  CONTINUE
c
c     SWITCH FX WITH FY FOR DATA SET A (TO MATCH DATA SET B COORD. SYSTEM)
C     aLSO TAKE THE WEIGHT OFF FROM THE FZ(I) FOR DATA SET A
c
      IF ((DATASET.EQ.'A').OR.(DATASET.EQ.'a')) THEN
      FXSW=FX(I)
      FYSW=FY(I)
      FX(I)=FYSW
      FY(I)=FXSW
      FZ(I)=FZ(I)-TWEIGHT
      ENDIF
C
C     FIND FORCE RATIO
C
```

```
      FXP(I)=FX(I)/TWEIGHT
      FYP(I)=FY(I)/TWEIGHT
      FZP(I)=FZ(I)/TWEIGHT
 170  CONTINUE
C
C
C       NOW FIND THE MAXIMUM VALUE OF THE FORCE FX,FY,FZ, AND USE EACH
C       AS A VALUE FOR THE PROMPT SPIKE.
C
C
      FXMAX=0.0
      FYMAX=0.0
      FZMAX=0.0
      FXMIN=0.0
      FYMIN=0.0
      FZMIN=0.0
      FXPMAX=0.0
      FYPMAX=0.0
      FZPMAX=0.0
      PRMAX=0.0
      TX=0.0
      TY=0.0
      TZ=0.0
      TP=0.0
      CALL MAX (FX,FXMAX,TX,TIMESTEP)
      CALL MAX (FY,FYMAX,TY,TIMESTEP)
      CALL MAX (FZ,FZMAX,TZ,TIMESTEP)
      CALL MAX (FXP,FXPMAX,TX,TIMESTEP)
      CALL MAX (FYP,FYPMAX,TY,TIMESTEP)
      CALL MAX (FZP,FZPMAX,TZ,TIMESTEP)
      CALL MIN (FX,FXMIN,TXM,TIMESTEP)
      CALL MIN (FY,FYMIN,TYM,TIMESTEP)
      CALL MIN (FZ,FZMIN,TZM,TIMESTEP)
      CALL MIN (FXP,FXPMIN,TXM,TIMESTEP)
      CALL MIN (FYP,FYPMIN,TYM,TIMESTEP)
      CALL MIN (FZP,FZPMIN,TZM,TIMESTEP)
C
C
C
C
C       FOR THE CASE OF SINGLE JUMP OR PERIODIEC JUMPING WHERE THE PERSON
C       LEAVE THE FLOOR, IT MAY BE POSIBLE TO ESTIMATE THE STATIC WEIGHT
C       OF THE PERSON(S).  THIS IS USEFUL IF THE WEIGHT IS UNKNOW OR AS
C       AS MEASURE OF DATA AQUISITION ACCURACY.
C       IT HAS BEEN FOUND THAT A PERSON LEAVES THE FLOOR WHEN JUMPING FOR
C       ABOUT 0.6 SECONDS (ABOUT 20 READINGS) AND COMES FOLLOWED DIRECTLY
C       BY THE FZMAX (THE HIT).
C
C
C
C
       EWEIGHT=0.0
      IF ((FNAME1.EQ.'SJ').OR.(FNAME1.EQ.'sj').OR.(FNAME1.EQ.'PJ').OR.
     1(FNAME1.EQ.'pj')) THEN
       EWEIGHT=-fzmin
       END IF
C
C       REPORT THE LOCATION OF THE PROMPT (IF ANY) FOR TRANSIENT ACTION
C
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      CALL MAX (PR,PRMAX,TP,TIMESTEP)
      DO 180 I=1,N
      PR(I)=0.0
 180  CONTINUE
      NP=INT(TP*FS)+1
      PRMAX=100.
      PR(NP)=PRMAX
      ELSE
C
C       FIND PEAKS FOR THE PROMPT FOR PERIODIC ACTION
C
C
      CALL MAX (PR,PRMAX,TP,TIMESTEP)
      CUTOFF=PRMAX/4.0
      NPK=0
      DO 190 I=1,N
```

```
      IF (PR(I).LE.CUTOFF) THEN
      PR(I)=0.0
      ELSE
      NPK=NPK+1
      PRP(NPK)=I
      PRMAX=100.
      PR(I)=PRMAX
      END IF
 190  CONTINUE
C
C         THIS LOOP IS TO ENSURE ONLY ONE READING PER PROMPT
C
      DO 210 Nz=1,NPK
      I=PRP(Nz)
      DO 200 NN=1,4
      PR(I+NN)=0.0
 200  CONTINUE
 210  CONTINUE
      END IF
C
C
C                  +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++      5. START THE FINAL OUTPUT        +++++++++++++
C                  +++++++++++++++++++++++++++++++++++++++
C
C       PRINT TIME VS FORCES IN X,Y,Z DIR. PLUS THE PROMPT
C
C
      WRITE (6,220)
 220  FORMAT (///,10X,'TIME  ',3X,'X-FORCE Ratio ',2X,'Y-FORCE Ratio',
     12X,'Z-FORCE Ratio',2X,'PROMPT',/)
      DO 230 I=1,N
      WRITE (6,240) TIME(I),FXP(I),FYP(I),FZP(I),PR(I)
 230  CONTINUE
 240  FORMAT (7X,F8.3,3(F14.6),3X,F10.0)
C
C
C     PRINT INFORMATION ABOUT THE FILE NAME TO THE SCREEN TO ENSURE THAT
C     CORRECT FILE IS BEING HANDLED
C
      WRITE (6,250) FNAME,FNAME3
 250  FORMAT (//,20X,' INPUT FILE INFORMATION',///,3X,'FILE NAME =  ',
     1A12,10X,'PARTICIPANTS =  ',A3,2X,'PERSONS.'/)
C
C       REPORT AN EXPLAINATION OF THE INFORMATION STORED IN THE FILE
C       THIS INFORMATION WILL BE PRINTED TO THE SCREEN AND SAVED IN
C       THE REPORT FILE (*.#RP)
C
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      WRITE (6,260) FNAME1,FNAME4
      ELSE
      WRITE (6,270) FNAME1,FNAME4
      END IF
 260  FORMAT (3X,'TRANSIENT ACTION =  ',A3,5X,'TEST No. = ',A3/)
 270  FORMAT (3X,'PERIODIC ACTION  =  ',A3,5X,'TEST No. = ',A3/)
C
C        THEN, PRINT OUTPUT SUMMARY
C
      WRITE (6,280) (WEIGHT(I),I=1,5),TWEIGHT,EWEIGHT
 280  FORMAT (//,1X,'WEIGHT(S) :',5(I5),3X,'Total :',F7.0,5X,' Estimate
     1:',F7.2/)
C
C       FINALLY, WRITE A SUMMARY ABOUT THE FORCES.
C
      WRITE (6,290) REPORT
 290  FORMAT (20X,'     FORCES INFORMATION ',///,13X,'FINAL REPORT FILE N
     1AME =  ',A12///)
      WRITE (6,300) FXMAX,FXPMAX,TX
      WRITE (6,310) FYMAX,FYPMAX,TY
      WRITE (6,320) FZMAX,FZPMAX,TZ
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      WRITE (6,330) PRMAX,TP
      ELSE
      WRITE (6,340) PRMAX
      END IF
C
C
```

```
 300  FORMAT (2X,'FXMAX =',F12.4,' LBS    RATIO TO WEIGHT=',F8.4,8X,'AT T
     1IME = ',F5.3,' SEC.',/)
 310  FORMAT (2X,'FYMAX =',F12.4,' LBS    RATIO TO WEIGHT=',F8.4,8X,'AT T
     1IME = ',F5.3,' SEC.',/)
 320  FORMAT (2X,'FZMAX =',F12.4,' LBS    RATIO TO WEIGHT=',F8.4,8X,'AT T
     1IME = ',F5.3,' SEC.',/)
 330  FORMAT (2X,'PRMAX =',F12.2,'     ',26X,'AT TIME = ',F5.3,' SEC.',/)
 340  FORMAT (2X,'PRMAX =',F12.2,'     ',26X,'AT TIME =      PERIODIC',/)
C
C
C                +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++   6. PREPARING PLOTS TO THE SCREEN    +++++++++++++
C                +++++++++++++++++++++++++++++++++++++++
C
C
C       CALL THE OPENING MENU FOR FINAL ON-SCREEN OUTPUT(UNDER DESIGN)
C
C       THE OPEN FLAG WILL OPEN THE MENU ONLY THE FIRST TIME THE PROGRAM
C       THE COLOR FLAG WILL BE SET TO (0) OR
C       (1) DEPENDING ON THE ITERATION NO. THIS FLAG WILL INTERCHANGE
C       THE BACKGROUND COLOR.
C
C
      IF (OFLAG.GT.0.0) THEN
      IF((DP.EQ.'Y').OR.(DP.EQ.'y')) THEN
      CALL OPEN (OFLAG,BW)
      READ (*,*)
      END IF
      ELSE
      GO TO 350
      END IF
C
C
C       CALL PLOTXY SUBROUTINE TO PLOT THE TIME VERSUS THE FORCE IN THE
C       X, Y, Z DIRECTIONS.
350   CONTINUE
      IF((DP.EQ.'Y').OR.(DP.EQ.'y')) THEN
      CALL PLOTXY (TIME,FXP,FYP,FZP,PR,FXPMAX,FYPMAX,FZPMAX,NP,TX,TY,TZ,
     1WEIGHT,TWEIGHT,CFALG,K,NA,FXPMIN,FYPMIN,FZPMIN,TXM,TYM,N,FNAME,
     2EWEIGHT,BW)
      ELSE
      print *,'GROUP: ',ll,'FILE: ',k
      END IF
      CLOSE (2)
      CLOSE (6)
C
C
C          ++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  +++++++++++  7. SPECTRAL ANALYSIS ESTIMATION OF FORCE-SERIES ++++++++
C          ++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C       spectral analysis will be called out only if the action is
C       periodic.  Also if more than on action is being processed, the
c       average correlation and average spectrs will be disabled
c
      IF((Fname1.eq.'ss').or.(Fname1.eq.'SS').or.(Fname1.eq.'jn').or.
     1(Fname1.eq.'JN').or.(Fname1.eq.'PJ').or.(Fname1.eq.'pj')) THEN
       if(k.eq.1) then
       noavg=0
       fileflag=fname1
       endif
       fileflagn=fname1
       if(fileflagn.ne.fileflag)noavg=1
      CALL SPECTRA (LL,K,NA,NAME,OF,RMS,ZMEAN,BW,noavg)
      END IF
C
C
C          ++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  +++++++++++  8. Reporting max forces to file "SUMMARY.OUT"  ++++++++
C          ++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
c
c     Find absolute maximum forces and report them to the matix fxmx(i)
c
        fxmx(k)=fxpmax
        if(fxpmax.lt.abs(fxpmin)) fxmx(k)=abs(fxpmin)
        fymx(k)=fypmax
```

```
         if(fypmax.lt.abs(fypmin)) fymx(k)=abs(fypmin)
         fzmx(k)=fzpmax
         twgt(k)=tweight
C
C
C
        IF (K.EQ.1) THEN
        WRITE (7,360)fname3,FNAME2
        ENDIF
        WRITE (7,370) K,TWGT(K),FXMX(K),FYMX(K),FZMX(K)
  360   FORMAT(//,5X,'SUMMARY OF TESTS OF',A5,'  PERS.  AT ',a1,' Hz.',//,
       11X,'Test      Weight      Fxmax      Fymax      Fzmax')
  370   FORMAT (2X,I2,5X,F8.2,3(4X,F6.3))
C
C
C         DONE.....
C
C
c       print *,ll,k
  380   CONTINUE
C
C
C         CALCULATE AND REPORT IMPORTANT STATISTICS FOR THE OBSERVED ACTION
C         This is done only if the same action is considered
C
        if(noavg.eq.0)then
        XABS=.TRUE.
        CALL MYSTAT (TWGT,NA,WMIN,WMAX,NWIN,NWAX,WMEAN,WVAR,WSTD,WVX,
       1WSKE,WKUR,WSE,XABS)
        CALL MYSTAT (FXMX,NA,XMIN,XMAX,NXIN,NXAX,XMEAN,XVAR,XSTD,XVX,
       1XSKE,XKUR,XSE,XABS)
         CALL MYSTAT (FYMX,NA,YMIN,YMAX,NYIN,NYAX,YMEAN,YVAR,YSTD,YVX,
       1YSKE,YKUR,YSE,XABS)
         CALL MYSTAT (FZMX,NA,ZMIN,ZMAX,NZIN,NZAX,ZMEAN,ZVAR,ZSTD,ZVX,
       1ZSKE,ZKUR,ZSE,XABS)
c
c        BLANK LINE TO SEPARATE DATA
c
        WRITE (7,373)
 373    FORMAT (1X)
C
C         REPORT TO THE OUTPUT FILE
C
        TT='MEAN'
        WRITE (7,371) TT,WMEAN,XMEAN,YMEAN,ZMEAN
  371   FORMAT (1X,A5,1X,F10.2,3(2X,F8.3))
  372   FORMAT (1X,A5,1X,I10,3(2X,I8),3X,I11)
        TT='MIN'
        WRITE (7,371) TT,WMIN,XMIN,YMIN,ZMIN
        TT='MAX'
        WRITE (7,371) TT,WMAX,XMAX,YMAX,ZMAX
        TT='VAR'
        WRITE (7,371) TT,WVAR,XVAR,YVAR,ZVAR
        TT='STD'
        WRITE (7,371) TT,WSTD,XSTD,YSTD,ZSTD
        TT='C.O.V'
        WRITE (7,371) TT,WVX,XVX,YVX,ZVX
        TT='NMIN'
        WRITE (7,372) TT,NWIN,NXIN,NYIN,NZIN
        TT='NMAX'
        WRITE (7,372) TT,NWAX,NXAX,NYAX,NZAX
        TT='SKEWN'
        WRITE (7,371) TT,WSKE,XSKE,YSKE,ZSKE
        TT='KURTO'
        WRITE (7,371) TT,WKUR,XKUR,YKUR,ZKUR
        TT='ST-ER'
        WRITE (7,371) TT,WSE,XSE,YSE,ZSE
          end if
c
c
c
  480   CONTINUE
        CLOSE(7)
C
C         ALL DONE
C
          CALL ENDGRAPHICS()
```

```
      STOP
      END
C     END OF THE MAIN PROGRAM DATA-P1.FOR
C#######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                    *
* *                         SUBROUTINE OPEN                            *
* *                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                    *
* *   APRIL 15,1992        OPENING LOGO......                          *
* *                                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ***************************************************************** *
C
C
C                  ++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT      ++++++++++++++++
C                  ++++++++++++++++++++++++++++++++
C
C
      SUBROUTINE OPEN (OFLAG,BW)
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,y3plot,
     2PLOT2METHOD
      INTEGER*2 BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     1TITLECOLOR,NUMBER,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,PLOT3COLOR
      CHARACTER TITLE*280,XAXISTITLE*80,YAXISTITLE*80
C       INCLUDE 'PLOT.DIF'
C       INITIALISE THINGS FOR THE FIRST PLOT
      INITGRAPHICS=.TRUE.
      INITBORDER=.TRUE.
      INITWINDOW=.TRUE.
      INITAXES=.FALSE.
      INITFONTS=.TRUE.
      PLOTMETHOD=.TRUE.
      Y2PLOT=.FALSE.
      Y3PLOT=.FALSE.
      SCRONLY=.TRUE.
      FONTHEIGHT=28
      FONTWIDTH=16
C       SET UP THE FANCY COLORS
      BORDERCOLOR=14
      WINDOWCOLOR=15
      IF (BW.EQ.1) THEN
      BORDERCOLOR=0
      WINDOWCOLOR=15
      END IF
C       SET UP THE PLOT LIMITS
      XMIN=0.0
      XMAX=1.0
      YMIN=0.0
      YMAX=1.0
C       POSITION THIS PLOT AT THE TOP LEFT OF THE SCREEN
      XBOTTOMLEFT=0.0
      YBOTTOMLEFT=0.0
      XTOPRIGHT=1.0
      YTOPRIGHT=1.0
      TITLE='    **           D  A  T  A  .  F  O  R            **'
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1XDATA,YDATA,Y2DATA,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,
     2YMIN,YMAX,XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,
     3YTOPRIGHT,PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,
     4PLOTCOLOR,TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,
     5FONTHEIGHT,FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3,
     6BW,PLOT2METHOD)
C       WAIT FOR USER TO HIT RETURN
c       READ(*,*)
Cc      CALL ENDGRAPHICS()
      OFLAG=0
      RETURN
      END
C       END OF MODULE   SUBROUTINE OPENING
C#######################################################################
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *                    SUBROUTINE PLOTXY                              *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *   APRIL 15,1992        PLOTTING FORCE-TIME HISTORIES              *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ***************************************************************** *
C
C
C                 +++++++++++++++++++++++++++++++++
C     ++++++++++++++++     1. DATA INPUT/OUTPUT       ++++++++++++++++
C                 +++++++++++++++++++++++++++++++++
C
      SUBROUTINE PLOTXY (TIME,FX,FY,FZ,PR,FXMAX,FYMAX,FZMAX,NP,TX,TY,TZ,
     1WEIGHT,TWEIGHT,CFALG,K,NF,FXMIN,FYMIN,FZMIN,TXM,TYM,N,FNAME,
     2EWEIGHT,BW)
      DIMENSION TIME(512), FX(512), FY(512), FZ(512), PR(512)
      INTEGER WEIGHT(5)
C       INCLUDE 'PLOT.DIF'
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,y3plot,
     2PLOT2METHOD
      INTEGER*2 BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     1TITLECOLOR,NUMBER,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,PLOT3COLOR
      CHARACTER TITLE*280,XAXISTITLE*80,YAXISTITLE*80,FNAME1*3,FNAME2*3,
     1FNAME3*3,FNAME4*3,FNAME*12,TITLE1*80,TITLE2*80,TITLE3*80,TITLE4*
     280,TITLE5*80,TITLE6*80,TITLE7*80,TITLE8*80,TITLE9*80,TITLE10*80
      NUMBER=N
C       INITIALISE THINGS FOR THE FIRST PLOT
      INITGRAPHICS=.TRUE.
      INITBORDER=.TRUE.
      INITWINDOW=.TRUE.
      INITAXES=.TRUE.
      INITFONTS=.TRUE.
      PLOTMETHOD=.TRUE.
      PLOT2METHOD=.TRUE.
      TITLEONLY=.FALSE.
      SCRONLY=.FALSE.
      Y2PLOT=.TRUE.
      FONTHEIGHT=10
      FONTWIDTH=5
C       SET UP THE FANCY COLORS DEPENDING ON THE VALUE OF CFLAG (0 OR 1)
      BORDERCOLOR=14
      IF (CFLAG.EQ.0.0) THEN
      WINDOWCOLOR=8
      CFLAG=1.0
      ELSE
      WINDOWCOLOR=1
      CFLAG=0.0
      END IF
      AXISCOLOR=10
      LABELCOLOR=11
      PLOTCOLOR=14
      PLOT2COLOR=4
      PLOT3COLOR=4
      TITLECOLOR=15
      IF(BW.EQ.1.0) THEN
         BORDERCOLOR=3
         WINDOWCOLOR=15
         AXISCOLOR=0
         LABELCOLOR=0
         PLOTCOLOR=0
         PLOT2COLOR=1
         PLOT3COLOR=3
         TITLECOLOR=4
      END IF
C
C     Set file name Component
C
      READ (FNAME,10) FNAME1,FNAME2,FNAME3,FNAME4
 10   FORMAT (A2,A1,A1,1X,A3)
```

```
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. FORCE IN THE X-DIR      ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
C
C       SET UP THE PLOT LIMITS for the screen part only
C
C
      PRED=2.0
      XMIN=1.0
      XMAX=6.0
      XTICSPACE=2.0
C
C     Call scale subroutine to set Ymin, Ymax, Yticspace
C
      CALL SCALE (FXMIN,FXMAX,YMIN,YMAX,YTICSPACE)
C
C       POSITION THIS PLOT AT THE TOP LEFT OF THE SCREEN
C
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.50
      XTOPRIGHT=0.50
      YTOPRIGHT=1.00
C       GIVE IT A TITLE
      TITLE='FX AS A RATIO TO WEIGHT     '
      XAXISTITLE='    T I M E ( s e c . )'
      YAXISTITLE='FORCE RATIO'
C
C       PLOT THE SECOND SET OF DATA (THE PROMPT)
C
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      PR(NP)=FXMAX/PRED
      ELSE
      DO 20 I=1,N
      IF (PR(I).GT.0.0) PR(I)=FXMAX/PRED
 20   CONTINUE
      END IF
C
C
C       READY TO CALL PROGRAM PLOT.FOR
C
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,FX,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3,BW
     6,PLOT2METHOD)
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      2. FORCE IN THE Y-DIR      ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
C
C       SET UP THE PLOT LIMITS
C
C
      XMIN=1.0
      XMAX=6.0
      XTICSPACE=2.0
C
C     Call scale subroutine to set Ymin, Ymax, Yticspace
C
      CALL SCALE (FYMIN,FYMAX,YMIN,YMAX,YTICSPACE)
C
C       POSITION THIS PLOT AT THE TOP RIGHT OF THE SCREEN
C
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.50
      XTOPRIGHT=1.00
      YTOPRIGHT=1.00
C       GIVE IT A TITLE
      TITLE='FY AS A RATIO TO WEIGHT     '
      XAXISTITLE='    T I M E ( s e c . )'
```

```
      YAXISTITLE='FORCE RATIO'
C
C        Prevent a new screen
C
C
      INITGRAPHICS=.FALSE.
      INITFONTS=.FALSE.
      INITAXES=.TRUE.
C
C
C        PLOT THE SECOND SET OF DATA (THE PROMPT)
C
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      PR(NP)=FYMAX/PRED
      ELSE
      DO 30 I=1,N
      IF (PR(I).GT.0.0) PR(I)=FYMAX/PRED
 30   CONTINUE
      END IF
C
C
C        READY TO CALL PROGRAM PLOT.FOR
C
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,FY,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3,BW
     6,PLOT2METHOD)
C
C
C                     +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      3. FORCE IN THE Z-DIR      ++++++++++++++++
C                     +++++++++++++++++++++++++++++++++
C
      XMIN=1.0
      XMAX=6.0
      XTICSPACE=2.0
C
C     Call scale subroutine to set Ymin, Ymax, Yticspace
C
      CALL SCALE (FZMIN,FZMAX,YMIN,YMAX,YTICSPACE)
C
C        POSITION THIS PLOT AT THE BOTTOM LEFT OF THE SCREEN
C
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.00
      XTOPRIGHT=0.50
      YTOPRIGHT=0.50
C        GIVE IT A TITLE
      TITLE='FZ AS A RATIO TO WEIGHT       '
      XAXISTITLE='   T I M E ( s e c . )'
      YAXISTITLE='FORCE RATIO'
C
C
C        PLOT THE SECOND SET OF DATA (THE PROMPT)
C
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      PR(NP)=FZMAX/PRED
      ELSE
      DO 40 I=1,N
      IF (PR(I).GT.0.0) PR(I)=FZMAX/PRED
 40   CONTINUE
      END IF
C
C
C        READY TO CALL PROGRAM PLOT.FOR
C
      INITAXES=.TRUE.
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,FZ,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3,BW
     6,PLOT2METHOD)
```

```
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++       4. OUTPUT VALUES          ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
C       just plot the title on this one
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.00
      XTOPRIGHT=1.00
      YTOPRIGHT=0.50
C
      TITLE='                        FORCE-RATIOS                   '
      WRITE (TITLE1,50) FNAME,FNAME3
 50   FORMAT ('FILE = ',A12,2X,'WITH ',A2,2X,'PER.')
      IF ((FNAME2.EQ.'t').OR.(FNAME2.EQ.'T')) THEN
      WRITE (TITLE3,60) FNAME1,FNAME4
      ELSE
      WRITE (TITLE3,70) FNAME1,FNAME4
      END IF
 60   FORMAT ('TRANSIENT MOTION = ',A3,2X,'TEST = ',A3/)
 70   FORMAT ('PERIODIC MOTION = ',A3,2X,'TEST = ',A3/)
      WRITE (TITLE4,80) (WEIGHT(I),I=1,5)
 80   FORMAT ('Weights = ',5(I3,','))
      WRITE (TITLE5,90) TWEIGHT,EWEIGHT
 90   FORMAT ('Total= ',F4.0,' ESTIMATE= ',F7.2,' lbs')
C
C     Find the max. value of force in the x,y directions (absolute force
C
      IF (ABS(FXMIN).GT.ABS(FXMAX)) FXMAX=FXMIN
      IF (ABS(FYMIN).GT.ABS(FYMAX)) FYMAX=FYMIN
      IF (ABS(FXMIN).GT.ABS(FXMAX)) TX=TXM
      IF (ABS(FYMIN).GT.ABS(FYMAX)) TY=TYM
      WRITE (TITLE6,100) K,NF
      WRITE (TITLE7,110) FXMAX,TX
      WRITE (TITLE8,120) FYMAX,TY
      WRITE (TITLE9,130) FZMAX,TZ
      WRITE (TITLE10,140)
 100  FORMAT (I2,' of 'I3,2X,' FMAX        TIME')
 110  FORMAT ('X-dir',3X,F8.3,6X,F6.3)
 120  FORMAT ('Y-dir',3X,F8.3,6X,F6.3)
 130  FORMAT ('Z-dir',3X,F8.3,6X,F6.3)
 140  FORMAT ('HIT ENTER TO CONTINUE....')
      CALL OUTPUTS (XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,TITLE,
     1TITLE1,TITLE2,TITLE3,TITLE4,TITLE5,TITLE6,TITLE7,TITLE8,TITLE9,
     2TITLE10,CFLAG,BW)
C
C
C       WAIT FOR USER TO HIT RETURN ( only for testing the program
C       , otherwise let the program run with no stops)
C
      READ (*,*)
C     IF (K.EQ.NF) CALL ENDGRAPHICS ()
      RETURN
      END
C       END OF SUBROUTINE  PLOTXY
C####################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                        SUBROUTINE MAX                           *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *   MAY   05,1992         FINDING THE MAX. VALUE OF DATA          *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *************************************************************** *
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
      SUBROUTINE MAX (DATA,FMAX,T,TIMESTEP)
```

```
C
C      TO FIND THE MAX. FORCE
C
       DIMENSION DATA(512)
       T=0.0
       N=0
       FMAX=-1E14
       NMAX=0
       DO 10 I=1,512
       N=N+1
       IF (DATA(I).GT.FMAX) THEN
       FMAX=DATA(I)
       NMAX=N
       GO TO 10
       END IF
 10    CONTINUE
       T=REAL(NMAX)*TIMESTEP
       RETURN
       END
C        END OF SUBROUTINE  MAX
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                        SUBROUTINE MIN                           *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *   MAY   05,1992        FINDING THE MIN. VALUE OF DATA           *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *************************************************************** *
C
C
C                      +++++++++++++++++++++++++++++++++
C      +++++++++++++++      1. DATA INPUT/OUTPUT       +++++++++++++++
C                      +++++++++++++++++++++++++++++++++
C
       SUBROUTINE MIN (DATA,FMIN,T,TIMESTEP)
C
C      TO FIND THE MIN. FORCE
C
       DIMENSION DATA(512)
       T=0.0
       N=0
       FMIN=0.0
       NMAX=0
       DO 10 I=1,512
       N=N+1
       IF (DATA(I).LT.FMIN) THEN
       FMIN=DATA(I)
       NMAX=N
       GO TO 10
       END IF
 10    CONTINUE
       T=REAL(NMAX)*TIMESTEP
       RETURN
       END
C        END OF SUBROUTINE  MIN
C######################################################################
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                        SUBROUTINE outputs                       *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *   MAY   15,1992        Plotting the final values of Forces      *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        START OF MODULE outputs.for
       INCLUDE 'FGRAPH.FI'
```

```
      SUBROUTINE OUTPUTS (XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     1TITLE,TITLE1,TITLE2,TITLE3,TITLE4,TITLE5,TITLE6,TITLE7,TITLE8,
     2TITLE9,TITLE10,CFLAG,BW)
      INCLUDE 'FGRAPH.FD'
      INTEGER*2 DUMMY,XWIDTH,YHEIGHT,IX1,IY1,IX2,IY2,IX,IY,FONTHEIGHT,
     1FONTWIDTH
      REAL*4 XW1,YW1,XW2,YW2,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,
     1YTOPRIGHT,YT
      CHARACTER TITLE*80,FONTCOMMAND*10,FONTSTRING*7,FONTFILE*12,TITLE1*
     180,TITLE2*80,TITLE3*80,TITLE4*80,TITLE5*80,TITLE6*80,TITLE7*80,
     2TITLE8*80,TITLE9*80,TITLE10*80
        RECORD           / VIDEOCONFIG / SCREEN
        RECORD           / WXYCOORD /    WXY
        RECORD           / XYCOORD /     POSITION
      COMMON SCREEN
C         SET UP FONT FOR USE
      FONTFILE='HELVB.FON'
      FONTCOMMAND="T'HELV'"
      IF (REGISTERFONTS(FONTFILE).LT.0) THEN
      WRITE (*,10) FONTFILE
 10   FORMAT (' FONT FILE ',A12,' NOT FOUND IN PLOT')
      STOP
      END IF
      FONTHEIGHT=28
      FONTWIDTH=16
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
 20   FORMAT ('H',I2.2,'W',I2.2,'B')
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
C       GET SCREEN DIMENSIONS
      XWIDTH=SCREEN.NUMXPIXELS
      YHEIGHT=SCREEN.NUMYPIXELS
C       SET A VIEW PORT, WITH CORNER COORDINATES DEFINED AS A
C       FRACTION OF THE TOTAL SCREEN SIZE
C       (0.0,0.0) IS BOTTOM LEFT OF SCREEN
C       (1.0,1.0) IS TOP RIGHT OF SCREEN
C       TRANSLATE CORNER COORDINATES TO PIXEL COORDINATES
      IX1=INT(XBOTTOMLEFT*FLOAT(XWIDTH))
      IX2=INT(XTOPRIGHT*FLOAT(XWIDTH))
      IY1=YHEIGHT-INT(YTOPRIGHT*FLOAT(YHEIGHT))
      IY2=YHEIGHT-INT(YBOTTOMLEFT*FLOAT(YHEIGHT))
      CALL SETVIEWPORT (IX1,IY1,IX2,IY2)
C       CREATE A REAL COORDINATE WINDOW
      XW1=0.0
      YW1=0
      XW2=1.0
      YW2=1.0
      DUMMY=SETWINDOW(.TRUE.,XW1,YW1,XW2,YW2)
C
C       SET THE COLOR OF THE BACKGROUND WINDOW DEPENDING ON THE FLAG
C       "CFLAG" ... NOTE THE SETUP HAS TO BE OPPOSITE TO THE PLOTXY
C
      IF (CFLAG.EQ.1.0) THEN
      DUMMY=SETCOLOR(8)
      ELSE
      DUMMY=SETCOLOR(1)
      END IF
      IF(BW.EQ.1.0)DUMMY=SETCOLOR(15)
      DUMMY=RECTANGLE_W($GFILLINTERIOR,XW1,YW1,XW2,YW2)
C       DRAW A BORDER AROUND THE VIEWPORT
      DUMMY=SETCOLOR(14)
      IF(BW.EQ.1.0)DUMMY=SETCOLOR(3)
      DUMMY=RECTANGLE_W($GBORDER,XW1,YW1,XW2,YW2)
C
C       This is the screen design of the data output
C
C
      TITLELENGTH=50
C
C       POSITION THE TITLE CENTERED (MORE OR LESS) ABOVE GRAPH
C
      YT=0.9*(YW2+YW1)
      CALL MOVETO_W (0.9*(XW2+XW1),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY-FONTHEIGHT/2
```

```
C
C        enter the program name (Title: from the main program)
C
      DUMMY=SETCOLOR(15)
      CALL MOVETO (IX,IY,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY+2,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
C
C        CHANGE THE FONT TO A FIXED FONT TO CREATE THE FORCES' TABLE
C        HERE, COURIER FONTS WILL BE USED.
C
C
      FONTFILE='courb.fon'
      FONTCOMMAND="T'courier'"
      FONTHEIGHT=12
      FONTWIDTH=9
      IF (REGISTERFONTS(FONTFILE).LT.0) THEN
      WRITE (*,10) FONTFILE
      STOP
      END IF
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
C
      YT=0.8*(YW2+YW1)
      CALL MOVETO_W (0.73*(XW2+XW1),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY-FONTHEIGHT/2
C
      DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY+10,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      CALL MOVETO (IX,IY+25,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
      CALL MOVETO (IX,IY+40,POSITION)
      CALL OUTGTEXT (TITLE3(1:TITLELENGTH))
      CALL MOVETO (IX,IY+60,POSITION)
      CALL OUTGTEXT (TITLE4(1:TITLELENGTH))
      CALL MOVETO (IX,IY+75,POSITION)
      CALL OUTGTEXT (TITLE5(1:TITLELENGTH))
      IF(BW.EQ.1.0)GOTO 22
      DUMMY=SETCOLOR(11)
      CALL MOVETO (IX,IY+11,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      CALL MOVETO (IX,IY+26,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
      CALL MOVETO (IX,IY+41,POSITION)
      CALL OUTGTEXT (TITLE3(1:TITLELENGTH))
      DUMMY=SETCOLOR(10)
      CALL MOVETO (IX,IY+61,POSITION)
      CALL OUTGTEXT (TITLE4(1:TITLELENGTH))
      CALL MOVETO (IX,IY+76,POSITION)
      CALL OUTGTEXT (TITLE5(1:TITLELENGTH))
22    CONTINUE
C
C         change font size
C
      FONTHEIGHT=10
      FONTWIDTH=8
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
C
C        Filling the Table of Forces
C
C
      DUMMY=SETCOLOR(14)
      IF(BW.EQ.1.0)DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY+110,POSITION)
      CALL OUTGTEXT (TITLE6(1:TITLELENGTH))
      DUMMY=SETCOLOR(15)
      IF(BW.EQ.1.0)DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY+130,POSITION)
      CALL OUTGTEXT (TITLE7(1:TITLELENGTH))
```

```
      CALL MOVETO (IX,IY+145,POSITION)
      CALL OUTGTEXT (TITLE8(1:TITLELENGTH))
      CALL MOVETO (IX,IY+160,POSITION)
      CALL OUTGTEXT (TITLE9(1:TITLELENGTH))
      DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY+180,POSITION)
      CALL OUTGTEXT (TITLE10(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY+181,POSITION)
      CALL OUTGTEXT (TITLE10(1:TITLELENGTH))
C
C       ALL DONE
C
      CALL UNREGISTERFONTS ()
      RETURN
      END
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *                        SUBROUTINE SCALE                           *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *    AUG   22,1992        VERTICAL PLOT SCALE                       *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *************************************************************** *
C                 +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT      +++++++++++++++
C                 +++++++++++++++++++++++++++++++++
C
      SUBROUTINE SCALE (FMIN,FMAX,ST,FIN,TICK)
C-----------------------------------------------------------------------
C This routine determines the starting and ending values to be used in
C a full scale plot.
C OUTPUT : ST = Starting value;  FIN = Ending value;  TICK .
C
C THIS IS A MODIFIED VERSION OF A PROGRAM WRITTEN ORIGINALLY BY R. HARIC
C-----------------------------------------------------------------------
      REAL*8 U,C
      TT=ALOG10(FMAX-FMIN)
      MT=IRINT(TT)
      TT=10.**(TT-MT)
      IF (TT.LE.2) THEN
      TT=2.
      ELSE IF (TT.GT.5) THEN
      TT=10.
      ELSE
      TT=5.
      END IF
      TICK=TT*10.**(MT-1)
      U=1.00001*TICK
      C=DLOG10(U)
      IF (C.LT.0) THEN
      L=IDINT(C)-1
      ELSE
      L=IDINT(C)
      END IF
      C=DBLE(10.)**L
      TICK=C*IDNINT(U/C)
      N1=IRINT(FMIN/TICK)
      N2=IRINT(FMAX/TICK)+1
      ST=N1*TICK
      FIN=N2*TICK
      RETURN
      END
C######################################################################
C######################################################################
```

SPECTRA.FOR

```
$STORAGE:2
$LARGE
C######################################################################
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                  *
* *                    SUBROUTINE SPECTRA                            *
* *                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                  *
* *   AUG   22,1992      AUTOCORRELATION AND AUTOSPECTRA PLOTS       *
* *                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * *************************************************************** *
C
C
C                  +++++++++++++++++++++++++++++++++
C     ++++++++++++++++       1. DATA INPUT/OUTPUT       ++++++++++++++++
C                  +++++++++++++++++++++++++++++++++
C
C----------------------------------------------------------------------
C Description:
C       PROGRAM TO ESTIMATE THE AUTOCORRELATION, AUTOSPECTRA,
C       AVERAGE AUTOCORRELATIONC, AND AVERAGE AUTOSECTRUM OF
C       FORCE-TIME SERIES. IT IS A MODIFICATION OF A PROGRAM ORIGINALLY
C       WRITTEN BY:
C       Ronald S. Harichandran, and E. Vanmarke, "Space-Time
C       Variation of Earthquake Ground Motion", Research Report
C       R84-12, Dept. of Civil Engineering, MIT, 1984.
C----------------------------------------------------------------------
C Description of output:
C
C       The autocorrelation function and the one-sided, normalized (i.e.
C       area) autospectrum of each force-series are stored in files with
C       the same name as the force-series and with filetype .ACN and .AS
C       respectively. The average of all the autocorrelation functions a
C       of the normalized autospectra are stored in files named AVERAGE.
C       and AVERAGE.ASP; If autospectra are estimated, then a file name
C       SUMMARY.OUT, containing useful summary tables, is created.
C----------------------------------------------------------------------
C Description of input:
C       Input are received from a file named "PARAM.DAT" which contains:
C       (1) General Informatio about the run:
C               N          No. of Samples
C               FS         Sampling Rate
C       (2) For autospectral estimation:
C               NA         Number of force-series
C               Name(L,I)    Names of force-series (Each in a separate lin
C       (3) Input for autospectral computations:
C               M       Section size (must be a power of 2),  2 <= M <= 2
C               IWIN    Window type, 1 for rect. window, 2 for Hamming wi
C               L       Half-width of lag window, 2 <= L<=(M/2+1);
C               NFFT    FFT size used for the spectral estimate,  (2L-1)
C               OF      Output frequency (<= fs/2)
C----------------------------------------------------------------------
      SUBROUTINE SPECTRA (LL,K,NA,NAME,OF,RMS,XMEAN,BW,noavg)
      COMPLEX SXY(513)
      REAL SACORR(1025),SASPEC(513)
      REAL SX(513),CCF(1025),XA(1025),AXIS(1025),NULL(1)
      CHARACTER FILE1*12,C*80,XLAB*80,YLAB*80,INPUTS*40,OUTPUTS*40
      CHARACTER*8 NAME(20,200)
      COMMON /FILES/ INPUTS,OUTPUTS,LINP,LOUT
      COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
      COMMON /CMPD1/ MAXM
      COMMON /OUT1/ C
      COMMON /SM/ FMAX,NP
      DATA (SACORR(I),I=1,512)/512*0./,(SACORR(I),I=513,1025)/513*0./
      DATA (AXIS(I),I=1,1025)/1025*0./
      DATA SASPEC/513*0./
      FMAX=OF
      MAXM=1024
C
C Begin spectral estimation
```

```
C
      DF=FS/FLOAT(NFFT)
      NF=NFFT/2+1
      XA(1)=0.
      XA(2)=FMAX-DF
      CALL AUTOSCL (XA,2,FTICK,FST,FMAX)
      NFMAX=FMAX/DF+1.00001
      NOF=ANINT(OF/DF)+1
      IF (NOF.GT.NF) NOF=NF
C
C Compute the autocorrelation functions and autospectra.
C
      II=K
      OPEN (6,STATUS='OLD',FILE=OUTPUTS(1:LOUT)//NAME(LL,II)//'.FRC')
C
C       skip information lines at the top of the input file (5 lines)
C
      READ (6,10)
 10   FORMAT (5(/))
      CALL CMPSE (CCF,SXY,VAR,PA,XMEAN)
      CLOSE (6)
      VAR=SQRT(VAR)
      RMS=VAR
C
C     ADD THE MEAN TO THE PEAK (IT WAS REMOVED IN THE SPECTRUM CALC.)
C
      IF (PA.GT.0) THEN
      PA=PA+XMEAN
      ELSE
      PA=PA-XMEAN
      END IF
      DO 20 J=1,M/2+1
      XA(J)=J-1
 20   SACORR(J)=SACORR(J)+CCF(J)/NA
C
      WRITE (FILE1,30) NAME(LL,II),'.ACN'
 30   FORMAT (8A)
      C='AUTOCORRELATION      '
ccc   OPEN (8,FILE=OUTPUTS(1:LOUT)//FILE1)
ccc   CALL OUT (8,M/2+1,CCF,0.,1.)
      XLAB='Lag'
      YLAB='Autocorrelation'
      XTICK=0.
      YTICK=0.
      CALL AUTOSCL (XA,M/2+1,XTICK,XST,XFIN)
      CALL AUTOSCL (CCF,M/2+1,YTICK,YST,YFIN)
      CALL PLTS (0.,FLOAT(M/2+1),YST,1.,XTICK,YTICK,XLAB,YLAB,C,AXIS,XA,
     1CCF,M/2+1,1,BW)
C
      WRITE (FILE1,30) NAME(LL,II),'.ASP'
      C='AUTOSPECTRUM        '
      DO 40 J=1,NF
      SX(J)=CABS(SXY(J))
      XA(J)=(J-1)*DF
 40   SASPEC(J)=SASPEC(J)+SX(J)/NA
ccc   OPEN (9,FILE=OUTPUTS(1:LOUT)//FILE1)
ccc   CALL OUT (9,NOF,SX,0.,DF)
      XLAB='Frequency (Hz)'
      YLAB='Spectral Density'
C         YTICK = -2.
      YTICK=0.
      CALL AUTOSCL (SX,NFMAX,YTICK,YST,YFIN)
      CALL PLTS (0.,FMAX,YST,YFIN,FTICK,YTICK,XLAB,YLAB,C,AXIS,XA,SX,
     1NFMAX,2,BW)
C
C         call plts for the final information on the screen
C
      C='   S P E C T R A L    A N A L Y S I S'
      CALL PLTS (0.,1.,0.,1.,1.,1.,XLAB,YLAB,C,AXIS,NULL,NULL,1,5,BW)
      WRITE (C,50) NAME(LL,II),RMS,PA,XMEAN
 50   FORMAT (1X,'  FILE: ',A8,T25,'RMS:',F9.4,T42,'PEAK:',F7.2,T62,'MEA
     1N:',F7.2)
      CALL PLTS (0.,1.,0.,1.,1.,1.,XLAB,YLAB,C,AXIS,NULL,NULL,1,7,BW)
C
C Output average correlations and spectrum
C
      if(noavg.eq.0) then
```

```
      IF (II.EQ.NA) THEN
      FILE1='AVERAGE.ACN'
      C='AVERAGE CORRELATION        '
      OPEN (10,FILE=OUTPUTS(1:LOUT)//FILE1)
      CALL OUT (10,M/2+1,SACORR,0.,1.)
      DO 60 J=1,M/2+1
 60   XA(J)=J-1
      XLAB='Lag'
      YLAB='Autocorrelation'
      XTICK=0.
      YTICK=0.
      CALL AUTOSCL (XA,M/2+1,XTICK,XST,XFIN)
      CALL AUTOSCL (SACORR,M/2+1,YTICK,YST,YFIN)
      CALL PLTS (0.,FLOAT(M/2+1),YST,1.,XTICK,YTICK,XLAB,YLAB,C,AXIS,XA,
     1SACORR,M/2+1,3,BW)
C
      FILE1='AVERAGE.ASP'
      C='AVERAGE SPECTRUM        '
      OPEN (11,FILE=OUTPUTS(1:LOUT)//FILE1)
      CALL OUT (11,NF,SASPEC,0.,DF)
      DO 70 J=1,M/2+1
 70   XA(J)=(J-1)*DF
      XLAB='Frequency (Hz)'
      YLAB='Spectral Density'
      YTICK=0.
      CALL AUTOSCL (SASPEC,NFMAX,YTICK,YST,YFIN)
      CALL PLTS (0.,FMAX,YST,YFIN,FTICK,YTICK,XLAB,YLAB,C,AXIS,XA,
     1SASPEC,NFMAX,4,BW)
C
C         call plts for the final information on the screen
C
      C='  A V E R A G E    A N A L Y S I S'
      CALL PLTS (0.,1.,0.,1.,1.,1.,XLAB,YLAB,C,AXIS,NULL,NULL,1,6,BW)
      C=' E N D    O F    F I L E    P R O C E S S I N G'
      CALL PLTS (0.,1.,0.,1.,1.,1.,XLAB,YLAB,C,AXIS,NULL,NULL,1,8,BW)
      END IF
      END IF
C
      END
      SUBROUTINE OUT (IF,NN,X,XMIN,XINC)
C-----------------------------------------------------------------------
C This subroutine outputs an array to a file.
C-----------------------------------------------------------------------
      REAL X(*)
      CHARACTER*80 C
      COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
      COMMON /OUT1/ C
C
 10   FORMAT (A80/)
 20   FORMAT (' M =',I5,';  No. of Samples =',I5,';  Sampling Frequency
     1=',F6.1/' Window = Hamming',';  Window half-width =',I4,';  NFFT =
     2',I5)
 30   FORMAT (' M =',I5,';  No. of Samples =',I5,';  Sampling Frequency
     1=',F6.1/' Window = Rectangular',';  Window half-width =',I4,';  NF
     2FT =',I5)
C
      OPEN (IF,STATUS='UNKNOWN')
      WRITE (IF,10) C
      IF (IWIN.EQ.1) WRITE (IF,30) M,N,FS,L,NFFT
      IF (IWIN.EQ.2) WRITE (IF,20) M,N,FS,L,NFFT
      WRITE (IF,*)
      DO 40 I=1,NN
 40   WRITE (IF,*) XMIN+(I-1)*XINC,X(I)
      CLOSE (IF)
      RETURN
      END
C#######################################################################
C#######################################################################
      SUBROUTINE CMPSE (CCF,SXY,XVAR,PA,XMEAN)
C-----------------------------------------------------------------------
C     This subroutine estimates the auto correlation functions, and the
C     normalized auto amplitude spectra.
C
C     The correlation method for power spectrum estimation is used.  Thi
C     modification of the program written by:
C     L. Rabiner, Bell Laboratories, Murray Hill, New Jersey 07974
C     R. Schafer,et.al, Georgia Institute of Technology, Atlanta, Georgi
```

```
C
C     This method is based on the technique described by C. M. Rader,
C     in the IEEE Trans. on Audio and Elect., Vol. 18, No. 4, pp 439-442
C-------------------------------------------------------------------
C Input:        M is the section size (must be a power of 2);
C                   2 <= M <= 1024.
C               N is the number of samples to be used in the analysis.
C               MODE is the data format type;
C                   MODE = 0   for autocorrelation and autospectrum;
C               FS is the sampling frequency in Hz (i.e., number of samp
C                   second).
C               IWIN is the window type;
C                   IWIN = 1   for rectangular window;
C                   IWIN = 2   for Hamming window.
C               (2L-1) is the window base width used in the spectral est
C                   2 <= L <= (M/2+1).
C               NFFT is the FFT size used in the spectral estimate;
C                   (2L-1) <= NFFT <= 1024.
C-------------------------------------------------------------------
C Variables returned to calling program:
C               CCF(M/2+1) = Autocorrelation function for +ve lags;
C               SXY(NFFT/2+1) = Autospectrum (take absolute of SXY);
C               XVAR = Variance of the force-series;
C               PA = Absolute peak of the force-series.
C-------------------------------------------------------------------
      REAL XA(1025),CCF(*)
      COMPLEX X(1025),Z(1025),XMN,XI,YI,SXY(*)
      COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
      PARAMETER (PI=3.141592654)
C
C DEFINE CONSTANTS. NSECT IS THE TOTAL NUMBER OF ANALYSIS SECTIONS.
C LSHFT IS THE SHIFT BETWEEN ADJACENT ANALYSIS SAMPLES
C
      LSHFT=M/2
      MHLF1=LSHFT+1
      NSECT=(FLOAT(N)+FLOAT(LSHFT)-1.)/FLOAT(LSHFT)
C
C SS IS THE GENERATOR SAMPLE NUMBER.  AT THE VERY FIRST CALL TO SUBROUTI
C GETX THIS IS SET AT ZERO SO THAT ANY NECESSARY INITIALIZATION
C CAN BE PERFORMED IN THESE SUBROUTINES.
C NRD IS NUMBER OF SAMPLES OF GENERATOR OUTPUT TO BE COMPUTED OR READ.
C
      SS=0.
      NRD=LSHFT
      XSUM=0.
      YSUM=0.
      XXSUM=0.
      YYSUM=0.
      PA=0.
C
C LOOP TO CALCULATE MEANS AND VARIANCES OF X AND Y DATA
C USE GETX TO READ NRD SAMPLES FROM X GENERATOR STARTING AT SAMPLE SS
C
      DO 30 K=1,NSECT
      IF (K.EQ.NSECT) NRD=N-(K-1)*NRD
      CALL GETX (XA,NRD,SS,N)
      DO 10 I=1,NRD
      XSUM=XSUM+XA(I)
 10   XXSUM=XXSUM+XA(I)*XA(I)
C
C COMPUTE THE ABSOLUTE MAXIMUM OF THE NRD VALUES OF XA.
C
      DO 20 I=1,NRD
 20   IF (ABS(XA(I)).GT.ABS(PA)) PA=XA(I)
      IF (K.EQ.1) SS=1.
      SS=SS+FLOAT(NRD)
 30   CONTINUE
      FN=FLOAT(N)
      XMEAN=XSUM/FN
      XVAR=XXSUM/FN-XMEAN*XMEAN
      SQVAR=SQRT(XVAR*XVAR)
      XMN=CMPLX(XMEAN,XMEAN)
C
C     REMOVE THE MEAN FROM THE PEAK (MEAN WILL BE A DELTA FUNCTION AT ZE
C
      IF (PA.GT.0) THEN
      PA=PA-XMEAN
```

```
      ELSE
      PA=PA+XMEAN
      END IF
C
C LOOP TO ACCUMULATE CORRELATIONS
C
      SS=1.
      NRDY=M
      NRDX=LSHFT
      DO 40 I=1,MHLF1
 40   Z(I)=(0.,0.)
      DO 110 K=1,NSECT
      NSECT1=NSECT-1
      IF (K.LT.NSECT1) GO TO 60
      NRDY=N-(K-1)*LSHFT
      IF (K.EQ.NSECT) NRDX=NRDY
      IF (NRDY.EQ.M) GO TO 60
      NRDY1=NRDY+1
      DO 50 I=NRDY1,M
 50   X(I)=(0.,0.)
C
C READ NRDY SAMPLES FROM X GENERATOR STARTING AT SAMPLE SS
C
 60   CALL GETX (XA,NRDY,SS,N)
      DO 70 I=1,NRDY
 70   X(I)=CMPLX(XA(I),XA(I))
      DO 80 I=1,NRDY
 80   X(I)=X(I)-XMN
      NRDX1=NRDX+1
      DO 90 I=NRDX1,M
 90   X(I)=CMPLX(0.,AIMAG(X(I)))
C
C CORRELATE X AND Y SECTIONS
C DO EVEN-ODD SEPARATION AND ACCUMULATE  CONJG(X)*Y
C
      CALL FFT (X,M,0)
      DO 100 I=2,LSHFT
      J=M+2-I
      XI=(X(I)+CONJG(X(J)))*.5
      YI=(X(J)-CONJG(X(I)))*.5
      YI=CMPLX(AIMAG(YI),REAL(YI))
      Z(I)=Z(I)+CONJG(XI)*YI
 100  CONTINUE
      XI=X(1)
      Z(1)=Z(1)+CMPLX(REAL(XI)*AIMAG(XI),0.)
      XI=X(MHLF1)
      Z(MHLF1)=Z(MHLF1)+CMPLX(REAL(XI)*AIMAG(XI),0.)
      SS=SS+FLOAT(LSHFT)
 110  CONTINUE
C
C INVERSE DFT TO GIVE CORRELATION
C
      DO 120 I=2,LSHFT
      J=M+2-I
      X(I)=Z(I)
      X(J)=CONJG(Z(I))
 120  CONTINUE
      X(1)=Z(1)
      X(MHLF1)=Z(MHLF1)
      CALL FFT (X,M,1)
C
C STORE NORMALIZED CORRELATIONS FOR POSITIVE LAGS.
C
      DO 130 I=1,MHLF1
      XA(I)=REAL(X(I))/FN
      CCF(I)=XA(I)*M/SQVAR
      XA(I)=CCF(I)
 130  CONTINUE
C
C WINDOW CORRELATION USING (2L-1) POINT WINDOW. FOR AUTOCORRELATION USE
C OF SYMMETRY.
C In section below, K = -1 for negative lags and K = 1 for positive lags
C
      K=-1
      J2=1
      L2=L
      NLAST=NFFT-L2+1
```

```
C
      IF (IWIN.EQ.2) THEN
      DO 140 I=J2,L2
 140  XA(I)=XA(I)*(0.54+0.46*COS(PI*FLOAT(K*(I-1))/FLOAT(L-1)))
      END IF
C
C Put negative lag values at right end of XA(I)
C
      IF (K.EQ.-1) THEN
      DO 150 I=2,L2
      J=NFFT+2-I
 150  XA(J)=XA(I)
      END IF
C
C
      DO 160 I=L2+1,NLAST
 160  XA(I)=0.
      DO 170 I=1,NFFT
 170  X(I)=CMPLX(XA(I),0.)
C
C COMPUTE NORMALIZED ONE-SIDED AUTOSPECTRUM OR CROSS SPECTRUM
C
      CALL FFT (X,NFFT,0)
      DO 180 I=1,NFFT/2+1
 180  SXY(I)=2.*X(I)/DF
      RETURN
      END
C#######################################################################
C#######################################################################
      SUBROUTINE AUTOSCL (X,N,TICK,ST,FIN)
C-----------------------------------------------------------------------
C This routine determines the starting and ending values to be used in P
C     for a full scale plot.
C INPUT : X = X or Y array;  N = Number of elements in X;
C         TICK = Distance between tick marks; a) >0 to use supplied valu
C                b) =0 to choose automatically
C OUTPUT : ST = Starting value;  FIN = Ending value;  TICK (If originall
C-----------------------------------------------------------------------
      REAL X(*)
      REAL*8 U,C
C
      FIN=X(1)
      ST=X(1)
      DO 10 I=1,N
      IF (X(I).GT.FIN) FIN=X(I)
 10   IF (X(I).LT.ST) ST=X(I)
C
      IF (TICK.GT.0.) THEN
      ST=TICK*IRINT(ST/TICK)
      FIN=TICK*(IRINT(FIN/TICK)+1)
      ELSE IF (TICK.EQ.0) THEN
      TT=ALOG10(FIN-ST)
      MT=IRINT(TT)
      TT=10.**(TT-MT)
      IF (TT.LE.2) THEN
      TT=2.
      ELSE IF (TT.GT.5) THEN
      TT=10.
      ELSE
      TT=5.
      END IF
      TICK=TT*10.**(MT-1)
      U=1.00001*TICK
      C=DLOG10(U)
      IF (C.LT.0) THEN
      L=IDINT(C)-1
      ELSE
      L=IDINT(C)
      END IF
      C=DBLE(10.)**L
      TICK=C*IDNINT(U/C)
      N1=IRINT(ST/TICK)
      N2=IRINT(FIN/TICK)+1
      ST=N1*TICK
      FIN=N2*TICK
      NT=N2-N1
      END IF
```

```
      RETURN
      END
      INTEGER FUNCTION IRINT(X)
      IF (X.LT.0) THEN
      IRINT=INT(X)-1
      ELSE
      IRINT=INT(X)
      END IF
      RETURN
      END
C######################################################################
C######################################################################
      SUBROUTINE FFT (X,N,INV)
C----------------------------------------------------------------------
C This subroutine computes the discrete Fourier Transform or the inverse
C     transform of X.
C----------------------------------------------------------------------
C Input:    X = 2**M complex array that initially contains the input
C               and on return contains the transform;
C           N = 2**M points. (Must be power of 2, else infinite loop resu
C           INV = 0 for direct transform; 1 for inverse transform.
C----------------------------------------------------------------------
C
      COMPLEX X(*),U,W,T
      M=ALOG(FLOAT(N))/ALOG(2.)+.1
      NV2=N/2
      NM1=N-1
      J=1
      DO 40 I=1,NM1
      IF (I.GE.J) GO TO 10
      T=X(J)
      X(J)=X(I)
      X(I)=T
 10   K=NV2
 20   IF (K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
 30   J=J+K
 40   CONTINUE
      PI=4.0*ATAN(1.0)
      DO 70 L=1,M
      LE=2**L
      LE1=LE/2
      U=(1.0,0.0)
      W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      IF (INV.NE.0) W=CONJG(W)
      DO 60 J=1,LE1
      DO 50 I=J,N,LE
      IP=I+LE1
      T=X(IP)*U
      X(IP)=X(I)-T
      X(I)=X(I)+T
 50   CONTINUE
      U=U*W
 60   CONTINUE
 70   CONTINUE
      IF (INV.EQ.1) RETURN
      DO 80 I=1,N
      X(I)=X(I)/N
 80   CONTINUE
      RETURN
      END
C######################################################################
C######################################################################
      SUBROUTINE GETX (X,NRD,SS,N)
C----------------------------------------------------------------------
C This subroutine passes the required length of a TIME SERIES to the cal
C     program.  On the first call to this subroutine SS = 0, and the ent
C     values of the force-series is read from logical unit 11, and the f
C     values are placed in array X.  On all subsequent calls NRD values
C     force-series starting from sample SS are placed in array X.
C----------------------------------------------------------------------
C
      REAL X(*),XSTORE(7500)
C
 10   FORMAT (45X,F13.6)
```

```
C
      J=SS
      IF (J.EQ.0) THEN
      READ (6,10) (XSTORE(I),I=1,N)
      J=1
      END IF
      DO 20 I=1,NRD
 20   X(I)=XSTORE(J+I-1)
      RETURN
      END
C######################################################################
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                  *
* *                       SUBROUTINE PLTS                            *
* *                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                  *
* *   AUG   22,1992        TO PLOT CORRELATION AND SPECTRUM          *
* *                                                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * **************************************************************** *
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
C
      SUBROUTINE PLTS (XMIN,XMAX,YMIN,YMAX,XTICSPACE,YTICSPACE,
     1XAXISTITLE,YAXISTITLE,TITLE,AXIS,X,Y,NUMBER,NN,BW)
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,y3plot,
     2PLOT2METHOD
      INTEGER*2 BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     1TITLECOLOR,NUMBER,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,PLOT3COLOR
      DIMENSION X(NUMBER), Y(NUMBER), AXIS(*)
      CHARACTER TITLE*280,XAXISTITLE*80,YAXISTITLE*80
C
C
C       INITIALISE THINGS FOR THE FIRST PLOT
C
      INITGRAPHICS=.TRUE.
      INITBORDER=.TRUE.
      INITWINDOW=.TRUE.
      INITAXES=.TRUE.
      INITFONTS=.TRUE.
      PLOTMETHOD=.TRUE.
      TITLEONLY=.FALSE.
C
C     y2plot should be set to "true" to show the zero axis
C
      Y2PLOT=.TRUE.
      Y3PLOT=.FALSE.
      FONTHEIGHT=10
      FONTWIDTH=5
C       SET UP THE FANCY COLORS
      BORDERCOLOR=14
      WINDOWCOLOR=1
      AXISCOLOR=10
      LABELCOLOR=11
      PLOTCOLOR=14
      PLOT2COLOR=4
      PLOT3COLOR=4
      TITLECOLOR=15
      IF(BW.EQ.1.0) THEN
         BORDERCOLOR=3
         WINDOWCOLOR=15
         AXISCOLOR=0
         LABELCOLOR=4
         PLOTCOLOR=0
         PLOT2COLOR=1
         PLOT3COLOR=3
         TITLECOLOR=0
```

```
      END IF
      IF (Nn.EQ.1) THEN
C
C                       ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. Auto Correlation        ++++++++++++++++
C                       ++++++++++++++++++++++++++++++++++
C
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.10
      XTOPRIGHT=0.50
      YTOPRIGHT=0.90
C
C
      ELSE IF (Nn.EQ.2) THEN
C
C                       ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      2. Auto Spectra          ++++++++++++++++
C                       ++++++++++++++++++++++++++++++++++
C
      INITGRAPHICS=.FALSE.
      INITFONTS=.FALSE.
      INITAXES=.TRUE.
      WINDOWCOLOR=1
      IF(BW.EQ.1.0)WINDOWCOLOR=15
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.10
      XTOPRIGHT=1.00
      YTOPRIGHT=0.90
      ELSE IF (Nn.EQ.3) THEN
C
C                       ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      3. Average Correlation     ++++++++++++++++
C                       ++++++++++++++++++++++++++++++++++
C
      WINDOWCOLOR=0
      IF(BW.EQ.1.0)WINDOWCOLOR=15
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.10
      XTOPRIGHT=0.50
      YTOPRIGHT=0.90
      ELSE IF (Nn.EQ.4) THEN
C
C                       ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      4. Average Spectra       ++++++++++++++++
C                       ++++++++++++++++++++++++++++++++++
C
      WINDOWCOLOR=0
      IF(BW.EQ.1.0)WINDOWCOLOR=15
      INITGRAPHICS=.FALSE.
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.10
      XTOPRIGHT=1.00
      YTOPRIGHT=0.90
      ELSE IF ((Nn.EQ.5).OR.(Nn.EQ.6)) THEN
C
C                       ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++   5. Top Title on the Screen     ++++++++++++++++
C                       ++++++++++++++++++++++++++++++++++
C
      INITGRAPHICS=.FALSE.
      TITLEONLY=.TRUE.
      INITAXES=.FALSE.
      INITFONTS=.TRUE.
      FONTHEIGHT=28
      FONTWIDTH=16
      WINDOWCOLOR=7
      TITLECOLOR=4
      IF (BW.EQ.1.0) THEN
      WINDOWCOLOR=15
      TITLECOLOR=4
      END IF
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.90
      XTOPRIGHT=1.00
      YTOPRIGHT=1.00
      ELSE IF ((Nn.EQ.7).OR.(Nn.EQ.8)) THEN
C
```

```
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      6. Bottom Title Screen    ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
       INITGRAPHICS=.FALSE.
       TITLEONLY=.TRUE.
       INITAXES=.FALSE.
       INITFONTS=.TRUE.
       FONTHEIGHT=18
       FONTWIDTH=9
       WINDOWCOLOR=7
       TITLECOLOR=0
       IF (BW.EQ.1.0) THEN
       WINDOWCOLOR=15
       TITLECOLOR=4
       END IF
       XBOTTOMLEFT=0.00
       YBOTTOMLEFT=0.00
       XTOPRIGHT=1.00
       YTOPRIGHT=0.10
       END IF
C
C     Call plot to print graphs to the screen
C
       CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
      1X,Y,AXIS,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
      2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
      3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
      4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
      5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3,BW
      6,PLOT2METHOD)
C
C
C        WAIT FOR USER TO HIT RETURN ( only for testing the program
C        , otherwise let the program run with no stops)
C
       IF ((Nn.EQ.7).OR.(Nn.EQ.8)) THEN
       READ (*,*)
       END IF
C      IF (Nn.EQ.8) CALL ENDGRAPHICS ()
       RETURN
       END
```

MYSTAT.FOR

```
C#######################################################################
C#######################################################################
      SUBROUTINE MYSTAT(X,Nn,XMIN,XMAX,NMIN,NMAX,XMEAN,XVAR,XSTD,XVX,
     1XSKE,XKUR,XSE,XABS)
      DIMENSION X(Nn)
      LOGICAL*1 XABS
      XMAX=-1E14
      XMIN=1e14
      NMAX=0
      NMIN=0
      XSUM=0.0
      XSUM2=0.0
      XSUM3=0.0
      XSUM4=0.0
      DO 10 I=1,Nn
      if(xabs) then
      DO 20 II=1,Nn
      X(I)=ABS(X(I))
20    CONTINUE
      END IF
C
C     DETERMINE THE MEAN
C
      XSUM=XSUM+X(I)
      XSUM2=XSUM2+X(I)**2.0
C
C     FINDING MAXIMUM AND MINIMUM
C
      IF (X(I).GT.XMAX) THEN
      XMAX=X(I)
      NMAX=I
      END IF
      IF (X(I).LT.XMIN) THEN
      XMIN=X(I)
      NMIN=I
      END IF
 10   CONTINUE
      XN=FLOAT(Nn)
      XMEAN =XSUM/XN
      XVAR =XSUM2/XN-XMEAN**2.0
      XSTD=SQRT(XVAR)
      XVX=(XSTD/XMEAN)*100.0
      XSE=XSTD/SQRT(XN)
C
C     COEFFICIENT OF SKEWNESS AND COEFFICIENT OF KURTOSIS
C
      DO 30 I=1,Nn
      XSUM3=XSUM3+(X(I)-XMEAN)**3.0
      XSUM4=XSUM4+(X(I)-XMEAN)**4.0
 30   CONTINUE
      XSKE=XSUM3/XSTD**3.0
      XKUR=XSUM4/XSTD**4.0
      RETURN
      END
```

PLOT.FOR

```
$STORAGE:2
$LARGE
C#####################################################################
C        START OF MODULE PLOT.FOR
      INCLUDE 'FGRAPH.FI'
      SUBROUTINE PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,
     1INITFONTS,XDATA,YDATA,Y2DATA,NUMBER,PLOTMETHOD,INVERTX,INVERTY,
     2XMIN,XMAX,YMIN,YMAX,XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,
     3XTOPRIGHT,YTOPRIGHT,PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,
     4LABELCOLOR,PLOTCOLOR,TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,
     5TITLEONLY,FONTHEIGHT,FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,
     6X3,Y3,NUM3,BW,PLOT2METHOD)
C        PURPOSE         TO PLOT A SET OF X-Y DATA USING THE MICROSOFT
C                        FORTRAN 5.0 GRAPHICS LIBRARY.  PLOT WILL PLOT
C                        AN 'UNLIMITED' SET OF X AND Y DATA USING
C                        CGA, EGA, VGA OR HERCULES GRAPHICS WITH
C                        USER DEFINED PLOT POSITIONING, AXIS LABELLING
C                        AND TITLING.
C        PROGRAMMER      LIAM E. GUMLEY, CURTIN UNIVERSITY OF TECHNOLOGY
C        REVISION        20 APRIL 1992 BY BASSEM K. KHAFAGI
C        NOTES           HERCULES GRAPHICS USERS MUST RUN
C                        MSHERC.COM (MICROSOFT HERCULES RESIDENT VIDEO
C                        SUPPORT ROUTINES, SUPPLIED WITH MICROSOFT
C                        FORTRAN 5.0) BEFORE ATTEMPTING TO CALL PLOT.
C                        PLOT ALSO USES THE MICROSOFT FORTRAN 5.0 FONT
C                        FILE HELVB.FON TO GENERATE CHARACTERS.
C                        HELVB.FON MUST BE IN THE CURRENT DIRECTORY WHEN
C                        PLOT IS CALLED OTHERWISE THE PROGRAM WILL
C                        STOP WITH A 'FONT FILE NOT FOUND' MESSAGE.
C                        NOTE THAT THE FONT FILE HELVB.FON ONLY
C                        NEEDS TO BE INITIALISED ONCE, ON THE FIRST
C                        CALL TO PLOT.  THE FONTS REMAIN RESIDENT IN
C                        MEMORY FOR SUBSEQUENT CALLS TO PLOT.
C                        ALSO NOTE THAT SELECTION OF COLORS MAY NEED TO
C                        BE CHANGED ON CGA OR MONOCHROME MONITORS, AS
C                        THESE ONLY SUPPORT TWO COLORS (BLACK OR WHITE).
C                        THE X AND Y AXIS LABELS ARE WRITTEN IN
C                        SCIENTIFIC NOTATION.  THE BASE IS WRITTEN NEXT
C                        TO THE TICS, AND THE EXPONENT PORTION, TAKEN
C                        FROM THE AXIS MAXIMUM VALUE, IS WRITTEN AT THE
C                        END OF THE AXIS.
C        CALLS
C        GRAPHICSMODE()  SET HIGHEST RESOLUTION ON CGA/EGA/VGA/HGC
C                        ALSO CALLS ROUTINES FROM MICROSOFT FORTRAN 5.0
C                        GRAPHICS LIBRARY GRAPHICS.LIB.  MAIN PROGRAMS
C                        WHICH CALL THIS SUBROUTINE (PLOT) MUST BE LINKED
C                        WITH GRAPHICS.LIB.
C                        E.G.
C                        >FL MAIN.FOR PLOT.FOR GRAPHICS.LIB
C        ON ENTRY
C        LOGICAL*1       INIT GRAPHICS    = TRUE : ENTER GRAPHICS MODE
C                                         = FALSE: NO ACTION
C        LOGICAL*1       INIT BORDER      = TRUE : DRAW WINDOW BORDER
C                                         = FALSE: NO ACTION
C        LOGICAL*1       INIT WINDOW      = TRUE : ERASE INSIDE WINDOW
C                                         = FALSE: NO ACTION
C        LOGICAL*1       INIT AXES        = TRUE : DRAW AXES AND LABELS
C                                         = FALSE: NO ACTION
C        LOGICAL*1       INIT FONTS       = TRUE : INITIALISE FONT FILE
C                                         = FALSE: NO ACTION
C        REAL*4          XDATA            ARRAY OF ORDINATES (ASCENDING)
C        REAL*4          YDATA            ARRAY OF COORDINATES
C        INTEGER*2       NUMBER           # OF ELEMENTS IN XDATA, YDATA
C        LOGICAL*1       PLOT METHOD      = TRUE : JOIN POINTS WITH LINES
C                                         = FALSE: DRAW DOTS AT POINTS
C        LOGICAL*1       INVERT X         = TRUE : X AXIS INVERTED
C                                         = FALSE: NO ACTION
C        LOGICAL*1       INVERT Y         = TRUE : Y AXIS INVERTED
C                                         = FALSE: NO ACTION
C        REAL*4          XMIN             MINIMUM TO PLOT FOR X AXIS
C        REAL*4          XMAX             MAXIMUM TO PLOT FOR X AXIS
C        REAL*4          YMIN             MINIMUM TO PLOT FOR Y AXIS
C        REAL*4          YMAX             MAXIMUM TO PLOT FOR Y AXIS
C        REAL*4          X TIC SPACE      TIC-LABEL INTERVAL FOR X AXIS
C        REAL*4          Y TIC SPACE      TIC-LABEL INTERVAL FOR Y AXIS
```

```
C         REAL*4            X BOTTOM LEFT    SCREEN PLOT LOCATION (0.0-1.0)
C                                            (0.0,0.0) = SCREEN BOTTOM LEFT
C                                            (1.0,1.0) = SCREEN TOP RIGHT
C         REAL*4            Y BOTTOM LEFT    SCREEN PLOT LOCATION (0.0-1.0)
C         REAL*4            X TOP RIGHT      SCREEN PLOT LOCATION (0.0-1.0)
C         REAL*4            Y TOP RIGHT      SCREEN PLOT LOCATION (0.0-1.0)
C         INTEGER*2         BORDER COLOR     COLOR FOR WINDOW BORDER
C         INTEGER*2         WINDOW COLOR     COLOR FOR WINDOW BACKGROUND
C         INTEGER*2         AXIS COLOR       COLOR FOR PLOT AXES AND TICS
C         INTEGER*2         LABEL COLOR      COLOR FOR AXIS LABELS
C         INTEGER*2         PLOT COLOR       COLOR FOR PLOTTED DATA
C         INTEGER*2         PLOT2 COLOR      COLOR FOR PLOTTED DATA (2ND SET)
C         INTEGER*2         TITLE COLOR      COLOR FOR TITLE TEXT
C         CHARACTER*80      X AXIS TITLE     TITLE TEXT FOR X AXIS
C         CHARACTER*80      Y AXIS TITLE     TITLE TEXT FOR Y AXIS
C         CHARACTER*80      TITLE            TITLE TEXT FOR PLOT
C         LOGICAL*1         TITLE ONLY       = TRUE : BORDER AND TITLE ONLY
C                                            = FALSE: NO ACTION
C         INTEGER*2         FONT HEIGHT      CHARACTER FONT HEIGHT (PIXELS)
C         INTEGER*2         FONT WIDTH       CHARACTER FONT WIDTH  (PIXELS)
C         ON EXIT           NOTHING CHANGED, NOTHING RETURNED
C-----------------------------------------------------------------------
      INCLUDE 'FGRAPH.FD'
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,Y3PLOT,
     2PLOT2METHOD
      INTEGER*2 DUMMY,XWIDTH,YHEIGHT,IX1,IY1,IX2,IY2,BORDERCOLOR,
     1WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,TITLECOLOR,NUMBER,I,IX,
     2IY,TICLENGTH,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,TITLELENGTH,
     3PLOT3COLOR
      REAL*4 X,Y,XMIN,YMIN,XMAX,YMAX,XDATA(NUMBER),YDATA(NUMBER),
     1Y2DATA(NUMBER),XW1,YW1,XW2,YW2,XTICSPACE,YTICSPACE,XBOTTOMLEFT,
     2YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,XT,YT,X3(NUM3),Y3(NUM3)
      CHARACTER TICLABEL*8,FONTFILE*12,FONTSTRING*7,TITLE*80,
     1XAXISTITLE*80,YAXISTITLE*80,FONTCOMMAND*10,TITLE1*80,TITLE2*80,
     2TITLE3*80,TITLE4*80,TITLE5*80,TITLE6*80,TITLE7*80,TITLE8*80
        RECORD          / VIDEOCONFIG / SCREEN
        RECORD          / WXYCOORD /    WXY
        RECORD          / XYCOORD /     POSITION
      COMMON SCREEN
C       SET THE NAME OF THE FONT FILE TO USE (MUST BE IN CURRENT DIR)
      FONTFILE='HELVB.FON'
      FONTCOMMAND="T'HELV'"
C       SET THE GRAPHICS MODE TO HIGHEST POSSIBLE RESOLUTION
      IF (INITGRAPHICS) CALL GRAPHICSMODE ()
C       REGISTER FONT FOR LABELS
      IF (INITFONTS) THEN
      CALL UNREGISTERFONTS ()
      IF (REGISTERFONTS(FONTFILE).LT.0) THEN
      WRITE (*,10) FONTFILE
 10   FORMAT (' FONT FILE ',A12,' NOT FOUND IN PLOT')
      STOP
      END IF
C         SET UP FONT FOR USE
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
 20   FORMAT ('H',I2.2,'W',I2.2,'B')
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
      END IF
C       GET SCREEN DIMENSIONS
      XWIDTH=SCREEN.NUMXPIXELS
      YHEIGHT=SCREEN.NUMYPIXELS
C       SET A VIEW PORT, WITH CORNER COORDINATES DEFINED AS A
C       FRACTION OF THE TOTAL SCREEN SIZE
C       (0.0,0.0) IS BOTTOM LEFT OF SCREEN
C       (1.0,1.0) IS TOP RIGHT OF SCREEN
C       TRANSLATE CORNER COORDINATES TO PIXEL COORDINATES
      IX1=INT(XBOTTOMLEFT*FLOAT(XWIDTH))
      IX2=INT(XTOPRIGHT*FLOAT(XWIDTH))
      IY1=YHEIGHT-INT(YTOPRIGHT*FLOAT(YHEIGHT))
      IY2=YHEIGHT-INT(YBOTTOMLEFT*FLOAT(YHEIGHT))
      CALL SETVIEWPORT (IX1,IY1,IX2,IY2)
C       CREATE A REAL COORDINATE WINDOW, WITH GRAPH AREA SIZED AT
C       60% OF THE TOTAL VIEWPORT SIZE (LEAVES 15% EITHER SIDE FOR
C       LABELS AND TICS)
      SCALE=0.60
C       IF THE GRAPH OCCUPIES LESS THAN 50% OF THE SCREEN IN
C       EITHER THE X OR Y DIRECTION, THEN THE LABELS WON'T FIT.
```

```
C        SO IF THE GRAPH IS LESS THAN 50% OF SCREEN IN X OR Y,
C        THEN SIZE THE GRAPH AREA AT 30% OF THE VIEWPORT SIZE.
      IF (XTOPRIGHT-XBOTTOMLEFT.LT.0.5.OR.YTOPRIGHT-YBOTTOMLEFT.LT.0.5)
     1SCALE=0.30
      IF (SCRONLY) THEN
      XW1=XMIN
      YW1=YMIN
      XW2=XMAX
      YW2=YMAX
      GO TO 30
      END IF
      XW1=XMIN-0.5*(1.0-SCALE)*ABS(XMAX-XMIN)
      YW1=YMIN-0.5*(1.0-SCALE)*ABS(YMAX-YMIN)
      XW2=XMAX+0.5*(1.0-SCALE)*ABS(XMAX-XMIN)
      YW2=YMAX+0.5*(1.0-SCALE)*ABS(YMAX-YMIN)
 30   DUMMY=SETWINDOW(.TRUE.,XW1,YW1,XW2,YW2)
C        DRAW A BORDER AROUND THE VIEWPORT
      IF (INITBORDER) THEN
      IF (INITWINDOW) THEN
      DUMMY=SETCOLOR(WINDOWCOLOR)
      DUMMY=RECTANGLE_W($GFILLINTERIOR,XW1,YW1,XW2,YW2)
      END IF
      DUMMY=SETCOLOR(BORDERCOLOR)
      DUMMY=RECTANGLE_W($GBORDER,XW1,YW1,XW2,YW2)
      END IF
C        SKIP TO TITLE DRAWING IF REQUIRED
      IF (TITLEONLY) GO TO 90
      IF (SCRONLY) GO TO 40
      IF (.NOT.SCRONLY) GO TO 50
C
C        This is the screen design of the openning logo
C
C
 40   TITLELENGTH=80
      DUMMY=SETCOLOR(8)
      IF (BW.EQ.1.0)DUMMY=SETCOLOR(15)
      YYW1=YW1+0.42
      YYW2=YW2-0.42
      DUMMY=RECTANGLE_W($GFILLINTERIOR,XW1,YYW1,XW2,YYW2)
      DUMMY=SETCOLOR(12)
      IF (BW.EQ.1.0)DUMMY=SETCOLOR(0)
      DUMMY=RECTANGLE_W($GBORDER,XW1,YYW1,XW2,YYW2)
C
C        POSITION THE TITLE CENTERED (MORE OR LESS) ABOVE GRAPH
C
      YT=0.5*(YMAX+YMIN)
      CALL MOVETO_W (1.0*(XMAX+XMIN),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY-FONTHEIGHT/2
C
C        enter the program name (Title: from the main program)
C
      DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
      DUMMY=SETCOLOR(11)
      IF (BW.EQ.1.0)DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY+2,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
C
C        Standard logo
C
      TITLE1='  DEPARTMENT OF CIVIL ENGINEERING'
      TITLE2='         MICHIGAN STATE UNIVERSITY'
      TITLE3=' '
      TITLE4='             HUMAN LOAD RESEARCH'
      TITLE5='       PROGRAMS FOR DISSERTATION'
      TITLE6=' '
      TITLE7='                BASSEM K. KHAFAGI'
      TITLE8='                     OCTOBER 1992'
C
C        Drawing the screen
C
C
```

```
      DUMMY=SETCOLOR(15)
      CALL MOVETO (IX,IY-200,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      CALL MOVETO (IX,IY-160,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
      CALL MOVETO (IX,IY-120,POSITION)
      CALL OUTGTEXT (TITLE3(1:TITLELENGTH))
      CALL MOVETO (IX,IY-80,POSITION)
      CALL OUTGTEXT (TITLE4(1:TITLELENGTH))
      CALL MOVETO (IX,IY+80,POSITION)
      CALL OUTGTEXT (TITLE5(1:TITLELENGTH))
      CALL MOVETO (IX,IY+120,POSITION)
      CALL OUTGTEXT (TITLE6(1:TITLELENGTH))
      CALL MOVETO (IX,IY+160,POSITION)
      CALL OUTGTEXT (TITLE7(1:TITLELENGTH))
      CALL MOVETO (IX,IY+200,POSITION)
      CALL OUTGTEXT (TITLE8(1:TITLELENGTH))
C
C       Adding shadow to the text
C
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY-198,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      DUMMY=SETCOLOR(1)
      CALL MOVETO (IX,IY-158,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY-118,POSITION)
      CALL OUTGTEXT (TITLE3(1:TITLELENGTH))
      CALL MOVETO (IX,IY-79,POSITION)
      CALL OUTGTEXT (TITLE4(1:TITLELENGTH))
      CALL MOVETO (IX,IY+84,POSITION)
      CALL OUTGTEXT (TITLE5(1:TITLELENGTH))
      CALL MOVETO (IX,IY+122,POSITION)
      CALL OUTGTEXT (TITLE6(1:TITLELENGTH))
      DUMMY=SETCOLOR(1)
      CALL MOVETO (IX,IY+162,POSITION)
      CALL OUTGTEXT (TITLE7(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY+202,POSITION)
      CALL OUTGTEXT (TITLE8(1:TITLELENGTH))
C       SKIP THE AXES DRAWING IF NOT NEEDED
 50   IF (.NOT.INITAXES) GO TO 140
C       DRAW AXES
      DUMMY=SETCOLOR(AXISCOLOR)
      CALL MOVETO_W (XMIN,YMAX,WXY)
      DUMMY=LINETO_W(XMIN,YMIN)
      DUMMY=LINETO_W(XMAX,YMIN)
      DUMMY=LINETO_W(XMAX,YMAX)
      DUMMY=LINETO_W(XMIN,YMAX)
C       PUT TIC MARKS AND LABELS ON AXES AT USER DEFINED SPACES
C       TIC MARKS ARE ALWAYS 2 PIXELS LONG
      TICLENGTH=2
C       TIC MARKS AND LABELS FOR X AXIS
      DO 70 X=XMIN,XMAX,XTICSPACE
C               INVERT X AXIS PLOTTING IF REQUIRED
      XT=X
      IF (INVERTX) XT=XMAX-ABS(XT-XMIN)
      CALL MOVETO_W (XT,YMIN,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      DUMMY=SETCOLOR(AXISCOLOR)
      DUMMY=LINETO(IX,IY+TICLENGTH)
C               TIC LABEL IS 5 CHARACTERS LONG
      WRITE (TICLABEL,60) X
C60   FORMAT (1PE8.1)
 60   FORMAT (1F5.1)
 61   FORMAT (1F6.2)
C               MAKE ROOM FOR THE TIC LABEL
      IX=IX-(5*FONTWIDTH)/2
      IY=IY+(FONTHEIGHT/2)
      CALL MOVETO (IX,IY,POSITION)
      DUMMY=SETCOLOR(LABELCOLOR)
      CALL OUTGTEXT (TICLABEL(1:5))
 70   CONTINUE
C       PUT TIC MARK AT XMAX IN CASE IT WAS MISSED
```

```
      XT=XMAX
      IF (INVERTX) XT=XMIN
      CALL MOVETO W (XT,YMIN,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      DUMMY=SETCOLOR(AXISCOLOR)
      DUMMY=LINETO(IX,IY+TICLENGTH)
C       TIC MARKS AND LABELS FOR Y AXIS
      DO 80 Y=YMIN,YMAX,YTICSPACE
C               INVERT Y AXIS PLOTTING IF REQUIRED
      YT=Y
      IF (INVERTY) YT=YMAX-ABS(YT-YMIN)
      CALL MOVETO W (XMIN,YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      DUMMY=SETCOLOR(AXISCOLOR)
      DUMMY=LINETO(IX-TICLENGTH,IY)
C               TIC LABEL IS 5 CHARACTERS LONG
      WRITE (TICLABEL,61) Y
C               MAKE ROOM FOR THE TIC LABEL
      IX=IX-(6*FONTWIDTH)
      IY=IY-(FONTHEIGHT/2)
      CALL MOVETO (IX,IY,POSITION)
      DUMMY=SETCOLOR(LABELCOLOR)
      CALL OUTGTEXT (TICLABEL(1:6))
 80   CONTINUE
C       PUT TIC MARK AT YMAX IN CASE IT WAS MISSED
      YT=YMAX
      IF (INVERTY) YT=YMIN
      CALL MOVETO W (XMIN,YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      DUMMY=SETCOLOR(AXISCOLOR)
      DUMMY=LINETO(IX-TICLENGTH,IY)
C       WRITE THE GRAPH TITLE AT TOP OF GRAPH REGION
 90   DUMMY=SETCOLOR(TITLECOLOR)
C       POSITION THE TITLE CENTERED (MORE OR LESS) ABOVE GRAPH
      TITLELENGTH=80
 100  IF (TITLE(TITLELENGTH:TITLELENGTH).EQ.' ') THEN
      TITLELENGTH=TITLELENGTH-1
      GO TO 100
      END IF
      YT=YMAX
      IF (TITLEONLY) YT=0.5*(YMAX+YMIN)
      CALL MOVETO W (0.5*(XMAX+XMIN),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IF (TITLEONLY) THEN
      IY=IY-FONTHEIGHT/2
      ELSE
      IY=IY-2*FONTHEIGHT
      END IF
      CALL MOVETO (IX,IY,POSITION)
      DUMMY=SETCOLOR(TITLECOLOR)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
C       GO TO RETURN STATEMENT IF ONLY TITLE TO BE DRAWN
      IF (TITLEONLY) GO TO 170
C       WRITE THE X AXIS TITLE
      TITLELENGTH=80
 110  IF (XAXISTITLE(TITLELENGTH:TITLELENGTH).EQ.' ') THEN
      TITLELENGTH=TITLELENGTH-1
      GO TO 110
      END IF
C       POSITION THE TITLE CENTERED (MORE OR LESS) BELOW GRAPH
      CALL MOVETO W (0.5*(XMAX+XMIN),YMIN,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY+(3*FONTHEIGHT)/2
      CALL MOVETO (IX,IY,POSITION)
      DUMMY=SETCOLOR(TITLECOLOR)
```

```
      CALL OUTGTEXT (XAXISTITLE(1:TITLELENGTH))
C       WRITE THE Y AXIS TITLE VERTICALLY, SINCE WE CAN'T ROTATE
      TITLELENGTH=80
 120  IF (YAXISTITLE(TITLELENGTH:TITLELENGTH).EQ.' ') THEN
      TITLELENGTH=TITLELENGTH-1
      GO TO 120
      END IF
C       POSITION THE TITLE CENTERED (MORE OR LESS) TO LEFT OF GRAPH
      CALL MOVETO_W (XMIN,0.5*(YMAX+YMIN),WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-8*FONTWIDTH
      IY=IY-(TITLELENGTH*FONTHEIGHT)/2
      DUMMY=SETCOLOR(TITLECOLOR)
C       NOW WRITE THE CHARACTERS ONE AT A TIME VERTICALLY
      DO 130 I=1,TITLELENGTH
      CALL MOVETO (IX,IY,POSITION)
      CALL OUTGTEXT (YAXISTITLE(I:I))
      IY=IY+FONTHEIGHT
 130  CONTINUE
C       PLOT THE DATA IN XDATA AND YDATA
 140  DUMMY=SETCOLOR(PLOTCOLOR)
C       DEFINE CLIPPING REGION
C       GET TOP LEFT POSITION
      CALL MOVETO_W (XMIN,YMAX,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      CALL GETPHYSCOORD (POSITION.XCOORD,POSITION.YCOORD,POSITION)
      IX1=POSITION.XCOORD
      IY1=POSITION.YCOORD
C       GET BOTTOM RIGHT POSITION
      CALL MOVETO_W (XMAX,YMIN,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      CALL GETPHYSCOORD (POSITION.XCOORD,POSITION.YCOORD,POSITION)
      IX2=POSITION.XCOORD
      IY2=POSITION.YCOORD
C       SET UP NEW VIEWPORT FOR CLIPPING
      CALL SETVIEWPORT (IX1,IY1,IX2,IY2)
      DUMMY=SETWINDOW(.TRUE.,XMIN,YMAX,XMAX,YMIN)
C       PLOT THE FIRST POINT, INVERTING IF NECESSARY
      XT=XDATA(1)
      YT=YDATA(1)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-XW2)
      CALL MOVETO_W (XT,YT,WXY)
      IF (.NOT.PLOTMETHOD) DUMMY=SETPIXEL_W(XT,YT)
C       PLOT THE REST OF THE DATA, INVERTING IF NECESSARY
      DO 150 I=2,NUMBER
      XT=XDATA(I)
      YT=YDATA(I)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-YW1)
      IF (PLOTMETHOD) THEN
      DUMMY=LINETO_W(XT,YT)
      ELSE
      CALL MOVETO_W (XT,YT,WXY)
      DUMMY=SETPIXEL_W(XT,YT)
      END IF
 150  CONTINUE
C
C       PLOT THE SECOND SET OF DATA ON THE SAME X-Y AXIS
C
      IF(.NOT.Y2PLOT) GO TO 170
      DUMMY=SETCOLOR(PLOT2COLOR)
      XT=XDATA(1)
      YT=Y2DATA(1)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-XW2)
      CALL MOVETO_W (XT,YT,WXY)
      IF (.NOT.PLOT2METHOD) DUMMY=SETPIXEL_W(XT,YT)
C       PLOT THE REST OF THE DATA, INVERTING IF NECESSARY
      DO 160 I=2,NUMBER
      XT=XDATA(I)
      YT=Y2DATA(I)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-YW1)
      IF (PLOT2METHOD) THEN
```

```
      DUMMY=LINETO_W(XT,YT)
      ELSE
      CALL MOVETO_W (XT,YT,WXY)
      DUMMY=SETPIXEL_W(XT,YT)
      END IF
 160  CONTINUE
C
C       PLOT THE THIRD SET OF DATA ON THE SAME X-Y AXIS
C
      IF(.NOT.Y3PLOT) GO TO 170
      DUMMY=SETCOLOR(PLOT3COLOR)
      XT=X3(1)
      YT=Y3(1)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-XW2)
      CALL MOVETO_W (XT,YT,WXY)
      IF (.NOT.PLOT2METHOD) DUMMY=SETPIXEL_W(XT,YT)
C       PLOT THE REST OF THE DATA, INVERTING IF NECESSARY
      DO 165 I=2,NUM3
      XT=X3(I)
      YT=Y3(I)
      IF (INVERTX) XT=XW2-ABS(XT-XW1)
      IF (INVERTY) YT=YW2-ABS(YT-YW1)
      IF (PLOT2METHOD) THEN
      DUMMY=LINETO_W(XT,YT)
      ELSE
      CALL MOVETO_W (XT,YT,WXY)
      DUMMY=SETPIXEL_W(XT,YT)
      END IF
 165  CONTINUE
C       ALL DONE
c     CALL UNREGISTERFONTS ()
 170  RETURN
      END
C#####################################################################
      SUBROUTINE GRAPHICSMODE ()
      INCLUDE 'FGRAPH.FD'
      INTEGER*2 DUMMY,MAXX,MAXY
      RECORD /VIDEOCONFIG/ MYSCREEN
      COMMON MAXX,MAXY
C
C     FIND GRAPHICS MODE.
C
      CALL GETVIDEOCONFIG (MYSCREEN)
      SELECT CASE( MYSCREEN.ADAPTER )
         CASE( $CGA )
      DUMMY=SETVIDEOMODE($HRESBW)
         CASE( $OCGA )
      DUMMY=SETVIDEOMODE($ORESCOLOR)
         CASE( $EGA, $OEGA )
      IF (MYSCREEN.MONITOR.EQ.$MONO) THEN
      DUMMY=SETVIDEOMODE($ERESNOCOLOR)
      ELSE
      DUMMY=SETVIDEOMODE($ERESCOLOR)
      END IF
         CASE( $VGA, $OVGA, $MCGA )
      IF (MYSCREEN.MONITOR.EQ.$MONO) THEN
      DUMMY=SETVIDEOMODE($VRES2COLOR)
      ELSE
      DUMMY=SETVIDEOMODE($VRES16COLOR)
      END IF
         CASE( $HGC )
      DUMMY=SETVIDEOMODE($HERCMONO)
         CASE DEFAULT
      DUMMY=0
      END SELECT
      IF (DUMMY.EQ.0) STOP 'ERROR:  CANNOT SET GRAPHICS MODE'
C
C     DETERMINE THE MINIMUM AND MAXIMUM DIMENSIONS.
C
      CALL GETVIDEOCONFIG (MYSCREEN)
      MAXX=MYSCREEN.NUMXPIXELS-1
      MAXY=MYSCREEN.NUMYPIXELS-1
      END
C#####################################################################
      SUBROUTINE ENDGRAPHICS ()
      INCLUDE 'FGRAPH.FD'
```

```
INTEGER*2 DUMMY
DUMMY=SETVIDEOMODE($DEFAULTMODE)
RETURN
END
```

FLOOR.FOR

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                     * *
* *                            FLOOR.FOR                                * *
* *                                                                     * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                     * *
* * THIS PROGRAM READS THE VOLTAGE DATA COLLECTED FROM THE PLATE FORM    * *
* * AND FINDS THE CORRESPONDING FORCES.  THE DATA IS STORED AT A RATE   * *
* * OF 100Hz BY 'TRAP.BAS'.                                             * *
* * THIS PROGRAM USED WITH ANY PERIODIC JUMPING TEST. HOWEVER, THE      * *
* *                                                                     * *
* * DEFAULT IS NOW 2 HZ PERIODIC JUMPING.  IF THE FREQUENCY DIFFERS     * *
* * FROM 2 HZ, YOU MAY NEED TO CHANGE THE COUNTER IN DO LOOP 80.  THE   * *
* * VALUE OF THE COUNTER IN THIS LOOP (30) DETEMINES THE RANGE OF       * *
* * SEARCHING FOR THE FIRST MAJOR PEAK.  SEE THE COMMENT JUST BEFORE    * *
* * LOOP 80.                                                            * *
* *                                                                     * *
* * THE PROGRAM NOW DOES NOT INCLUDE THE MASS OF THE PARTICIPENTS IN    * *
* * THE CALCULATION OF THE MASS, DAMP, AND TOTAL FORCE MATRICES.        * *
* * IF YOU NEED TO CONSIDER WITH THE MASS OF THE FLOOR PLUS PART.,      * *
* * YOU MAY HAVE TO CHANGE THE MATRIX FMASS(I) IN LOOP 250 TO M(I).     * *
* * ALSO, YOU WILL NEED TO CHANGE THE DEFINITON OF MASS MA IN LOOP 360* *
* * TO BE EQUAL TO MASS(I,J) INSTEAD OD TMASS(I,J)                      * *
* *                                                                     * *
* *                                                                     * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                     * *
* *    OCTOBER 4, 1989                                                  * *
* *                                                                     * *
* *                                                                     * *
* *                                         BASSEM K. KHAFAGI           * *
* *                                         YING X. CAI                 * *
* *                                                                     * *
* *                                                                     * *
* *                                                                     * *
* *                                         UNIVERSITY OF IDAHO         * *
* *                                         MOSCOW, ID 83843            * *
* *                                         U.S.A                       * *
* *                                                                     * *
* *                                                                     * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ***************************************************************** * *
C
C
C                    +++++++++++++++++++++++++++++++++
C     +++++++++++++++       1. DATA INPUT/OUTPUT       +++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
C
      PROGRAM FLOOR
      DIMENSION VOLT(1000,9),X(300,9),CALI(9),D(150,9),DEF(128,9)
      DIMENSION A(128,9),B(128,9),VEL(128,9),ACCL(128,9),FIT(128,9)
      DIMENSION PHIT(9),MN(9),MNN(9),MA(9,9),DA(9,9),DAA(9,9),DAM(9,9)
      DIMENSION DAMP(9,9),VNODE(9),ANODE(9),XNODE(9),FA(9),FV(9),FX(9)
      DIMENSION STIF(9,9),FORCES(128,9),PART(40),FMASS(9),TOTFORCE(124)
      DIMENSION TMASS(9,9),BVOLT(1,9)
      REAL*8 K(9,9),M(9),LAMBDA(9),E(9),PHI(9,9),MASS(9,9),NODE(9)
      REAL MN,MNN,MA
      CHARACTER FNAME*12,DATE*12,STIME*12,ETIME*12,FORCE*12,TFORCE*12
      CHARACTER FNAME1*3,FNAME2*1,FNAME3*1,FNAME4*1,FNAME5*6,REPORT*12
      LOGICAL WANTX
C
C     NOW, WE READ THE FILE NAME OF THE DATA SET.  WE WILL USE THIS FILE
C     NAME AS A BASE FOR NAMING THE OUTPUT FILES.  IF THE DATA FILE NAME
C     IS 20PJ2-10.OC9, IT MEANS THAT THIS DATA SET FOR 20 PEOPLE
C     PERFORMING PERIODIC JUMPING (PJ) AND THIS IS GROUP (2) AND TEST NO.
C     (10) THAT WAS DONE IN OCT. 1989 (OC9).  THE OUTPUT FILES WILL BE
C     IN THE FORM OF 20PF2F10.OC9, 20PF2T10.OC9, AND 20PF2R10.OC9.  THE
C     PART (PF2F,PF2T,PF2R) REPRESENT PERIODIC JUMPING (P), FINAL FORCES
C     (F) FOR GROUP (2) THEN THE LAST LETTER IN THIS PART REPRESENT THE
C     FILE CONTENT. (F) REPRESENTS NODAL FORCES FOR THIS TEST, (T) REP-
```

```
C     RESENTS TOTAL FORCES FROM THE PERIODIC JUMPING ON THE FLOOR, AND
C     (R) REPRESENTS THE FINAL REPORT FOR THIS TEST.
C
      PRINT 2
      READ(*,3)NODEBASE
      PRINT 5
      READ(*,4)FNAME1,FNAME2,FNAME3,FNAME4,FNAME5
      WRITE(FNAME,6)FNAME1,FNAME2,FNAME3,FNAME4,FNAME5
      WRITE(FORCE,7)FNAME1,FNAME3,FNAME5
      WRITE(TFORCE,8)FNAME1,FNAME3,FNAME5
      WRITE(REPORT,9)FNAME1,FNAME3,FNAME5
 2    FORMAT(1X,' WHICH NODE WILL BE USED AS A BASE FOR CALCULATIONS')
 3    FORMAT(I1)
 4    FORMAT(A3,A1,A1,A1,A6)
 5    FORMAT(1X,' WHAT IS NAME OF INPUT FILE?')
 6    FORMAT(A3,A1,A1,A1,A6)
 7    FORMAT(A3,'F',A1,'F',A6)
 8    FORMAT(A3,'F',A1,'T',A6)
 9    FORMAT(A3,'F',A1,'R',A6)
      OPEN(1,FILE=FNAME)
      OPEN(2,FILE='INPUT')
      OPEN(3,FILE=FORCE)
      OPEN(4,FILE=TFORCE)
      OPEN(5,FILE=REPORT)
      OPEN(6,FILE='PHI')
C
C
C
C                   +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++       2. REDUCING DATA TO 128 SAMPLE      +++++++++++++
C                   +++++++++++++++++++++++++++++++++++++++
C
C
C     READ THE DATE, STARTING TIME, AND ENDING TIME OF THE TEST.  THEN,
C     DELETE THE FIRST FOUR-COLUMNS OF DATA (STRAIN GAGE DATA), AND SAVE
C     THE LVDT VOLTAGE IN THE FILE "LVDT.VLT"
C
          READ(1,12)DATE,STIME,ETIME
   12     FORMAT(A12//,A12/,A12)
C
          DO 10 I=1,1000
                 READ(1,15) Y,Y,Y,Y,(VOLT(I,J),J=1,9)
   10     CONTINUE
   15            FORMAT(13F8.0)
C
C
C         SELECT THE FIRST VOLTAGE READING AS THE BASE TO NORMALIZE
C         THE VOLTAGE READING AGAINST THE PART. WEIGHTS AND SAVE IT IN
C         MATRIX "BVOLT"
C
          DO 18 I=1,9
                BVOLT(1,I)=VOLT(1,I)
   18     CONTINUE
C
C
C
C     SEARCH THE FIRST MAJOR VOLTAGE PEAK VALUE AND SELECT 300
C     SAMPLE POINTS SUCH THAT 149 READINGS BEFORE IT AND 150 AFTER.
C     THE SEARCH WILL USE LVDT NO 5 AS A BASIS SINCE IT IS A MID. NODE
C     IN THE FLOOR SYSTEM.  ALSO IF NVMAX IS >850 WE WILL ASSUME IT 850
C      AND IF IT IS < 150 WE WILL ASSUME IT 150.  IF THE TEST ENDED BEFORE
C     1000 SAMPLES THE SEARCH WILL BE TERMINATED AT "NEND" VARIABLE
C
      J=NODEBASE
      N=0
      NEND=1000
      DO 19 I=1,1000
         N=N+1
         IF((VOLT(I,J).EQ.0.0).AND.(VOLT(I+1,J+1).EQ.0.0)) GO TO 210
   19 CONTINUE
  210 NEND=N
  211 WRITE(*,212) NEND
  212 FORMAT(1X,'THE TEST ENDED AT SAMPLE NO: (',I4,')')
      VMAX=0.0
      N=0
      DO 20 I=1,1000
         N=N+1
```

```
            IF (ABS(VOLT(I,J)).GT.VMAX) THEN
              VMAX=ABS(VOLT(I,J))
              NVMAX=N
              GO TO 20
            ENDIF
   20 CONTINUE
      NV=NVMAX
      MEND=NEND-150
      IF(NVMAX.GE.MEND)THEN
      NVMAX=MEND
      ELSE IF(NVMAX.LE.150)THEN
      NVMAX=150
      END IF
      DO 22 I=1,300
                      DO 25 J=1,9
                      VOLT(I,J)=VOLT(NVMAX+I-150,J)
  25                  CONTINUE
  22  CONTINUE
C
C
C     NORMALIZE DATA BY SUBTRACTING EACH RAW OF DATA FROM THE FIRST
C     VLOTAGE READING WHICH REPRESENT THE STATIC WEIGHT OF THE FLOOR
C     AND PARTICIPANTS.  THEN THE DEFLECTION IS CALCULATED BY DIVIDING
C     EACH COLUMN BY THE CORESPONDING CALIBRATION FACTOR OF THAT LVDT.
C     EACH COLUMN REPRESENT THE DATA READ BY ONE OF THE LVDT'S.  THE
C     CALIBRATION FACTORS ARE STORED IN THE FIRST PART OF FILE "INPUT"
C
C
       DO 30 J=1,9
            READ(2,*) CALI(J)
          DO 40 I=1,300
            X(I,J)=(VOLT(I,J)-BVOLT(1,J))/(-3276.7*CALI(J))
   40     CONTINUE
   30  CONTINUE
C
C     THE DEFLECTION WILL BE REDUCED NOW TO 150-SAMPLE BY DELETING
C     EVERY OTHER SAMPLE, AND SAVE AS MATRIX D(150,9)
C
      DO 60 I=1,300,2
                 N=(I+1)/2
                      DO 70 J=1,9
                            D(N,J)=X(I,J)
   70                 CONTINUE
   60 CONTINUE
C
C     SEARCH THE FIRST MAJOR DEFLECTION PEAK VALUE AND SELECT 128
C     SAMPLE POINTS SUCH THAT 4 READINGS BEFORE IT AND 123 AFTER.
C     THE SEARCH WILL USE LVDT NO 5 AS A BASIS SINCE IT IS A MID. NODE
C     IN THE FLOOR SYSTEM.
C
C      SINCE THE FREQ. OF THE TEST IS 2 HZ, AND THE RATE OF COLLECTING
C      DATA IS NOW REDUCED TO 50 HZ, A FULL JUMP WILL BE REPRESENTED BY
C      25 SAMPLES.  IN ORDER TO BE ABLE TO GET TO THE MAX. PEAK, WE MUST
C      SEARCH FOR IT IN A RANGE GREATER OR EQUAL THAN 25.  SO, WE SET
C      THE LOOP COUNTER TO 30.
C
      J=NODEBASE
      DMAX=0.0
      NMAX=0
      N=0
      DO 80 I=1,30
         N=N+1
            IF (D(I,J).GT.DMAX) THEN
              DMAX=D(I,J)
              NMAX=N
              GO TO 80
            ENDIF
   80 CONTINUE
            MAX=NMAX
C
C    NOW THE REDUCED SAMPLE OF DEFLECTIONS WILL BE STORED IN MATRIX
C    DEF(128,9) TO USE IT IN THE PROGRAM.
C    THE REDUCED NO. OF SAMPLES (128) WILL BE DENOTED AS "NS"
C
      NS=128
      DO 90 I=1,NS
                      DO 100 J=1,9
```

```
                    DEF(I,J)=D(NMAX+I-5,J)
  100               CONTINUE
   90 CONTINUE
C
C
C                 +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++      3. FINDING FITTED DEFLECTION      +++++++++++++
C                 +++++++++++++++++++++++++++++++++++++++
C
C
C
C     PERFORM PERIODIC REGRESSION OF SELECTED DEFLECTION SAMPLES
C     BY USING THE FOURIER SERIES TO GET FITTED DEFLECTION.
C
C     CALCULATE FOURIER COEFFICIENTS
C
      PI=3.141593
      DO 110 J=1,9
          SUM1=0.
               DO 120 I=1,NS
                  SUM1=SUM1+DEF(I,J)
  120          CONTINUE
          A(0,J)=SUM1/NS
               DO 130 I=1,20
                    SUM2=0
                    SUM3=0
                        DO 140 IJ=1,NS
                          SUM2=SUM2+DEF(IJ,J)*COS(2*I*PI*IJ/NS)
                          SUM3=SUM3+DEF(IJ,J)*SIN(2*I*PI*IJ/NS)
  140                   CONTINUE
                    A(I,J)=2*SUM2/NS
                    B(I,J)=2*SUM3/NS
  130          CONTINUE
C
C     CALCULATE PREDICTED DEFLECTION HISTORY
C
               DO 150 I=1,NS
                   SUM1=0
                   SUM2=0
                       DO 160 JJ=1,20
                           SUM1=SUM1+A(JJ,J)*COS(2*PI*JJ*I/NS)
                           SUM2=SUM2+B(JJ,J)*SIN(2*PI*JJ*I/NS)
  160                  CONTINUE
                    FIT(I,J)=A(0,J)+SUM1+SUM2
  150          CONTINUE
  110  CONTINUE
C
C
C                 +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++      4. VELOCITY AND ACCELERATION      +++++++++++++
C                 +++++++++++++++++++++++++++++++++++++++
C
C
C     ****************************************************************
C     PROGRAM TO DO CENTRAL DIFFERENTIAL
C     NOTE: H IS THE TIME INTERVAL, H= 1/(FREQUENCE OF SELECTED SAMPLE
C     POINTS).  SINCE THE SAMPLES HAVE BEEN REDUCED TO 50 HZ, H=.02 SEC.
C
      H=0.02
      DO 180 J=1,9
         DO 190 I=3,126
           VEL(I,J)=(-FIT(I+2,J)+8*FIT(I+1,J)-8*FIT(I-1,J)+FIT(I-2,J))
     $                /(12*H)
           ACCL(I,J)=(-FIT(I+2,J)+16*FIT(I+1,J)-30*FIT(I,J)+16*FIT(I-1
     $              ,J)-FIT(I-2,J))/(12*H*H)
  190    CONTINUE
  180 CONTINUE
C
C
C                 +++++++++++++++++++++++++++++++++++++++
C     +++++++++++++      5. FLOOR AND PART. MASS           +++++++++++++
C                 +++++++++++++++++++++++++++++++++++++++
C
C
C
C     HERE, WE CALCULATE THE MASS MATRIX FOR THE FLOOR
C     SYSTEM AND THE PARTICIPANTS USING THE WEIGHT AND LOCATION OF
C     EACH PARTICIPANT AND APPLYING IT TO THE CORRECT NODE.
```

```
C
C     THE ARRAYS INCLUDE:
C         PART(40)   - WEIGHTS AND LOCATIONS OF PARTICIPANTS
C         M(9)       - MASSES OF PARTICIPANTS BROKEN UP INTO 9 NODES
C         NODE(9)    -  PARTICIPANT AND FLOOR MASS FOR 9 NODES
C         MASS(9,9) - MASS MATRIX TO BE SAVED IN FILE "MASS"
C
C
C     READ THE PARTICIPANT WEIGHT-LOCATION FILE, (THIS IS IN THE SECOND
C     PART OF THE FILE "INPUT".
C
      N=0
      DO 220 J=1,40
            READ(2,*) PART(J)
            PART(J)=PART(J)/386.4
         N=N+1
             IF(PART(J).GT.0.0) THEN
               NPART=N
               GO TO 220
             ENDIF
 220  CONTINUE
C
C     CALCULATE TOTAL WEIGHT CORRESPONDING TO EACH NODE
C
      NODE(1)=PART(39)+PART(29)+2./3.*(PART(37)+PART(27)+PART(19))
     *         +4./9.*PART(17)
      NODE(2)=1./3.*(PART(19)+PART(20))+2./3.*(PART(7)+PART(8))
     *         +2./9.*(PART(17)+PART(18))+PART(9)+PART(10)
      NODE(3)=PART(30)+PART(40)+2./3.*(PART(20)+PART(28)+PART(38))
     *         +4./9.*PART(18)
      NODE(4)=PART(35)+PART(25)+2./3.*PART(15)+2./9.*(PART(17)+PART(13))
     *         +1./3.*(PART(37)+PART(27)+PART(33)+PART(23))
      NODE(5)=1./9.*(PART(17)+PART(18)+PART(13)+PART(14))
     *         +1./3.*(PART(7)+PART(8)+PART(3)+PART(4)+PART(15)+PART(16))
     *         +PART(5)+PART(6)
      NODE(6)=PART(26)+PART(36)+2./3.*PART(16)+2./9.*(PART(18)+PART(14))
     *         +1./3.*(PART(28)+PART(38)+PART(24)+PART(34))
      NODE(7)=PART(31)+PART(21)+2./3.*(PART(33)+PART(23)+PART(11))
     *         +4./9.*PART(13)
      NODE(8)=PART(1)+PART(2)+2./3.*(PART(3)+PART(4))
     *         +2./9.*(PART(13)+PART(14))+1./3.*(PART(11)+PART(12))
      NODE(9)=PART(22)+PART(32)+2./3.*(PART(24)+PART(34)+PART(12))
     *         +4./9.*PART(14)
C
C    INPUT THE MASS OF THE FLOOR ALONE IN ARRAY FMASS(9)
C
      DO 225 I=1,9
           FMASS(I)=0.358592
 225  CONTINUE
           FMASS(2)=0.222567
           FMASS(5)=0.222567
           FMASS(8)=0.222567
C
C     PUT FLOOR MASSE IN THE FORM OF A 9x9 FLOOR MASS MATRIX
C
      DO 226 I=1,9
            J=I
                TMASS(I,J)=FMASS(J)
 226  CONTINUE
C
C     ADD MASS OF PARTICIPANTS TO MASS OF FLOOR FOR EACH NODE
C
      DO 227 I=1,9
      M(I)=NODE(I)+FMASS(I)
 227  CONTINUE
C
C
C     PUT NODAL MASSES IN THE FORM OF A 9x9 MASS MATRIX
C
      DO 230 I=1,9
            J=I
                MASS(I,J)=M(J)
 230  CONTINUE
 240  FORMAT(9F10.6)
C
C
C
```

```
C                ++++++++++++++++++++++++++++++++++++++
C     +++++++++++++          6. DAMPING MATRIX            +++++++++++++
C                ++++++++++++++++++++++++++++++++++++++
C
C
C PROGRAM TO SET UP AND SOLVE CHARACTERISTIC VALUE PROBLEMS OF THE TYPE:
C             (LAMBDA)*M*X = K*X
C
C THIS PROGRAM READS THE MATRICES K AND M AND CONVERTS IT TO THE STANDARD
C EIGENVALUE FORM, (LAMBDA)*X = A*K BY THE FOLLOWING TRANSFORMATION:
C             (LAMBDA)*(M**0.5)*X = (M**-0.5)*K*(M**-0.5)*(M**0.5)*X
C
C WE THEN SOLVE FOR THE EIGENVALUES OF THE X' = (M**0.5)*X.  (SOME
C REFERENCES DESCRIBE THIS AS SOLVING FOR THE EIGENVALUES IN THE
C GENERALIZED COORDINATES.  WE FINALLY SOLVE FOR X = (M**-0.5)*X'.
C
C IMPORTANT ASSUMPTIONS:
C    K    = IS SYMMETRIC AND REAL STIFFNESS MATRIX
C    M    = IS DIAGONAL MATRIX OF THE MASS USED IN CALCULATION.
C
C LAMBDA = N-VECTOR OF REAL DOUBLE PRECISION VALUES.  THE CALCULATED
C          EIGENVALUES ARE PUT IN THIS VECTOR.
C
C    E    = N-VECTOR.  WORKING VECTOR.
C
C    X    = N-BY-N MATRIX OF REAL DOUBLE PRECISION VALUES.  THE CALCULATED
C          EIGENVECTORS ARE PLACES COLUMN-BY-COLUMN IN THIS MATRIX.
C
C NDIM    = THE ACTUAL DIMENSION OF THE ARRAYS IN THE MAIN PROGRAM.
C
C    N    = THE ORDER OF MATRICES K AND M.
C
C WANTX   = LOGICAL.  IF WANTX = .FALSE. THEN SYMEIG SKIPS THE CALCULATION
C          OF THE EIGENVECTORS, SAVING SOME TIME.
C
      NDIM = 9
      WANTX = .TRUE.
      N=9
C
C     READ MATRIX K, ROW-BY-ROW.  ALSO, CALCULATE M**-0.5.
C     IF YOU NEED TO WORK WITH THE MASS OF THE FLOOR PLUS PARTICIPANTS,
C     YOU HAVE TO CHANGE THE MATRIX FMASS(I) IN THIS LOOP TO M(I)
C
      DO 250 I = 1, N
         READ(2,*) (K(I,J),J=1,N)
         M(I) = 1.0D0/DSQRT(FMASS(I))
  250 CONTINUE
C
C     CALCULATE (M**-0.5)*K
C
      DO 260 I = 1, N
         DO 260 J = 1, N
            PHI(I,J) = M(I)*K(I,J)
C
C    SAVE THE STIFFNESS MATRIX IN THE ARRAY STIF(9,9)
C
            STIF(I,J)=K(I,J)
  260 CONTINUE
C
C     CALCULATE (M**-0.5)*K*(M**-0.5)
C
      DO 270 I = 1, N
         DO 270 J = 1, N
            K(I,J) = PHI(I,J)*M(J)
  270 CONTINUE
C
C     CALCULATE THE EIGENVALUES (AND EIGENVECTORS IF DESIRED).
C
      CALL SYMEIG (NDIM,N,K,LAMBDA,E,WANTX,PHI)
C
C     SKIP THE EIGENVECTORS IF WE DIDN'T ASK FOR THEM
C
      IF (.NOT.WANTX) GO TO 320
C
C     TRANSFORM THE EIGENVECTORS BACK FROM THE "GENERALIZED COORDINATES"
C
      DO 310 J = 1, N
```

```
         DO 310 I = 1, N
            PHI(I,J) = PHI(I,J)*M(I)
  310 CONTINUE
  320 CONTINUE
C
C     WRITE THE EIGENVECTORS INTO FILE='PHI'. THE EIGENVECTORS PHI(J),
C       J=1,2,...,9, ARE PLACED ROW-BY-ROW IN THIS FILE.
C
      DO 330 J=1,9
         WRITE (6,340)(PHI(I,J),I=1,9)
  330 CONTINUE
  340 FORMAT(9E12.6)
      REWIND 6
C
      DO 350 I=1,9
         DO 360 J=1,9
         MA(I,J)=TMASS(I,J)
  360 CONTINUE
  350 CONTINUE
C
C     CALCULATE MNN(N)=[PHIT](N)*[MA]*[PHI](N),[PHIT](N)  IS THE
C     TRANSPOSITION OF NTH EIGENVECTOR.
C
      DO 370 I=1,9
         READ(6,340)(PHIT(J),J=1,9)
         II=1
         JJ=9
         KK=9
         CALL MATRIX(PHIT,MA,MN,II,JJ,KK)
         II=1
         JJ=9
         KK=1
         CALL MATRIX(MN,PHIT,MNN(I),II,JJ,KK)
  370 CONTINUE
      REWIND 6
C
C     CLEAR OUT THE MATRIX DA BEFORE STARTING
C
      DO 380 I=1,9
         DO 390 J=1,9
            DA(I,J)=0.0
  390    CONTINUE
  380 CONTINUE
C
C     CALCULATE DA = SUM. OF ((2*XIN*OMEGAN/MNNN)*[PHI]*[PHIT])
C                  = SUM. OF ((2*XIN*OMEGAN/MNNN)*DAA)
C
      XIN=0.015
      DO 395 I=1,9
         READ(6,340)(PHIT(J),J=1,9)
         II=9
         JJ=1
         KK=9
         CALL MATRIX(PHIT,PHIT,DAA,II,JJ,KK)
         DO 400 J=1,9
            DO 410 L=1,9
               DA(J,L)=DA(J,L)+(2*XIN*DSQRT(LAMBDA(I))/MNN(I))*DAA(J,L)
  410       CONTINUE
  400    CONTINUE
  395 CONTINUE
      REWIND 6
C
C
C     CALCULATE [DAMP]=[MA]*[DA]*[MA]
C
      II=9
      JJ=9
      KK=9
      CALL MATRIX(MA,DA,DAM,II,JJ,KK)
      CALL MATRIX(DAM,MA,DAMP,II,JJ,KK)
C
C
C
C
C                    ++++++++++++++++++++++++++++++++++++++
C     +++++++++++++            7. TOTAL FORCES             +++++++++++++
C                    ++++++++++++++++++++++++++++++++++++++
C
```

```
C
C
C
C     CALCULATE  [FA]=[M]*[ANODE],[FV]=[DAMP]*[VNODE],[FX]=[K]*[XNODE],AND
C        [FORCE]=[FA]+[FV]+[FX]
C
      DO 420 I=1,124
               DO 430 J=1,9
                      XNODE(J)=DEF(I+2,J)
                      VNODE(J)=VEL(I+2,J)
                      ANODE(J)=ACCL(I+2,J)
 430           CONTINUE
         II=9
         JJ=9
         KK=1
         CALL MATRIX(MA,ANODE,FA,II,JJ,KK)
         CALL MATRIX(DAMP,VNODE,FV,II,JJ,KK)
         CALL MATRIX(STIF,XNODE,FX,II,JJ,KK)
         DO 440 J=1,9
            FORCES(I,J)=FA(J)+FV(J)+FX(J)
 440     CONTINUE
         WRITE(3,470) (FORCES(I,J),J=1,9)
 420  CONTINUE
         DO 450 I=1,124
         DO 460 J=1,9
           TOTFORCE(I)=FORCES(I,J)+TOTFORCE(I)
 460  CONTINUE
         WRITE(4,*)TOTFORCE(I)
 450   CONTINUE
 470  FORMAT(9F11.4)
C
C     TO FIND THE MAX. FORCE ON THE FLOOR
C
      N=0
      FMAX=0.0
      NMAX=0
      DO 480 I=1,124
         N=N+1
             IF (TOTFORCE(I).GT.FMAX) THEN
               FMAX=TOTFORCE(I)
               NMAX=N
               GO TO 480
             ENDIF
 480  CONTINUE
         WRITE(*,490)FNAME,NPART,DATE,STIME,ETIME,NODEBASE,NV,VMAX
         WRITE(5,490)FNAME,NPART,DATE,STIME,ETIME,NODEBASE,NV,VMAX
         WRITE(*,500)NODEBASE,MAX,FMAX,FMAX/NPART,FMAX/NPART/3.5
         WRITE(5,500)NODEBASE,MAX,FMAX,FMAX/NPART,FMAX/NPART/3.5
 490  FORMAT(//,28X,'OUTPUT SUMMARY',//,1X,'INPUT FILE NAME : ',A12,//
     *,1X,'NO. OF PART.     : ',I2,//
     *,1X,'DATE OF TEST     : ',A12,//,1X,'TEST STARTED AT : ',A12,//,1X,
     *'TEST ENDED AT    : ',A12//,1X,'MAX. VOLT. (NODE ',I1,') IS SAMPLE
     *NO. ',I3,2X,'=',1X,F10.2,/)
 500  FORMAT(1X,'MAX. PEAK  (NODE ',I1,') IS SAMPLE NO.',I3,3X,'= ',F9.2
     *,2X,'lb',//,1X,'THE AVERAGE DYNAMIC HUMAN LOAD IS       =',F9.2,3X,
     *'lb/PERSON',//,1X,'THE DESIGN EQUIVELANT LOAD IS          =',1X,
     *F7.2,4X,'PSF',/)
      STOP
      END
C
C***********************************************************************
C
      SUBROUTINE SYMEIG (NDIM,N,A,D,E,WANTX,X)
      INTEGER NDIM,N
      LOGICAL WANTX
      REAL*8 A(NDIM,NDIM),D(NDIM),E(NDIM),X(NDIM,NDIM)
      REAL*8 ALPHA,BETA,GAMMA,KAPPA,AIJ,T,C,S,F,DABS,DSQRT
C
C     COMPUTES EIGENVALUES AND EIGENVECTORS OF REAL SYMMETRIC MATRICES
C     FROM "LINEAR MATHEMATICS FOR ENGINEERING APPLICATION" BY JOEL
C     H. FERZIGER  (1978).
C
C     NDIM = DECLARED ROW DIMENSION OF A (AND X)
C     N = ORDER OF A
C
C     A = N-BY-N MATRIX (DOUBLE PRECISION)
C         A IS ASSUMED TO BE SYMMETRIC.  ONLY THE DIAGONAL AND LOWER
```

```
C          TRIANGULAR PORTIONS OF A ARE USED.  IF POSSIBLE, PUT THE
C          BIGGEST ELEMENTS IN THE UPPER LEFT CORNER.  UPPER TRIANGLE
C          PORTION OF A IS UNALTERED UNLESS A AND X ARE IN THE SAME
C          ARRAY.
C
C      D = N-DIMENSIONAL VECTOR, (DOUBLE PRECISION)
C          STORES THE OUTPUT EIGENVALUES.
C
C      E = N-DIMENSIONAL VECTOR, (DOUBLE PRECISION) USED FOR WORK SPACE.
C
C  WANTX = A LOGICAL VARIABLE. IF WANTX = .TRUE. THEN THE EIGENVECTORS ARE
C          CALCULATED.  IF WANTX = .FALSE. THEN THE EIGENVECTORS ARE NOT
C          CALCULATED.
C
C      X = N-BY-N MATRIX (DOUBLE PRECISION)
C          IF WANTX = .TRUE. THEN THE OUTPUT X(I,J) IS THE EIGENVECTOR
C          ASSOCIATED WITH EIGENVALUE D(J).
C
C   CALCULATES THE EIGENVECTORS AND REPLACES THE INPUT MATRIX, A, WITH ITS
C   EIGENVECTORS.
C
C
C      HOUSEHOLDER TRIDIAGONALIZATION
C
       D(1) = A(1,1)
       IF (N.LE.2) GO TO 8
       NM1 = N-1
       DO 7 K = 2, NM1
C
C      FIND REFLECTION WHICH ZEROS A(I,K-1), I=K+1,...,N
C
       ALPHA = 0.
       DO 1 I = K, N
           D(I) = A(I,K-1)
           ALPHA = ALPHA + D(I)*D(I)
 1     CONTINUE
       ALPHA = DSQRT(ALPHA)
       IF (D(K).LT.0.) ALPHA = -ALPHA
       D(K) = D(K) + ALPHA
       BETA = ALPHA*D(K)
       IF(BETA.EQ.0.) GO TO 6
C
C      APPLY REFLECTION TO BOTH ROWS AND COLUMNS OF REST OF A
C      USE AND COMPUTE ONLY LOWER TRIANGLE
C
       KAPPA = 0.
       DO 3 I = K, N
           GAMMA = 0.
           DO 2 J = K, N
               IF (I.GE.J) AIJ = A(I,J)
               IF (I.LT.J) AIJ = A(J,I)
               GAMMA = GAMMA + AIJ*D(J)
 2         CONTINUE
           E(I) = GAMMA/BETA
           KAPPA = KAPPA + D(I)*E(I)
 3     CONTINUE
       KAPPA = 0.5*KAPPA/BETA
       DO 5 I = K, N
           E(I) = E(I) - KAPPA*D(I)
           DO 4 J = K, I
               A(I,J) = A(I,J) - D(I)*E(J) - E(I)*D(J)
 4         CONTINUE
 5     CONTINUE
C
 6     A(K,K-1) = D(K)
       D(K) = A(K,K)
       A(K,K) = BETA
       E(K) = -ALPHA
 7     CONTINUE
 8     D(N) = A(N,N)
       E(N) = A(N,N-1)
C
C      ACCUMULATE TRANSFORMATIONS
C      PRODUCES X SO THAT XT*(INPUT A)*X IS TRIDIAGONAL
C
       IF(.NOT.WANTX) GO TO 20
       X(N,N) = 1.
```

```
      DO 15 KB = 1,NM1
          K = N-KB
          BETA = A(K,K)
          DO 11 I = K, N
              X(I,K) = 0.
              X(K,I) = 0.
 11       CONTINUE
          X(K,K) = 1.
          IF (K.EQ.1 .OR.BETA. EQ.0.) GO TO 15
          DO 14 J = K, N
              GAMMA = 0.
              DO 12 I = K, N
                  GAMMA = GAMMA + X(I,J)*A(I,K-1)
 12           CONTINUE
              GAMMA = GAMMA/BETA
              DO 13 I = K, N
                  X(I,J) = X(I,J) - GAMMA*A(I,K-1)
 13           CONTINUE
 14       CONTINUE
 15   CONTINUE
C
C     TRIDIAGONAL QR ALGORITHM
C     IMPLICIT SHIFT FROM LOWER 2-BY-2
C
 20   DO 27 MB = 2, N
          M = N+2-MB
          MM1 = M-1
          ITER = 0
          L = 1
 21       E(L) = 0.
C
C         FIND L SUCH THAT E(L) IS NEGLIGIBLE
C
          L = M
 22       S = DABS(D(L-1)) + DABS(D(L))
          T = S + DABS(E(L))
          IF (T.EQ.S) GO TO 23
          L = L-1
          IF (L.GE.2) GO TO 22
C
C         IF E(M) IS NEGLIGIBLE, THE D(M) IS AN EIGENVALUE, SO...
C
 23       IF (L.EQ.M) GO TO 27
          IF (ITER.GE.30) GO TO 27
          ITER = ITER + 1
C
C         FORM IMPLICIT SHIFT
C
          T = (D(M-1) - D(M))/(E(M) + E(M))
          S = DSQRT(1. + T*T)
          IF (T.LT.0.) S = -S
          S = D(M) - E(M)/(T+S)
          E(L) = D(L) - S
          F = E(L+1)
C
C         CHASE NONZERO F DOWN THE MATRIX
C
          DO 26 J = L,MM1
              T = DABS(E(J)) + DABS(F)
              ALPHA = T*DSQRT((E(J)/T)**2 + (F/T)**2)
              C = E(J)/ALPHA
              S = F/ALPHA
              BETA = S*(D(J+1) - D(J)) + 2.0*C*E(J+1)
              E(J) = ALPHA
              E(J+1) = E(J+1) - C*BETA
              T = S*BETA
              D(J) = D(J) + T
              D(J+1) = D(J+1) - T
              IF (J.EQ.MM1) GO TO 24
              F = S*E(J+2)
              E(J+2) = -C*E(J+2)
C
 24           IF(.NOT.WANTX) GO TO 26
              DO 25 I = 1, N
                  T = X(I,J)
                  X(I,J) = C*T + S*X(I,J+1)
                  X(I,J+1) = S*T - C*X(I,J+1)
```

```
 25           CONTINUE
C
 26       CONTINUE
          GO TO 21
C
 27   CONTINUE
      RETURN
      END
C
C    ******************************************************************
C
C     SUBROUTINE MATRIX.FOR
C
C    ******************************************************************
C
      SUBROUTINE MATRIX(A,B,C,M,N,L)
      INTEGER M,N,L
      DIMENSION A(M,N),B(N,L),C(M,L)
      DO 10 I=1,M
         DO 20 K=1,L
            C(I,K)=0.0
            DO 30 J=1,N
               C(I,K)=C(I,K)+A(I,J)*B(J,K)
 30         CONTINUE
 20      CONTINUE
 10   CONTINUE
      RETURN
      END
```

APPENDIX B

LOAD SIMULATION PROGRAMS

SJ-HLS.FOR

```
$STORAGE:2
$LARGE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* *                                                                     *
* *                          HLS.FOR                                    *
* *                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* *                                                                     *
* *     OCT 23  , 1992                                                  *
* *                                                                     *
* *                                                                     *
* *                                            BASSEM K. KHAFAGI        *
* *                                                                     *
* *                                                                     *
* *                                                                     *
* *                                            MICHIGAN STATE UNIVERSITY *
* *                                            EAST LANSING, MI 48823   *
* *                                            U.S.A                    *
* *                                                                     *
* *                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* *                                                                     *
* *  THIS PROGRAM READS THE PROCESSED *.CRL FILES FOR A CERTAIN ACTION*
* *  AND PERFORMS THE FOLLOWING TASKS:                                  *
* *                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
* * ******************************************************************* *
C
C                     +++++++++++++++++++++++++++++++++
C      ++++++++++++++++    1. INPUT/OUTPUT MANAGEMENT    +++++++++++++++
C                     +++++++++++++++++++++++++++++++++
C
C
C      PROGRAM hls.FOR
       INTEGER WEIGHT(1000)
C      INTEGER WEIGHT(10)
       CHARACTER INPUTS*40,FOUT*12
       CHARACTER FNAME1*2,FNAME*12,OUTPUTS*40
       COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
       COMMON /FILES/ INPUTS,OUTPUTS,LINP,LOUT
c
c  common statement to save the array u from function uni between calls
c
       common u(17)
c
c  in order to assure that no sequence of random numbers generated by
c  function uni will be used twice, the seed array for uni (array u)
c  is saved from run to run in file 'seedarray'.  in order to get
```

```
c  started, for the first run only, a separate program is run the only
c  purpose of which is to invoke the function rstart that is in uni.
c  rstart initializes the seed array u.  for all subsequent runs the
c  seed array from the previous run will be used instead of invoking
c  rstart by running the separate program.
c
        open(1,file='sj-hls.dat')
       do 4 i=1,17
       read(1,9) u(i)
9      format(1x,e27.20)
4        continue
c
c        DATA INPUT PROVIDED BY THE USER
c
c
1000     format (/1x,a,' ? '$)
       PRINT *,'
      1                    '
       PRINT *,'
      1                    '
C
         write (*,1000) 'No. of Persons Performing this Actions'
5        read (*,*) NPART
         if (NPART.lt. 0.) then
                 write (*,'(/a)')
     *           ' *** ERROR *** NPART is less than 0 - Reenter'
                 go to 5
         else if (NPART.gt.1000.) then
          write (*,'(/a)')
     *    ' **In this BETA version...No. of Part. can not exceed 1000'
                 go to 5
         end if
C
C
16       write (*,1000) 'Do you Want to input Participants Weight..(Y/N)'
         read (*,'(a)') IWT
         IF((IWT.EQ.'Y').or.(IWT.EQ.'y')) THEN
C
C
              do 6 ij=1,npart
239           write (*,1010) Ij
1010          format (/1x,'Weight of Participant No.:'I4,' ? '$)
         read (*,*) weight(ij)
         if (weight(ij).le. 0.0) then
                write (*,'(/a)')
     *          ' *** ERROR *** Weight is less than 0 - Reenter'
              goto 239
         elseif(weight(ij).gt.300.0) then
                write (*,'(/a)')
     * ' Sorry,!!! My limitation is a Weight less than 300 lb. Reenter'
              goto 239
         elseif(weight(ij).lt.30.0) then
                write (*,'(/a)')
     * ' Sorry,!!! Infants were not considered in HLS !!!.. Reenter'
              goto 239
         endif
6        continue
         else IF((IWT.EQ.'N').or.(IWT.EQ.'n')) THEN
C
C
C
         DO 17 In=1,NPART
c
c   Weight r.v.:
c
c   a normally distributed deviation from the nominal value will
c   be used.
c   (mean=+173 lbs, std.dev.=16 lbs) for men
c   (mean=+143 lbs, std.dev.=20 lbs) for women
```

```
c   see REFRENCE: .....
c
        rnum=rnor(0)
      AVGWT=160.0
      STDWT=17.0
18    weight(in)=AVGWT+rnum*STDWT
      if((weight(in).lt.0.0).or.(weight(in).gt.300.0))go to 18
      if(npart.le.15) print *,'Weight of Participant No.:',In,'   is = '
     1,weight(in),' lbs.'
17      continue
        else
        goto 16
        end if
C
C
C
      fname1='SJ'
      WRITE (FNAME,40) FNAME1
      WRITE (FOUT,41) FNAME1
 40   FORMAT (A2,'-HLS.DAT')
 41   FORMAT (A2,'-HLS.OUT')
C
C       FLAG FOR THE OPEN SCREEN
C
      OFLAG=1
C
C Read INPUT Data from PARAMETER FILE "hls.DAT"
C
      READ (1,10) INPUTS
      READ (1,10) OUTPUTS
      READ (1,3) BW
 3    format (f3.1)
      na=1
      n=90
 10   FORMAT (A40)
C
C     DETERMINE THE LOCATION OF INPUT AND OUTPUT FILES
C
C
C       1.0 INPUTS
C
      LINP=LEN_TRIM(INPUTS)
      IF (LINP.NE.0) THEN
      INPUTS=INPUTS(1:LINP)//'\'
      LINP=LINP+1
      END IF
C
C       2.0 OUTPUTS
C
      LOUT=LEN_TRIM(OUTPUTS)
      IF (LOUT.NE.0) THEN
      OUTPUTS=OUTPUTS(1:LOUT)//'\'
      LOUT=LOUT+1
      END IF
C
C     OPEN NEEDED FILES
      OPEN (7,FILE=OUTPUTS(1:LOUT)//'hls.OUT')
      OPEN (4,FILE=OUTPUTS(1:LOUT)//fout)
C
C
       call sjump(Npart,weight,fname,oflag,avgwt,stdwt)
C
C
C       DONE WITH ALL FILE PROCESSING
C
C#######################################################################
C#######################################################################
      PRINT *,'                '
     1
```

```
      PRINT *,'
     1                   '
      PRINT *,'          SUCCESSFUL RUN !!!!  HERE IS THE SAVED OUTPUT
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'          TWO FILES WERE GENERATED AS OUTPUT......
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'          FILE NO. 1 ...      NAME : SJ-HLS.OUT
     1                   '
      PRINT *,'                           CONTENT : FORCE-TIME HISTORIES
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'          FILE NO. 2 ...      NAME : HLS.OUT
     1                   '
      PRINT *,'                           CONTENT : STATISTICAL OUTPUT
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'
     1                   '
      PRINT *,'   HLS.FOR, Vs.1.0,  NOV. 1992......      B. KHAFAGI, MSU
     1                   '
      PRINT *,'
     1                   '
C#######################################################################
C#######################################################################
      STOP
      END
C     END OF THE MAIN PROGRAM DATA-P1.FOR
C#######################################################################
C
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ****************************************************************** *
C
C                    ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      SUBROUTINE SJ (SINGLE JUMP)
++++++++++++++++
C                    ++++++++++++++++++++++++++++++++++
C
C
        SUBROUTINE sjump(Npart,weight,fname,oflag,avgwt,stdwt)
      DIMENSION avgt(8), stdt(8),avga(8),stda(8),avgf(5),stdf(5)
      DIMENSION allin(22,5),ALLOUT(22,5),diff(22,5)
      DIMENSION covt(7,7),cova(8,8),covf(5,5),h(8,8),x(8)
      DIMENSION corf(5,5),crv(20),cort(7,7),cora(8,8)
      DIMENSION FXT(90), FYT(90), FZT(90), TIMET(90), AT(8), TT(8)
      DIMENSION A(1000,8),T(1000,8),PHI(1000),ALL(22,1000),DATA(1000)
      dimension Error(90),CONTROL(90),e(90)
      INTEGER WEIGHT(1000)
C     DIMENSION A(10,8),T(10,8),PHI(10),ALL(22,10),DATA(10)
C     INTEGER WEIGHT(10)
      LOGICAL*1 XABS
```

```
      CHARACTER INPUTS*40,outputs*40
      CHARACTER FNAME*12
      COMMON /CMP/ M,N,FS,IWIN,L,NFFT,DF
      COMMON /FILES/ INPUTS,OUTPUTS,LINP,LOUT
c
c  common statement to save the array u from function uni between calls
c
      common u(17)
C
C Read spectral estimation parameters for the error term
C
      var=0.0244720
      frmin=0.0
      frmax=10.0
      FS=37.
      dtt=1.0/FS
      nfft=128
      TIMESTEP=1.0/FS
      do 23 i=1,n
      timet(i)=timet(i-1)+timestep
      if(i.eq.1) timet(i)=0.0
23    continue
C
C
C                     +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      3. FILES' PROCESSING        ++++++++++++++++
C                     +++++++++++++++++++++++++++++++++
C
C
        npt=7
        npa=8
        npf=5
        read (1,110) avgphi,stdphi,(avgt(i),i=1,npt),(stdt(i),i=1,npt)
        do 112 i=1,npt
        read (1,114) (cort(i,j),j=1,npt)
112     continue
        read (1,116) (avga(i),i=1,npa),(stda(i),i=1,npa)
        do 118 i=1,npa
        read (1,119) (cora(i,j),j=1,npa)
118     continue
        read (1,122) (avgf(i),i=1,npf),(stdf(i),i=1,npf)
        do 124 i=1,npf
        read (1,126) (corf(i,j),j=1,npf)
124     continue
110     format (2(f7.3),/,7(f7.3),/,7(f7.3))
114     format (7(f7.3))
116     format (8(f7.3),/,8(f7.3))
119     format (8(f7.3))
122     format (5(f7.3),/,5(f7.3))
126     format (5(f7.3))
136     format (f12.8)
C
C
        allin(1,1)=avgwt
        allin(1,2)=stdwt
        allin(2,1)=avgphi
        allin(2,2)=stdphi
        do 167 i=1,7
        allin(i+2,1)=avgt(i)
        allin(i+2,2)=stdt(i)
167     continue
        do 168 i=1,8
        allin(i+9,1)=avga(i)
        allin(i+9,2)=stda(i)
168     continue
        do 169 i=1,5
        allin(i+17,1)=avgf(i)
        allin(i+17,2)=stdf(i)
169     continue
```

```
c
c
c       now the log normal varables are calculated
c

c ++++++++++++++++  phase lag
ccc     phivx2=(stdphi/avgphi)**2.0
ccc     stdphi=sqrt(log(1.0+phivx2))
ccc     avgphi=log(avgphi)-(log(1.0-phivx2)/2.0)
c ++++++++++++++++  loads as log normal
ccc     do 47 i=1,npa
ccc     if((i.eq.3).or.(i.eq.6)) then
ccc     vx2=(stda(i)/avga(i))**2.0
ccc     avga(i) =log(avga(i))-(log(1.0-vx2)/2.0)
ccc     stda(i) =sqrt(log(1.0+vx2))
ccc     endif
47      continue
ccc     do 45 i=1,npf
ccc     if((i.eq.1).or.(i.eq.3).or.(i.eq.5)) then
ccc     vx2=(stdf(i)/avgf(i))**2.0
ccc     avgf(i) =log(avgf(i))-(log(1.0-vx2)/2.0)
ccc     stdf(i) =sqrt(log(1.0+vx2))
ccc     endif
45      continue
c
c       now find the cov. matrix for amplit. (a), time t, force f
c       by multiplying correlation coefficient times st.dev (i), (j)
c
c
        do 125 i=1,npt
        do 127 j=1,npt
        covt(i,j)=cort(i,j)*stdt(i)*stdt(j)
127     continue
125     continue
        do 135 i=1,npa
        do 137 j=1,npa
        cova(i,j)=cora(i,j)*stda(i)*stda(j)
137     continue
135     continue
        do 155 i=1,npf
        do 157 j=1,npf
        covf(i,j)=corf(i,j)*stdf(i)*stdf(j)
157     continue
155     continue
c
c
      DO 380 K=1,Npart

C       ZERO THE FLAG Zi (USED IN THE LOAD MODELING PROCESS)
C

      Z1=0.0
      Z2=0.0
      Z3=0.0
      Z4=0.0
      Z5=0.0
      Z6=0.0
      Z7=0.0
      TI1=0.0
      TI2=0.0
C     now pick the random variables that represent the control points
C
c
c    response lag phi
c
c    a log-normally distributed deviation from the nominal value will
c    be used.
c    (mean=+0.28  s, std.dev.=0.09lbs) for men
c    see REFRENCE: .....
c
```

```
          rnum=rnor(0)
         avp=avgphi+rnum*stdphi
ccc      avp=exp(avp)
         phi(k)=avp
c
c       each of the time control will be assumed at this point noramlly
c       distributed with correlated random variables called for time t2-t7
c
          call hmatrix (covt,crv,npt,npt,h,x)
         tt(1)=0.00
         tt(2)=avgt(1)+crv(1)
         tt(3)=avgt(2)+crv(2)
         tt(4)=avgt(3)+crv(3)
         tt(5)=avgt(4)+crv(4)
         tt(6)=avgt(5)+crv(5)
         tt(7)=avgt(6)+crv(6)
         tt(8)=avgt(7)+crv(7)
          call hmatrix (cova,crv,npa,npa,h,x)
         at(1)=avga(1)+crv(1)
         at(2)=avga(2)+crv(2)
         at(3)=avga(3)+crv(3)
ccc      at(3)=exp(avga(3)+crv(3))
         at(4)=avga(4)+crv(4)
         at(5)=avga(5)+crv(5)
ccc      at(6)=exp(avga(6)+crv(6))
         at(6)=avga(6)+crv(6)
         at(7)=avga(7)+crv(7)
         at(8)=avga(8)+crv(8)
          call hmatrix (covf,crv,npf,npf,h,x)
         fxx=avgf(1)+crv(1)
ccc      fxx=exp(avgf(1)+crv(1))
         fxm=avgf(2)+crv(2)
ccc      fyx=exp(avgf(3)+crv(3))
         fyx=avgf(3)+crv(3)
         fym=avgf(4)+crv(4)
         DO 195 I=1,n
C
C          CALCULATE THE VALUES OF THE CONTROL MODEL CONTROL(I) FOR AMPLITU
C          ERROR BY USEIN THE SAMPLE TIME CONTROL POINTS INSTEAD OF MEAN TI
C          CONTROL POINTS.
C
         IF (TIMET(I).LT.AVP) THEN
         CONTROL(I)=0.0
         GO TO 195
         ELSE IF (TIMET(I).LT.(TT(2)+AVP)) THEN
         STEP=((AT(2)-AT(1))/(TT(2)-TT(1)))/FS
         CONTROL(I)=CONTROL(I-1)+STEP
         Z1=Z1+1.
         IF (Z1.EQ.1.0) CONTROL(I)=AT(1)
         GO TO 195
         ELSE IF (TIMET(I).LT.(TT(3)+AVP)) THEN
C
C       USE A SECOND DEGREE PARABOLA TO MODEL THIS PART.  THE INITIAL
C       CONDITIONS FOR THE PARABOLA IS A VALUE OF A(2) AT TIME=0 (AT T2)
C       AND A VALUE OF A(3) AT TIME=dt (AT T3) AND SLOPE OF 0 AT 0
C
C       F(t) = AT^2 + BT + C
C       AO=dA/dT^2
C       BO=0
C       CO=A2
C       dT=T3-T3
C       dA=A3-A2
C       T(6) IS THE REFRENCE INITIAL TIME
C
CC       DT=TT(3)-TT(2)
CC       DA=AT(3)-AT(2)
CC       AO=DA/DT**2.0
CC       BO=0.0
CC       CO=AT(2)
```

```
CC      CONTROL(I)=AO*TI1**2+BO*TI1+CO
CC      TI1=TI1+1.0/FS
        STEP=((AT(3)-AT(2))/(TT(3)-TT(2)))/FS
        CONTROL(I)=CONTROL(I-1)+STEP
        Z2=Z2+1.
        IF (Z2.EQ.1.0) CONTROL(I)=AT(2)
      GO TO 195
      ELSE IF (TIMET(I).LT.(TT(4)+AVP)) THEN
      STEP=((AT(4)-AT(3))/(TT(4)-TT(3)))/FS
      CONTROL(I)=CONTROL(I-1)+STEP
      Z3=Z3+1.
      IF (Z3.EQ.1.0) CONTROL(I)=AT(3)
      GO TO 195
      ELSE IF (TIMET(I).LT.(TT(5)+AVP)) THEN
      IF ((Z4.EQ.0.0).AND.(CONTROL(I-1).LT.-1.0)) CONTROL(I-1)=AT(4)
      STEP=((AT(5)-AT(4))/(TT(5)-TT(4)))/FS
      CONTROL(I)=CONTROL(I-1)+STEP
      Z4=Z4+1.
      IF (Z4.EQ.1.0) CONTROL(I)=AT(4)
      GO TO 195
      ELSE IF (TIMET(I).LT.(TT(6)+AVP)) THEN
      STEP=((AT(6)-AT(5))/(TT(6)-TT(5)))/FS
      CONTROL(I)=CONTROL(I-1)+STEP
      Z5=Z5+1.
      IF (Z5.EQ.1.0) CONTROL(I)=AT(5)
      GO TO 195
      ELSE IF (TIMET(I).LT.(TT(7)+AVP)) THEN
C
C     USE A SECOND DEGREE PARABOLA TO MODEL THIS PART.  THE INITIAL
C     CONDITIONS FOR THE PARABOLA IS A VALUE OF A(6) AT TIME=0 (AT T6)
C     AND A VALUE OF A(7) AT TIME=dt (AT T7) AND SLOPE OF 0 AT dT
C
C     F(t) = AT^2 + BT + C
C     AO=[A6-A7]/dT^2
C     BO=-2*A*dT
C     CO=A6
C     dT=T7-T6
C     T(6) IS THE REFRENCE INITIAL TIME
C
       DT=TT(7)-TT(6)
       AO=(AT(6)-AT(7))/DT**2.0
       BO=-2*AO*DT
       CO=AT(6)
       CONTROL(I)=AO*TI2**2+BO*TI2+CO
       TI2=TI2+1.0/FS
      GO TO 195
      ELSE IF (TIMET(I).LT.(TT(8)+AVP)) THEN
      STEP=((AT(8)-AT(7))/(TT(8)-TT(7)))/FS
      CONTROL(I)=CONTROL(I-1)+STEP
      Z7=Z7+1.
      IF (Z7.EQ.1.0) CONTROL(I)=AT(7)
      GO TO 195
      END IF
      CONTROL(I)=0.0
 195  CONTINUE
      do 196 i=1,8
      t(k,i)=tt(i)
      a(k,i)=at(i)
 196  CONTINUE
c
c
c
C
C                     +++++++++++++++++++++++++++++++++++    +++++++++++++++
C     ++++++++++++++++          5. ERROR MODELING
C                     +++++++++++++++++++++++++++++++++++
        call simu (VAR,FrMIN,FrMAX,DTt,NFFT,error)
C
      DO 210 I=1,n
```

```
      e(i)=error(i)
      FZt(I)=CONTROL(I)+e(i)
       Fxt(I)=e(i)/10.0
       Fyt(I)=e(i)/30.0
      ti=timestep*i-avp
      cut =ti-tt(6)
      if((ti.ge.tt(4)).and.(ti.le.(tt(5)+0.03)))then
      FZt(I)=CONTROL(I)
      endif
      if((cut.lt.timestep).and.(ti.ge.tt(6)))then
      fxt(i)=fxx
      fxt(i-1)=fxm
      fyt(i)=fyx
      fyt(i-1)=fym
      endif
 210  CONTINUE
C
C                    +++++++++++++++++++++++++++++++++++++
C     +++++++++++++     5. START THE FINAL OUTPUT        +++++++++++++
C                    +++++++++++++++++++++++++++++++++++++
C
C       PRINT TIME VS FORCES IN ACTUAL VERSUS ESTIMATED MODEL, AND ERROR
C
      IF (NPART.GT.10) THEN
      if(k.eq.1) then
      PRINT *,' FOR MORE THAN 10 PART., LOAD HIS1TORIES ARE NOT SAVED'
      PRINT *,' HOWEVER, CONTROL POINT STATISTICS ARE STILL SAVED.'
      PRINT *,' Press any key when ready.'
      read(*,*)
      endif
      ELSEIF (NPART.LE.20) THEN
      WRITE (4,240)
 240  FORMAT (///,10X,'TIME  ',3X,'Z-FORCE Ratio ',2X,'MODEL Ratio',2X,'
     1ERROR ARRAY'/)
      DO 250 I=1,90
      WRITE (4,260) TIMEt(I),FZT(I),CONTROL(I),E(I)
 250  CONTINUE
 260  FORMAT (7X,F8.3,2X,F14.6,2X,F14.6,2X,F14.8)
      WRITE (4,300) WEIGHT(K)
 300  FORMAT (//,1X,'WEIGHT(S) :',I5,)
C
C       FINALLY, WRITE A SUMMARY ABOUT THE FORCES.
C
      WRITE (4,310) fname
 310  FORMAT (20X,'    FORCES INFORMATION ',///,13X,'FINAL REPORT FILE N
     1AME =   ',A12///,5X,'POINT',5X,'TIME',5X,'AMP.'/)
      DO 330 I=1,8
      WRITE (4,320) I,T(K,I),A(K,I)
 320  FORMAT (1X,I3,3X,F8.3,3X,F14.6)
 330  CONTINUE
      WRITE (4,340) PHI(K)
 340  FORMAT (1X,'Response Lag (PHI) = ',F7.4)
      ENDIF
      PHIK=PHI(K)
C
C
C
C       CALL PLOTXY SUBROUTINE TO PLOT THE TIME VERSUS THE FORCE IN THE
C       X, Y, Z DIRECTIONS.
C
C       NOW FIND THE MAXIMUM VALUE OF THE FORCE FX,FY,FZ, AND USE EACH
C       FOR THE RANGE OF SCREEN PLOTS.
C
C
      FXMAX=0.0
      FYMAX=0.0
      FZMAX=0.0
      FXMIN=0.0
      FYMIN=0.0
```

```
      FZMIN=0.0
      CALL MAX (FXT,FXMAX,TX,TIMESTEP)
      CALL MAX (FYT,FYMAX,TY,TIMESTEP)
      CALL MAX (FZT,FZMAX,TZ,TIMESTEP)
      CALL MIN (FXT,FXMIN,TXM,TIMESTEP)
      CALL MIN (FYT,FYMIN,TYM,TIMESTEP)
      CALL MIN (FZT,FZMIN,TZM,TIMESTEP)
C
C
      FWEIGHT=WEIGHT(K)
      IF (NPART.GT.10) THEN
      IF (K.EQ.1) PRINT *,'NO OUTPUTS TO THE SCREEN .........    MEMORY
     1LIMITATIONS .... !!!'
      PRINT *,'SIMULATION NO. :   ',K
      ELSE
      IF (OFLAG.GT.0.0) THEN
      CALL OPEN (OFLAG,BW)
      ELSE
      GO TO 350
      END IF
350   CALL PLOTXY (TIMET,FZT,CONTROL,FZMAX,FWEIGHT,BW,K,Npart,FZMIN
     1,90,FNAME,AVP,TT,AT,8,FxT,fxMIN,fxMAX,fyt,fymin,fymax)
      ENDIF
C
C           +++++++++++++++++++++++++++++++++++++++++++++++++++
C  +++++++++++  8. Reporting control points to file "hls.OUT"  +++++
C           ++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IF (K.EQ.1) WRITE (7,360)
      WRITE (7,370) K,WEIGHT(K),PHIK,(TT(J),J=2,8),(AT(I),I=
     11,8),FXMAX,FXMIN,FYMAX,FYMIN,FZMAX
 360  FORMAT (//,25X,'OUTPUT SUMMARY',//,3X,'No  WT     PHI      T2
     1 T3       T4       T5       T6       T7       T8       A1       A2       A
     13       A4       A5       A6       A7       A8    FXmax    FXmin   FYmax
     1 FYmin    FZmax')
 370  FORMAT (1X,I3,1X,i4,21(2X,F6.3))
C
C
C        PREPARE DATA FOR THE STATISTICS PACKAGE
C
         ALL(1,K)=float(WEIGHT(K))
         ALL(2,K)=PHIK
         ALL(3,K)=tT(2)
         ALL(4,K)=tT(3)
         ALL(5,K)=tT(4)
         ALL(6,K)=tT(5)
         ALL(7,K)=tT(6)
         ALL(8,K)=tT(7)
         ALL(9,K)=tT(8)
         ALL(10,K)=aT(1)
         ALL(11,K)=aT(2)
         ALL(12,K)=aT(3)
         ALL(13,K)=aT(4)
         ALL(14,K)=aT(5)
         ALL(15,K)=aT(6)
         ALL(16,K)=aT(7)
         ALL(17,K)=aT(8)
         ALL(18,K)=FXMax
         ALL(19,K)=FXMin
         ALL(20,K)=FYMax
         ALL(21,K)=FYMin
         ALL(22,K)=FZMAX
C
C        DONE WITH ALL FILE PROCESSING
C
 380  CONTINUE
C
c
c        add counter for the no. of time to call mystat
```

```
c
        nms=22
      XABS=.false.
C
C       CALCULATE AND REPORT IMPORTANT STATISTICS FOR THE OBSERVED
ACTION
      do 600 i=1,nms
      do 610 k=1,NPART
      data(k)=all(i,k)
610   continue
      if(npart.eq.1)go to 611
      CALL MYSTAT (data,NPART,DMIN,DMAX,DMEAN,DSTD,DSE,XABS)
611   continue
c
c    same estimated variables in a new arrat ALLOUT in which the
parameter
c     i is the variable being processed (eg. weight, phi,..) and the
parameter
c    nout is the output variable number defined as follows:
c
      nest=2
      allout(i,1) =Dmean
      allout(i,2) =Dstd
600   continue
C
C     REPORT THESE VALUES TO THE SUMMERY OUTPUT FILE
C
c
c      BLANK LINE TO SEPARATE DATA
c
       do 666 i=1,22
        diff(i,1)=abs((allin(i,1)-ALLOUT(I,1))/allin(i,1)*100.0)
        diff(i,2)=abs((allin(i,2)-ALLOUT(I,2))/allin(i,2)*100.0)
666    continue
      WRITE (7,633)
 633   FORMAT (1X/)
      WRITE (7,361)
 361  FORMAT (//,25X,'OUTPUT SUMMARY',//,3X,'No  WT      PHI       T2
     1 T3       T4       T5       T6       T7       T8      A1       A2       A
     13       A4       A5       A6       A7       A8')
      WRITE (7,653) (ALLin(I,1),I=1,17)
      WRITE (7,650) (ALLOUT(I,1),I=1,17)
      WRITE (7,655) (diff(i,1),I=1,17)
      WRITE (7,654) (ALLin(I,2),I=1,17)
      WRITE (7,650) (ALLOUT(I,2),I=1,17)
      WRITE (7,655) (diff(i,2),I=1,17)
 650  FORMAT (' Sim',f6.0,f8.3,15(F8.3))
 653  FORMAT (1x,'Mean values',/,' Act',f6.0,f8.3,15(F8.3))
 654  FORMAT (1x,'Std. Deviations',/,' Act',f6.0,f8.3,15(F8.3))
 655  FORMAT (' Dif',3x,f3.0,'%  ',f5.1,'%  ',15(F5.1,'%  ')/)
C
      CALL ENDGRAPHICS ()
      RETURN
      END
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                      SUBROUTINE OPEN                            *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *   APRIL 15,1992           OPENING LOGO......                    *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ************************************************************    *
C
C
```

```
C                       +++++++++++++++++++++++++++++++++
C     ++++++++++++++++        1. DATA INPUT/OUTPUT        +++++++++++++++
C                       +++++++++++++++++++++++++++++++++
C
C
      SUBROUTINE OPEN (OFLAG,BW)
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,Y3PLOT
      INTEGER*2 BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     1TITLECOLOR,NUMBER,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,PLOT3COLOR
      CHARACTER TITLE*280,XAXISTITLE*80,YAXISTITLE*80
C       INCLUDE 'PLOT.DIF'
C       INITIALISE THINGS FOR THE FIRST PLOT
      INITGRAPHICS=.TRUE.
      INITBORDER=.TRUE.
      INITWINDOW=.TRUE.
      INITAXES=.FALSE.
      INITFONTS=.TRUE.
      PLOTMETHOD=.FALSE.
      Y2PLOT=.FALSE.
      Y3PLOT=.FALSE.
      SCRONLY=.TRUE.
      FONTHEIGHT=28
      FONTWIDTH=16
C       SET UP THE FANCY COLORS
      BORDERCOLOR=5
      WINDOWCOLOR=7
      LABELCOLOR=4
C       SET UP THE PLOT LIMITS
      XMIN=0.0
      XMAX=1.0
      YMIN=0.0
      YMAX=1.0
C       POSITION THIS PLOT AT THE TOP LEFT OF THE SCREEN
      XBOTTOMLEFT=0.0
      YBOTTOMLEFT=0.0
      XTOPRIGHT=1.0
      YTOPRIGHT=1.0
      TITLE='      **    S  J  -  H   L   S . F  O  R    **'
      IF(BW.EQ.1) THEN
      BORDERCOLOR=0
      WINDOWCOLOR=15
      LABELCOLOR=1
      END IF
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1XDATA,YDATA,Y2DATA,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,
     2YMIN,YMAX,XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,
     3YTOPRIGHT,PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,
     4PLOTCOLOR,TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,
     5FONTHEIGHT,FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,X3,Y3,NUM3)
C       WAIT FOR USER TO HIT RETURN
        READ(*,*)
Cc      CALL ENDGRAPHICS()
      OFLAG=0
      RETURN
      END
C       END OF MODULE   SUBROUTINE OPENING
C###################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *                       SUBROUTINE PLOTXY                         *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                 *
* *    APRIL 15,1992          PLOTTING FORCE-TIME HISTORIES         *
* *                                                                 *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ******************************************************************** *
C
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
      SUBROUTINE PLOTXY (TIME,FZ,PR,FZMAX,WEIGHT,BW,K,NF,FZMIN,N,
     1FNAME,PHIK,X3,Y3,NUM3,fx,fxMIN,fxMAX,fy,fymin,fymax)
      DIMENSION TIME(90), FZ(90), PR(90)
      DIMENSION X3(NUM3), Y3(NUM3),XNULL(8),YNULL(8)
C       INCLUDE 'PLOT.DIF'
      LOGICAL*1 INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1PLOTMETHOD,INVERTX,INVERTY,TITLEONLY,SCRONLY,Y2PLOT,Y3PLOT
c
      INTEGER*2 BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     1TITLECOLOR,NUMBER,PLOT2COLOR,FONTHEIGHT,FONTWIDTH,PLOT3COLOR
c
      CHARACTER TITLE*280,XAXISTITLE*80,YAXISTITLE*80,
     1FNAME*12,TITLE1*80,TITLE2*80,TITLE3*80,TITLE4*
     280,TITLE5*80,TITLE6*80,TITLE7*80,TITLE8*80,TITLE9*80,TITLE10*80,
     3TITLE11*80,TITLE12*80,TITLE13*80
      NUMBER=N
C
C       INITIALISE THINGS FOR THE FIRST PLOT
C
      INITGRAPHICS=.TRUE.
      INITBORDER=.TRUE.
      INITWINDOW=.TRUE.
      INITAXES=.TRUE.
      INITFONTS=.TRUE.
      PLOTMETHOD=.TRUE.
      TITLEONLY=.FALSE.
      SCRONLY=.FALSE.
      Y2PLOT=.false.
      Y3PLOT=.TRUE.
      FONTHEIGHT=10
      FONTWIDTH=5
C       SET UP THE FANCY COLORS DEPENDING ON THE VALUE OF CFLAG (0 OR 1)
      BORDERCOLOR=14
      IF (CFLAG.EQ.0.0) THEN
      WINDOWCOLOR=8
      CFLAG=1.0
      ELSE
      WINDOWCOLOR=1
      CFLAG=0.0
      END IF
      AXISCOLOR=10
      LABELCOLOR=11
      PLOTCOLOR=14
      PLOT2COLOR=4
      PLOT3COLOR=15
      TITLECOLOR=15
C
C                    +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. FORCE IN THE x-DIR      ++++++++++++++++
C                    +++++++++++++++++++++++++++++++++
C
      XMIN=0.0
      XMAX=2.5
      XTICSPACE=0.5
C
C       CREATE THE ZERO AXIS USING THE ARRAY XNULL
C
        XNULL(8)=xmax
        YNULL(8)=0.0
C     Call scale subroutine to set Ymin, Ymax, Yticspace,XMIN,XMAX,...
C
```

```
      CALL SCALE (1.3*fxmin,1.3*fxmax,YMIN,YMAX,YTICSPACE)
C
C       POSITION THIS PLOT
C
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.45
      XTOPRIGHT=0.50
      YTOPRIGHT=0.90
C       GIVE IT A TITLE
      TITLE='      Fx AS A RATIO TO WEIGHT        '
      XAXISTITLE='   T I M E ( s e c . )'
      YAXISTITLE='FORCE RATIO'
C
C     USE THE BLACK AND WHITE COMBINATION IF SCREEN CAPTURING IS NEEDED
C
      IF(BW.EQ.1) THEN
      BORDERCOLOR=0
      WINDOWCOLOR=15
      AXISCOLOR=1
      LABELCOLOR=4
      PLOTCOLOR=0
      PLOT2COLOR=1
      PLOT3COLOR=2
      TITLECOLOR=1
      END IF
C
C       READY TO CALL PROGRAM PLOT.FOR
C
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,Fx,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,XNULL,YNULL,NUM3)
C
C                   +++++++++++++++++++++++++++++++++
C     +++++++++++++++       1. FORCE IN THE y-DIR      +++++++++++++++
C                   +++++++++++++++++++++++++++++++++
C
C
      CALL SCALE (1.2*fymin,1.2*fymax,YMIN,YMAX,YTICSPACE)
C
C       POSITION THIS PLOT
C
      INITGRAPHICS=.FALSE.
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.45
      XTOPRIGHT=1.00
      YTOPRIGHT=0.90
C       GIVE IT A TITLE
      TITLE='      Fy AS A RATIO TO WEIGHT        '
      XAXISTITLE='   T I M E ( s e c . )'
      YAXISTITLE='FORCE RATIO'
C
C     USE THE BLACK AND WHITE COMBINATION IF SCREEN CAPTURING IS NEEDED
C
      IF(BW.EQ.1) THEN
      BORDERCOLOR=0
      WINDOWCOLOR=15
      AXISCOLOR=1
      LABELCOLOR=4
      PLOTCOLOR=0
      PLOT2COLOR=1
      PLOT3COLOR=2
      TITLECOLOR=1
      END IF
C
C       READY TO CALL PROGRAM PLOT.FOR
C
```

```
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,Fy,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,XNULL,YNULL,NUM3)
C
C                       +++++++++++++++++++++++++++++++++
C     ++++++++++++++++       1. FORCE IN THE Z-DIR       ++++++++++++++++
C                       +++++++++++++++++++++++++++++++++
C
C     Call scale subroutine to set Ymin, Ymax, Yticspace,XMIN,XMAX,...
C
      CALL SCALE (1.2*fzmin,1.2*fzmax,YMIN,YMAX,YTICSPACE)
C
C       POSITION THIS PLOT
C
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.00
      XTOPRIGHT=0.50
      YTOPRIGHT=0.45
C       GIVE IT A TITLE
      TITLE='      FZ AS A RATIO TO WEIGHT      '
      XAXISTITLE='   T I M E ( s e c . )'
      YAXISTITLE='FORCE RATIO'
C
C     USE THE BLACK AND WHITE COMBINATION IF SCREEN CAPTURING IS NEEDED
C
      IF(BW.EQ.1) THEN
      BORDERCOLOR=0
      WINDOWCOLOR=15
      AXISCOLOR=1
      LABELCOLOR=4
      PLOTCOLOR=0
      PLOT2COLOR=1
      PLOT3COLOR=2
      TITLECOLOR=1
      END IF
C
C       READY TO CALL PROGRAM PLOT.FOR
C
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,FZ,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,XNULL,YNULL,NUM3)
C                       +++++++++++++++++++++++++++++++++
C     +++++++++++++++       2. OUTPUT VALUES          ++++++++++++++++
C                       +++++++++++++++++++++++++++++++++
C
C       just plot the title on this one
      XBOTTOMLEFT=0.50
      YBOTTOMLEFT=0.00
      XTOPRIGHT=1.00
      YTOPRIGHT=0.45
      INITGRAPHICS=.FALSE.
C
      TITLE='                  CONTROL POINTS                    '
      WRITE (TITLE1,40)
 40   FORMAT ('Single Jump With One Person')
      WRITE (TITLE2,50) WEIGHT
 50   FORMAT ('Weight= ',F4.0,' lbs')
C
C     Find the max. value of force in the x,y directions (absolute force
C
      WRITE (TITLE3,60) K,NF
      I=0
      WRITE (TITLE13,80) PHIK
```

```
      I=1
      WRITE (TITLE4,70) I,X3(1),Y3(1)
      I=I+1
      WRITE (TITLE5,70) I,X3(2),Y3(2)
      I=I+1
      WRITE (TITLE6,70) I,X3(3),Y3(3)
      I=I+1
      WRITE (TITLE7,70) I,X3(4),Y3(4)
      I=I+1
      WRITE (TITLE8,70) I,X3(5),Y3(5)
      I=I+1
      WRITE (TITLE9,70) I,X3(6),Y3(6)
      I=I+1
      WRITE (TITLE10,70) I,X3(7),Y3(7)
      I=I+1
      WRITE (TITLE11,70) I,X3(8),Y3(8)
 60   FORMAT (I2,' OF'I3,1X,'POINT  TIME        AMP.')
 70   FORMAT (8X,I2,2X,F8.4,2X,F10.5)
 80   FORMAT (1X,'Response Lag (PHI) = ',F7.4)
      WRITE (TITLE12,90)
 90   FORMAT ('HIT ENTER TO CONTINUE....')
      CALL OUTPUTS (XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,TITLE,
     1TITLE1,TITLE2,TITLE3,TITLE4,TITLE5,TITLE6,TITLE7,TITLE8,TITLE9,
     2TITLE10,TITLE11,TITLE12,TITLE13,CFLAG,BW)
C
C
C                     +++++++++++++++++++++++++++++++++
C     ++++++++++++++++  3. Top Title on the Screen     ++++++++++++++++
C                     +++++++++++++++++++++++++++++++++
C
      INITGRAPHICS=.FALSE.
      TITLEONLY=.TRUE.
      INITAXES=.FALSE.
      INITFONTS=.TRUE.
      FONTHEIGHT=28
      FONTWIDTH=16
      WINDOWCOLOR=7
      TITLECOLOR=4
      XBOTTOMLEFT=0.00
      YBOTTOMLEFT=0.90
      XTOPRIGHT=1.00
      YTOPRIGHT=1.00
      TITLE='S  I  N  G  L  E    J  U  M  P'
C
C     USE THE BLACK AND WHITE COMBINATION IF SCREEN CAPTURING IS NEEDED
C
      IF(BW.EQ.1) THEN
      WINDOWCOLOR=15
      TITLECOLOR=1
      END IF
      CALL PLOT (INITGRAPHICS,INITBORDER,INITWINDOW,INITAXES,INITFONTS,
     1TIME,FZ,PR,NUMBER,PLOTMETHOD,INVERTX,INVERTY,XMIN,XMAX,YMIN,YMAX,
     2XTICSPACE,YTICSPACE,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     3PLOT2COLOR,BORDERCOLOR,WINDOWCOLOR,AXISCOLOR,LABELCOLOR,PLOTCOLOR,
     4TITLECOLOR,XAXISTITLE,YAXISTITLE,TITLE,TITLEONLY,FONTHEIGHT,
     5FONTWIDTH,SCRONLY,Y2PLOT,Y3PLOT,PLOT3COLOR,XNULL,YNULL,NUM3)
C
C       WAIT FOR USER TO HIT RETURN ( only for testing the program
C       , otherwise let the program run with no stops)
C
      READ (*,*)
      IF (K.EQ.NF+1) CALL ENDGRAPHICS ()
      RETURN
      END
C       END OF SUBROUTINE  PLOTXY
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
```

```
* *                       SUBROUTINE MAX                              *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *    MAY   05,1992         FINDING THE MAX. VALUE OF DATA           *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ***************************************************************** *
C
C
C                     ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                     ++++++++++++++++++++++++++++++++++
C
      SUBROUTINE MAX (DATA,FMAX,T,TIMESTEP)
C
C     TO FIND THE MAX. FORCE
C
      DIMENSION DATA(90)
      T=0.0
      N=0
      FMAX=-1E14
      NMAX=0
      DO 10 I=1,90
      N=N+1
      IF (DATA(I).GT.FMAX) THEN
      FMAX=DATA(I)
      NMAX=N
      GO TO 10
      END IF
 10   CONTINUE
      T=REAL(NMAX)*TIMESTEP
      RETURN
      END
C       END OF SUBROUTINE  MAX
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *                       SUBROUTINE MIN                              *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *    MAY   05,1992         FINDING THE MIN. VALUE OF DATA           *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ***************************************************************** *
C
C
C                     ++++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                     ++++++++++++++++++++++++++++++++++
C
      SUBROUTINE MIN (DATA,FMIN,T,TIMESTEP)
C
C     TO FIND THE MIN. FORCE
C
      DIMENSION DATA(90)
      T=0.0
      N=0
      FMIN=0.0
      NMAX=0
      DO 10 I=1,90
      N=N+1
      IF (DATA(I).LT.FMIN) THEN
```

```
      FMIN=DATA(I)
      NMAX=N
      GO TO 10
      END IF
 10   CONTINUE
      T=REAL(NMAX)*TIMESTEP
      RETURN
      END
C       END OF SUBROUTINE  MIN
C######################################################################
C######################################################################
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *                      SUBROUTINE outputs                           *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                   *
* *    MAY   15,1992        Plotting the final values of Forces       *
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C       START OF MODULE outputs.for
      INCLUDE 'FGRAPH.FI'
      SUBROUTINE OUTPUTS (XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,YTOPRIGHT,
     1TITLE,TITLE1,TITLE2,TITLE3,TITLE4,TITLE5,TITLE6,TITLE7,TITLE8,
     2TITLE9,TITLE10,TITLE11,TITLE12,TITLE13,CFLAG,BW)
      INCLUDE 'FGRAPH.FD'
      INTEGER*2 DUMMY,XWIDTH,YHEIGHT,IX1,IY1,IX2,IY2,IX,IY,FONTHEIGHT,
     1FONTWIDTH
      REAL*4 XW1,YW1,XW2,YW2,XBOTTOMLEFT,YBOTTOMLEFT,XTOPRIGHT,
     1YTOPRIGHT,YT
      CHARACTER TITLE*80,FONTCOMMAND*10,FONTSTRING*7,FONTFILE*12,TITLE1*
     180,TITLE2*80,TITLE3*80,TITLE4*80,TITLE5*80,TITLE6*80,TITLE7*80,
     2TITLE8*80,TITLE9*80,TITLE10*80,TITLE11*80,TITLE12*80,TITLE13*80
        RECORD           / VIDEOCONFIG / SCREEN
        RECORD           / WXYCOORD /    WXY
        RECORD           / XYCOORD /     POSITION
      COMMON SCREEN
C         SET UP FONT FOR USE
      FONTFILE='HELVB.FON'
      FONTCOMMAND="T'HELV'"
      IF (REGISTERFONTS(FONTFILE).LT.0) THEN
      WRITE (*,10) FONTFILE
 10   FORMAT (' FONT FILE ',A12,' NOT FOUND IN PLOT')
      STOP
      END IF
      FONTHEIGHT=28
      FONTWIDTH=16
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
 20   FORMAT ('H',I2.2,'W',I2.2,'B')
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
C       GET SCREEN DIMENSIONS
      XWIDTH=SCREEN.NUMXPIXELS
      YHEIGHT=SCREEN.NUMYPIXELS
C       SET A VIEW PORT, WITH CORNER COORDINATES DEFINED AS A
C       FRACTION OF THE TOTAL SCREEN SIZE
C       (0.0,0.0) IS BOTTOM LEFT OF SCREEN
C       (1.0,1.0) IS TOP RIGHT OF SCREEN
C       TRANSLATE CORNER COORDINATES TO PIXEL COORDINATES
      IX1=INT(XBOTTOMLEFT*FLOAT(XWIDTH))
      IX2=INT(XTOPRIGHT*FLOAT(XWIDTH))
      IY1=YHEIGHT-INT(YTOPRIGHT*FLOAT(YHEIGHT))
      IY2=YHEIGHT-INT(YBOTTOMLEFT*FLOAT(YHEIGHT))
      CALL SETVIEWPORT (IX1,IY1,IX2,IY2)
C       CREATE A REAL COORDINATE WINDOW
      XW1=0.0
      YW1=0
```

```
      XW2=1.0
      YW2=1.0
      DUMMY=SETWINDOW(.TRUE.,XW1,YW1,XW2,YW2)
C
C        SET THE COLOR OF THE BACKGROUND WINDOW DEPENDING ON THE FLAG
C        "CFLAG" ... NOTE THE SETUP HAS TO BE OPPOSITE TO THE PLOTXY
C
C
      IF (CFLAG.EQ.1.0) THEN
      DUMMY=SETCOLOR(8)
      ELSE
      DUMMY=SETCOLOR(1)
      END IF
      IF(BW.EQ.1) DUMMY =SETCOLOR(15)
      DUMMY=RECTANGLE_W($GFILLINTERIOR,XW1,YW1,XW2,YW2)
C        DRAW A BORDER AROUND THE VIEWPORT
      DUMMY=SETCOLOR(14)
      IF(BW.EQ.1) DUMMY =SETCOLOR(0)
      DUMMY=RECTANGLE_W($GBORDER,XW1,YW1,XW2,YW2)
C
C        This is the screen design of the data output
C
C
      TITLELENGTH=50
C
C        POSITION THE TITLE CENTERED (MORE OR LESS) ABOVE GRAPH
C
      YT=0.9*(YW2+YW1)
      CALL MOVETO_W (0.9*(XW2+XW1),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY-FONTHEIGHT/2
C
C        enter the program name (Title: from the main program)
C
      DUMMY=SETCOLOR(15)
      IF(BW.EQ.1) DUMMY =SETCOLOR(0)
      CALL MOVETO (IX,IY,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      IF(BW.EQ.1) DUMMY =SETCOLOR(0)
      CALL MOVETO (IX,IY+2,POSITION)
      CALL OUTGTEXT (TITLE(1:TITLELENGTH))
C
C        CHANGE THE FONT TO A FIXED FONT TO CREATE THE FORCES' TABLE
C        HERE, COURIER FONTS WILL BE USED.
C
C
      FONTFILE='courb.fon'
      FONTCOMMAND="T'courier'"
      FONTHEIGHT=12
      FONTWIDTH=9
      IF (REGISTERFONTS(FONTFILE).LT.0) THEN
      WRITE (*,10) FONTFILE
      STOP
      END IF
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
C
      YT=0.8*(YW2+YW1)
      CALL MOVETO_W (0.73*(XW2+XW1),YT,WXY)
      CALL GETCURRENTPOSITION (POSITION)
      IX=POSITION.XCOORD
      IY=POSITION.YCOORD
      IX=IX-(TITLELENGTH*FONTWIDTH)/2
      IY=IY-FONTHEIGHT/2
C
```

```
      DUMMY=SETCOLOR(0)
      IF(BW.EQ.1) DUMMY =SETCOLOR(1)
      CALL MOVETO (IX,IY+10,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      CALL MOVETO (IX,IY+25,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
      DUMMY=SETCOLOR(11)
      IF(BW.EQ.1) GOTO 22
      CALL MOVETO (IX,IY+11,POSITION)
      CALL OUTGTEXT (TITLE1(1:TITLELENGTH))
      CALL MOVETO (IX,IY+26,POSITION)
      CALL OUTGTEXT (TITLE2(1:TITLELENGTH))
22    CONTINUE
C
      FONTHEIGHT=10
      FONTWIDTH=8
      WRITE (FONTSTRING,20) FONTHEIGHT,FONTWIDTH
      DUMMY=SETFONT(FONTCOMMAND//FONTSTRING)
      DUMMY=SETCOLOR(14)
      IF(BW.EQ.1) DUMMY =SETCOLOR(0)
C
      CALL MOVETO (IX,IY+40,POSITION)
      CALL OUTGTEXT (TITLE3(1:TITLELENGTH))
      DUMMY=SETCOLOR(15)
      IF(BW.EQ.1) DUMMY =SETCOLOR(1)
      CALL MOVETO (IX,IY+55,POSITION)
      CALL OUTGTEXT (TITLE4(1:TITLELENGTH))
      CALL MOVETO (IX,IY+68,POSITION)
      CALL OUTGTEXT (TITLE5(1:TITLELENGTH))
      CALL MOVETO (IX,IY+81,POSITION)
      CALL OUTGTEXT (TITLE6(1:TITLELENGTH))
      CALL MOVETO (IX,IY+94,POSITION)
      CALL OUTGTEXT (TITLE7(1:TITLELENGTH))
      CALL MOVETO (IX,IY+107,POSITION)
      CALL OUTGTEXT (TITLE8(1:TITLELENGTH))
      CALL MOVETO (IX,IY+120,POSITION)
      CALL OUTGTEXT (TITLE9(1:TITLELENGTH))
      CALL MOVETO (IX,IY+133,POSITION)
      CALL OUTGTEXT (TITLE10(1:TITLELENGTH))
      CALL MOVETO (IX,IY+146,POSITION)
      CALL OUTGTEXT (TITLE11(1:TITLELENGTH))
C
      DUMMY=SETCOLOR(2)
      IF(BW.EQ.1) DUMMY =SETCOLOR(0)
      CALL MOVETO (IX,IY+164,POSITION)
      CALL OUTGTEXT (TITLE13(1:TITLELENGTH))
C
C
      DUMMY=SETCOLOR(0)
      CALL MOVETO (IX,IY+185,POSITION)
      CALL OUTGTEXT (TITLE12(1:TITLELENGTH))
      DUMMY=SETCOLOR(4)
      CALL MOVETO (IX,IY+186,POSITION)
      CALL OUTGTEXT (TITLE12(1:TITLELENGTH))
C
C       ALL DONE
C
      CALL UNREGISTERFONTS ()
      RETURN
      END
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                     *
* *                          SUBROUTINE SCALE                           *
* *                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *                                                                     *
* *    AUG   22,1992        VERTICAL PLOT SCALE                         *
```

```
* *                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * ****************************************************************** *
C                 +++++++++++++++++++++++++++++++++
C     ++++++++++++++++      1. DATA INPUT/OUTPUT       ++++++++++++++++
C                 +++++++++++++++++++++++++++++++++
C
      SUBROUTINE SCALE (FMIN,FMAX,ST,FIN,TICK)
C-----------------------------------------------------------------------
C This routine determines the starting and ending values to be used in
C a full scale plot.
C OUTPUT : ST = Starting value;  FIN = Ending value;  TICK .
C
C THIS IS A MODIFIED VERSION OF A PROGRAM WRITTEN ORIGINALLY BY R. HARIC
C-----------------------------------------------------------------------
      REAL*8 U,C
      TT=ALOG10(FMAX-FMIN)
      MT=IRINT(TT)
      TT=10.**(TT-MT)
      IF (TT.LE.2) THEN
      TT=2.
      ELSE IF (TT.GT.5) THEN
      TT=10.
      ELSE
      TT=5.
      END IF
      TICK=TT*10.**(MT-1)
      U=1.00001*TICK
      C=DLOG10(U)
      IF (C.LT.0) THEN
      L=IDINT(C)-1
      ELSE
      L=IDINT(C)
      END IF
      C=DBLE(10.)**L
      TICK=C*IDNINT(U/C)
      N1=IRINT(FMIN/TICK)
      N2=IRINT(FMAX/TICK)+1
      ST=N1*TICK
      FIN=N2*TICK
      RETURN
      END
      INTEGER FUNCTION IRINT(X)
      IF (X.LT.0) THEN
      IRINT=INT(X)-1
      ELSE
      IRINT=INT(X)
      END IF
      RETURN
      END
C######################################################################
C######################################################################
c    function uni(jdum): uses mcgill superduper to generate uniform
c    random numbers between 0 and 1.  the argument (jdum) is just a
c    dummy argument - i.e. it isn't used in the subprogram.  for
c    more information contact dr. george marsaglia washington state
c    university pullman,washington.
c
c    the function rstart is invoked once at the beginning of the main
c    program to initialize uni.  the arguments for rstart can be
c    chosen arbitrarily.
c
c*******************************************************************
c
c
      function uni(jdum)
      integer nbits,i,j,m,il
      common u(17)
```

```
      data nbits,i,j,m,i1/48,17,5,32707,1971/
c     THIS SEED WAS USED TO SIMULATE random motions
      jdum=0
      uni=u(i)-u(j)
      if(uni.lt.0.) uni=uni+1.
      u(i)=uni
      i=i-1
      if(i.eq.0) i=17
      j=j-1
      if(j.eq.0) j=17
      return
c
c  rstart initializes uni
c
      entry rstart(is,js)
c  initializes u(1) to u(17) bit-by-bit, using fibonacc1 mod m, lag 2
      i2=is
      i3=js
      do 2 l=1,17
      s=.5
      u(l)=0.
      do 2 ll=1,nbits
      k=i3-i1
      i1=i2
      i2=i3
      i3=k
      if(i3.gt.0) go to 2
      i3=i3+m
      u(l)=u(l)+s
2     s=.5*s
      rstart=is*10000+js
      return
      end
c
c *********************************************************************
c
c
c   function rnor(j): generates normally distributed random numbers be-
c   tween -infinity and +infinity with a mean=0.0 and a standard devia-
c   tion=1.0.
c
c  ********************************************************************
c
      function rnor(j)
c patchwork normal, 0<x<2.290094, linear and cubic pretests.
c
      j=0
      rnor=2.290094*(2*uni(0)-1.)
      x=abs(rnor)
      if(x.le.1.098) return
      y=uni(0)
      if(x+y.le.2.084) return
      if(.753398*x+y.lt.1.89122) go to 3
2     rnor=sign(2.290094-x,rnor)
      return
3     h=1.66244+x*(.5088388-x*(1.310391-x*.3513116))
      if(y.gt.h) go to 2
      q=2.71622-x*(1.961507-x*.3628954)
      if(y.le.q) return
      if((q+.0054.lt.y).and.(y.lt.h-.002)) go to 4
      if(alog(y).le..602802-.5*x**2) return
      if(.602802-.5*(2.290094-x)**2.gt.alog(2.-y)) go to 2
4     x=sqrt(5.244531-2.*alog(uni(0)))
      if(uni(0)*x.gt.2.290094) go to 4
      rnor=sign(x,rnor)
      return
      end
c
c
```

```
c  *******************************************************************
c
CCC     FROM DR. RONALD HARICHANDRAN
        subroutine hmatrix (cov,crv,m,n,h,x)
      DIMENSION cov(m,n),h(m,n),crv(m), x(m)
      REAL CIJ
130     format (1x,7(f7.3))
        DO 10 J=1,N
          H(J,1)=COV(J,1)/SQRT(COV(1,1))
          IF(J.EQ.1)GOTO 10
             DO 30 K=1,J
             IF(K.EQ.1)GOTO 20
             IF(K.EQ.J) THEN
                CIJ=0.0
                  DO 40 I=1,J-1
                  CIJ=CIJ+(H(J,I))**2
40                CONTINUE
                H(J,J)=SQRT(COV(J,J)-CIJ)
                GOTO 10
             END IF
          CIJ=0.0
                DO 50 I=1,K-1
                CIJ=CIJ+H(J,I)*H(K,I)
50              CONTINUE
          H(J,K)=(COV(J,K)-CIJ)/H(K,K)
20        H(K,J)=0.0
30           CONTINUE
10     CONTINUE
c
c      now multiply the matrix h time a vextor of standard R.V. to get
c      the correlated random variables
c
       do 70 i=1,m
       x(i)=rnor(0)
70     continue
     DO 80 I=1,M
          Crv(I)=0.0
          DO 90 J=1,N
             Crv(I)=Crv(I)+h(I,J)*x(J)
 90       CONTINUE
 80  CONTINUE
       RETURN
       END
C######################################################################
C######################################################################
      SUBROUTINE MYSTAT(X,Nn,XMIN,XMAX,XMEAN,XSTD,XSE,XABS)
      DIMENSION X(Nn)
      LOGICAL*1 XABS
      XMAX=-1E14
      XMIN=1e14
      XSUM=0.0
      XSUM2=0.0
      DO 10 I=1,Nn
      if(xabs) then
      DO 20 II=1,Nn
      X(I)=ABS(X(I))
20    CONTINUE
      END IF
C
C     DETERMINE THE MEAN
C
      XSUM=XSUM+X(I)
      XSUM2=XSUM2+X(I)**2.0
C
C     FINDING MAXIMUM AND MINIMUM
C
      IF (X(I).GT.XMAX) THEN
      XMAX=X(I)
      END IF
```

```
      IF (X(I).LT.XMIN) THEN
      XMIN=X(I)
      END IF
 10   CONTINUE
      XN=FLOAT(Nn)
      XMEAN =XSUM/XN
      XSTD=SQRT(XSUM2/XN-XMEAN**2.0)
      XSE=XSTD/SQRT(XN)
      RETURN
      END
C######################################################################
c
c----------------------------------------------------------------------
--------
c Program to generate a random function from a specified SDF
c Calls: fft, urng from nswclib
c----------------------------------------------------------------------
--------
        SUBROUTINE SIMU(VAR,FrMIN,FrMAX,DT,NFFT,error)
        complex x(128)
        real r(513),error(90)
        intrinsic sin, cos, cabs
        parameter (pi=3.141592654, maxfft=1024)
        data x / 128*(0., 0.)/
c Define arithmetic function to compute area under two-sided SDF
        sint(f1,f2) =(f2-f1)*(s(f2)+s(f1))/2
        df = 1./dt/nfft
        dfo2 = df/2
        nf = nfft/2 + 1
        do 12 ii=1,513
        r(ii)=uni(0)
12      continue
c take care of end regions
        nf1 = ifix(frmin/df) + 1
        nf2 = ifix(frmax/df) + 2
        f1 = frmin
        df1 = frmin - (nf1-1)*df
        if (df1 .gt. dfo2) then
          phi = r(nf1+1)*2.*pi
          if (frmax-frmin .le. df) then
       x(nf1+1)=sqrt(abs(sint(frmin,frmax)))*cmplx(cos(phi),sin(phi))
            go to 35
          end if
          n1 = nf1 + 2
          f2 = df*nf1 + dfo2
          x(nf1+1) = sqrt(abs(sint(f1,f2))) * cmplx(cos(phi),sin(phi))
        else
          phi = r(nf1)*2.*pi
          if (frmax-frmin .le. df) then
            x(nf1)=sqrt(abs(sint(frmin,frmax)))*cmplx(cos(phi),sin(phi))
            go to 35
          end if
          n1 = nf1 + 1
          f2 = df*(nf1-1) + dfo2
          x(nf1) = sqrt(abs(sint(f1,f2))) * cmplx(cos(phi), sin(phi))
        end if

        f2 = frmax
        df2 = (nf2-1)*df - frmax
        if (df2 .gt. dfo2) then
          n2 = nf2 - 2
          f1 = (nf2-2)*df - dfo2
          phi = r(nf2-1)*2.*pi
          x(nf2-1) = sqrt(abs(sint(f1,f2))) * cmplx(cos(phi), sin(phi))
        else
          n2 = nf2 - 1
          f1 = (nf2-1)*df - dfo2
          phi = r(nf2)*2.*pi
          x(nf2) = sqrt(abs(sint(f1,f2))) * cmplx(cos(phi), sin(phi))
```

```
        end if

c Assign transforms to middle region

        f1 = (n1-1)*df
        s1 = s(f1)
        do 30 i = n1, n2
          f2 = f1 + df
          s2 = s(f2)
          ssint = df * (s1+s2)/2
          phi = r(i)*2.*pi
          x(i) = sqrt(abs(ssint)) * cmplx (cos(phi), sin(phi))
          f1 = f2
          s1 = s2
30      continue

c Make x conjugate symmetric

35      n1 = max0 (2, nf1)
        n2 = min0 (nf2, nf-1)
        do 40 i = n1, n2
40      x(nfft+2-i) = conjg(x(i))

c Let x(1)=0 (zero freq.), and im[x(nf)]=0 (Nyquist freq.). This ensures
zero
c     mean, real, series. Power at Nyquist freq. must also be doubled.
c     Power at zero freq. (=2*cabs(x(1))**2) is lost.

        x(1) = cmplx (0., 0.)
        x(nf) = sqrt(2.) * cmplx (cabs(x(nf)), 0.)

c Obtain inverse fft

        call fft (x, nfft, 1)

c Scale x to obtain target variance

        sum = 0.
        sum2 = 0.
        do 50 i = 1, nfft
        x(i) = x(i)
        sum = sum + real(x(i))
50      sum2 = sum2 + cabs(x(i))**2
        sum = sum/nfft
        sum2 = sum2/nfft - sum*sum
        vcrn = sqrt(abs(var/sum2))
c
c
c
        do 60 i = 1, 90
        x(i) = (x(i) - sum) * vcrn
        error(i)=real(x(i))
60      continue
        end
c
c
      function s(f)
c-----------------------------------------------------------------------
--------
c Compute the value of the error spectrum at freq. F
c-----------------------------------------------------------------------
--------
      dimension freq(512),process(512)
      if(ijump.eq.0) then
        READ (1,10) ifreq
10      formAT(I5)
        do 144 i=1,512
        read (1,138)freq(i),process(i)
144     continue
```

```
 138     format (2f16.11)
         ijump=1
        fstep =freq(2)
        endif
        itest=int(f/fstep)+1
        i=itest
        if(f.eq.freq(i)) then
        s=process(i)
        elseif((f.gt.freq(i)).and.(f.le.freq(i+1))) then
        fper=(f-freq(i))/fstep
        s=(process(i+1)-process(i))*fper+process(i)
        endif
        return
        end
C#######################################################################
C#######################################################################
      SUBROUTINE FFT (X,N,INV)
C-----------------------------------------------------------------------
C This subroutine computes the discrete Fourier Transform or the inverse
C     transform of X.
C-----------------------------------------------------------------------
C Input:   X = 2**M complex array that initially contains the input
C                and on return contains the transform;
C          N = 2**M points. (Must be power of 2, else infinite loop resu
C          INV = 0 for direct transform; 1 for inverse transform.
C-----------------------------------------------------------------------
C
      COMPLEX X(*),U,W,T
      M=ALOG(FLOAT(N))/ALOG(2.)+.1
      NV2=N/2
      NM1=N-1
      J=1
      DO 40 I=1,NM1
      IF (I.GE.J) GO TO 10
      T=X(J)
      X(J)=X(I)
      X(I)=T
 10   K=NV2
 20   IF (K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
 30   J=J+K
 40   CONTINUE
      PI=4.0*ATAN(1.0)
      DO 70 L=1,M
      LE=2**L
      LE1=LE/2
      U=(1.0,0.0)
      W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      IF (INV.NE.0) W=CONJG(W)
      DO 60 J=1,LE1
      DO 50 I=J,N,LE
      IP=I+LE1
      T=X(IP)*U
      X(IP)=X(I)-T
      X(I)=X(I)+T
 50   CONTINUE
      U=U*W
 60   CONTINUE
 70   CONTINUE
      IF (INV.EQ.1) RETURN
      DO 80 I=1,N
      X(I)=X(I)/N
 80   CONTINUE
      RETURN
      END
C#####################################################################
```

LIST OF REFERENCES

1. Allen, D.E., "Building Vibrations from Human Activities," Concrete International:Design and Construction, American Concrete Institute, Vol. 12, No. 6, June 1990, pp. 66-73.

2. Allen, D.E., "Floor Vibration from Aerobics," Canadian Joumal of Civil Engineering, Vol.17, No. 5, May 1990, pp. 771-779.

3. Allen, D.E., and Pemica, G., "Floor Vibration Measurements in a Shopping Center," Canadian Journal of Civil Engineering, Vol. 9, No. 2, June 1982, pp. 149-155.

4. Allen, D.E., Pemica, G., and Rainer, J.H., "Building Vibration Due to Human Activities," Proceedings Sixth Annual Structures Congress, Orlando, Florida, Aug. 1987, pp. 438-447.

5. Allen, D.L., and Swallow, J.C., "Annoying Floor Vibrations--Diagnosis and Therapy," Sound and Vibration, Vol. 9, No. 3, Mar. 1975, pp. 12-17.

6. American Standards Building Requirements, "American Standard Specifications for Portable Steel and Wood Grandstands," ASA, Z20.1-1941, May 1941.

7. Bachmann, H., "Case Studies of Structures with Man-Induced Vibration," Journal of Structural Engineerings, ASCE, Vol. 118, No. 3, March 1992, pp. 631-647.

8. Bachmann, H., "Vibration of Building Structures Caused by Human Activities; Case Study of a Gymnasium," Technical Translation 2077, National Research Council of Canada, Ottawa, Ontario, Canada.

9. Bachmann, H., "Vibration Upgrading of Gymnasia, Dance Halls, and Footbridges", Structural Engineering International, Feb. 1992, pp. 118-124.

10. Blank, L. T., "Statistical Procedures for Engineering, Management, and Science," McGraw Hill, New York, N. Y., 1980.

11. Box, George E. P., "Time series analysis; forecasting and control," San Francisco, Holden-Day, 1970.

12. Burke, G.G., A. Ebrahimpour, J.S. Haldeman, J.J. Pinkard, R.L. Sack, W.E. Saul and G.L. Thinnes, "Automated Data Acquisition: CAE for the Experimentalist," UPCAEDM '85 Proceedings, July 1985.

13. Clough, R. W., and J. Penzien, "Random Processes," Dynamics of Structures, 1st ed., Mcgraw-Hill, New York, 1975.

14. Ebrahimpour, A., "Modeling Spectator Induced Dynamic Loads," A National Science Foundation study report presented to the University of Idaho, Moscow, in partial fulfillment of the requirements for the degree of Doctor of Philosophy with a Major in Civil Engineering, 1987.

15. Ebrahimpour, A. and R.L. Sack, "Modeling Dynamic Occupant Loads," Journal of Structural Engineering, ASCE, Vol. 115, No. 6, June, 1989, pp. 1476-1496.

16. Ebrahimpour, A. and R.L. Sack, "Statistical Analysis of Occupant Dynamic Loads," Proc.ASCE Specialty Conference on Probabilistic Mechanics and Structural and Geotechnical Safety, Blacksburg, Virginia, May 1988.

17. Ebrahimpour, A., R.L. Sack and P.D. VanKleek, "Computing Crowd Loads Using a Nonlinear Equation of Motion," Proc. CIVIL-COMP89, Sept. 19-21, 1989, London.

18. Eliashakoff, I., "Probabilistic Methods in the Theory of Structural," John Wiley & Sons, New York, N.Y., 1983.

19. Ellingwood, B. et al., "Structural Serviceability: a Critical Appraisal and Research Needs," Journal of Structural Engineering, ASCE, Vol. 112, No. 12, December 1986, pp. 2646-2664.

20. Ellingwood, B., Galambos, T., MacGregor, J., Cornell, C.,"Development of a Probability Based Load Criterion for American National Standard A58," NBS Special Publication 577, National Bureau of Standards, 1980.

21. Gibra, Isaac N., "Probability and statistical inference for scientists and engineers," Englewood Cliffs, N.J., Prentice-Hall, 1973.

22. Golenko, D., and Smiryagin, V., "Random-number generator for the "Strela" electronic computer, Computing Center AN SSSR, 1960.

23. Greimann, L.F., and Klaiber, F.W., "Dynamic Forces Induced by Spectators," Journal of the Structural Division, ASCE, Vol. 104, No. ST2, Feb. 1978, pp.348-351.

24. Griffiths & Hill, "Applied statistics algorithms," Chichester : Ellis Horwood for the Royal Statistical Society, c1985.

25. Harichandran, R. Unpublished notes and fortran programs for stocastic processes, 1992.

26. Helstrom, Carl W., "Probability and stochastic processes for engineers," New York, Macmillan ; London : Collier Macmillan, c1984.

27. Jenkins, Gwilym M., "Spectral analysis and its applications," San Francisco, Holden-Day, 1968.

28. Johnson, Mark E., "Multivariate statistical simulation," New York, Wiley, c1987.

29. McDonald, B., "The Dynamic Loading Due to Stadium Crowds; A Statistical Measure of the Coherency of Human Movement, " Independent study report presented to the University of Wisconsin, Madison, in partial fulfillment of the requirements for the degree of Master of Master of Science in Civil Engineering, 1984.

30. Motooka, Y., "Methods of obtaining random numbers by means of digital computer (Japanese)," Journal of Inst. Elect. Eng. Japan, V. 74, No. 5, 1954, pp. 579-588.

31. Murray, T., "Building Floor Vibration," AISC Journal, Mar. 1991, pp. (19-1)-(19-18).

32. Neumann, J., "Various techniques used in connection with random digits." NBS applications, Math. Series, No. 12, 1951, pp. 36-38.

33. Newland, D. E., An Introduction to Random Vibrations and Spectral Analysis, 1st ed., Longman Group Limited, 1975.

34. Priestley, M. B.," Spectral analysis and time series," London ; New York : Academic Press, 1981.

35. Probabilistic Methods in Structural Engineering, G. Augusti, Chapman and Hall, New York, 1984.

36. Robinson, Enders A., "Digital signal processing and time series analysis," San Francisco, Holden-Day, c1978.

37. Saul, W.E. and C.Y. Tuan, "Review of Live Loads Due to Human Movements," Journal of Structural Engineerings, ASCE, Vol. 112, No. 5, May 1986, pp. 995-1004.

38. Saul, W.E., C.Y. Tuan and B. McDonald, "Loads due to Human Movement," Structural Safely Studies, Symposium Proceedings, Structural Division, ASCE, Denver, May 1985, pp. 107-117.

39. Tilden, C.J., "Kinetic Effect of Crowds," Proceedings, ASCE, Vol. 34, No. 3, Mar. 1913, pp. 325-340.

40. Tuan, C.Y., "Loads Due to Human Movements on Assembly Structures", Ph.D. dissertation, Department of Civil & Environmental Engineering, Univ. of Wisconsin, Madison, July 1983.

41. Uniform Building Code, 1985 Edition, Section 2304(a), pp. 108 and 124-127.

42. Vankleek, P.D., "Measurement of Crowd-Induced Dynamic Loads," thesis presented to the University of Idaho, Moscow, in partial fulfillment of the requirements for the degree of Master of Science in Civil Engineering, Sep 1988.