LA-NUREG-6473-MS
Informal Report
NRC-8

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

A Computer Method for Analyzing HTGR Core Block Response to Seismic Excitation

by

J. L. Merson
J. G. Bennett

NOTICE
This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any legal warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

✓
Manuscript completed: August 1976
Issued: September 1976

Prepared for the US Nuclear Regulatory Commission
Office of Nuclear Regulatory Research

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

A COMPUTER METHOD FOR
ANALYZING HTGR CORE BLOCK RESPONSE TO SEISMIC EXCITATION

by

J. L. Merson and J. G. Bennett

ABSTRACT

An analytical method is developed to describe the seismic response of a High Temperature Gas-Cooled Reactor core. Models characterized by a small number of degrees-of-freedom are used to examine the large-deflection inelastic behavior of HTGR core block assemblages. The equations of motion are integrated numerically by Newmark's method, and close tolerances on the solution are enforced by an equilibrium iteration scheme. Some uses of this method are mentioned.

I. INTRODUCTION

The FYSMOD computer program was developed to study the seismic response of a High Temperature Gas-Cooled Reactor (HTGR) core.¹ The study of an HTGR core is a complex problem because the core consists of about 5000 hexagonal graphite fuel blocks with gaps and cooling channels, control rods, and spring packs stacked in a relatively rigid pre- and poststressed concrete pressure vessel. The primary core material response is nonlinear to stress, and the block clearances introduce additional nonlinearity. The extreme complexity of the HTGR core creates the need for a simple and dependable analytical tool for studying its seismic response.

Ignoring the realities of computer time and memory capacity, an ideal analytical model would treat the core in full detail. This ideal model would have all separate physical characteristics of the system represented. It would also be a three-dimensional model with all degrees-of-freedom allowed, easy to modify, and its output would be meaningful to the analyst. A number of simplified models have been developed and methods to date have centered around lumped spring-mass studies.^{1,2,3,4}

The one-dimensional model developed here is shown in Fig. 1. Although the model is shown as four blocks in an elastic box, FYSMOD is capable of representing up to 64 blocks and could readily be expanded to accommodate more. The free-body diagram shown in Fig. 2 identifies the forces that are considered on each block. An example of treating separate effects separately is found in the treatment of energy loss. Rather than combine all energy losses into a single viscous damping term or use a coefficient of restitution, we separate the friction losses from material hysteretic losses, and represent the impact energy loss with a material hysteresis loop.

The code described in the following sections is the one-dimensional version of FYSMOD. A two-dimensional version was also developed and used. Its

Fig. 1. One-dimensional model.

Fig. 2. Free-body diagram of physical model component.

approach is the same as used in the one-dimensional version. The extensions to the model required for the two-dimensional version are described in Appendix A.

The remainder of the report contains a detailed description of the equilibrium-iteration method and energy dissipation models used. Also given are a summary of program capabilities and limitations and some applications. Input instructions are given in Sec. V. A source listing of the one-dimensional program is found in Appendix B. A brief description of a plotting routine for displaying results from FYSMOD is also given.

II. THE EQUILIBRIUM-ITERATION SCHEME

The method used to solve the response of a nonlinear system is the equilibrium-iteration method combined with the component-element approach.⁴ A flow-chart of the equilibrium-iteration scheme is shown in Fig. 3. The method is a time-history approach that traces each mass through a succession of equilibrium configurations. Each succeeding configuration is based on the preceding state, i.e., the preceding equilibrium state is used as a set of initial conditions for the new state.

Referring to Fig. 3, it should be noted that the boundary conditions must be known functions of time for input to this program. These can be either analytic functions or time-histories. This step presumes that the core response can be

Fig. 3. Equilibrium-iteration scheme.

decoupled from the Prestressed Concrete Reactor Vessel (PCRV). This is a reasonable assumption since the mass of an actual HTGR core is less than one-tenth that of the PCRV. A separate lumped mass analysis would supply PCRV motion for input to this code. However, this is not a necessary step for determining the influence of parameters on core response.

A. Integration Method

Referring to Step 3 of Fig. 3, it is clear that a temporal integration scheme must be chosen. Studies have been made comparing integration schemes of linear and nonlinear systems,^{5,6} and each method has its advantages. Since it is not our purpose to study the various schemes, our choice was governed by the relative ease of programming the method, and the Newmark β method with $\beta = 1/4$ was chosen.⁷ Other acceptable methods can be utilized. All stable schemes exhibit similar gross amplitude and frequency response characteristics, and differences appear to be in the accuracy of a given method.

It is desirable to have the time-histories of individual masses for 10 seconds or more to realistically represent seismic events. However, the accumulation of roundoff errors occurring in each of the large number of integration steps destroys any accuracy advantage of one integration scheme over another. It is believed that any of the schemes will preserve the characteristic magnitudes of the forces and motions of the system.

Given the acceleration $\ddot{x}_i^t$ of a block i at time t , Steps 2 and 3 of Fig. 3 apply the following equations.

$$\ddot{x}_i^{t+dt} = \ddot{x}_i^t, \quad (1)$$

then

$$\dot{x}_i^{t+dt} = \dot{x}_i^t + \frac{\ddot{x}_i^{t+dt} + \ddot{x}_i^t}{2} dt \quad (2)$$

and

$$x_i^{t+dt} = x_i^t + \dot{x}_i^t dt + (0.5 - \beta) \ddot{x}_i^t dt^2 + \beta \ddot{x}_i^{t+dt} dt^2, \quad (3)$$

where

x_i^t = displacement of the i^{th} block at time t ,

$\dot{x}_i^t$ = velocity of the i^{th} block at time t ,

$\ddot{x}_i^t$ = acceleration of the i^{th} block at time t , and

β = Newmark numerical parameter ($1/6 \leq \beta \leq 1/4$).

B. Component Elements

From Step 3 of Fig. 3, the displacements obtained by Eq. (3) were based on Eq. (1). In general, the block acceleration at time $t+dt$ is not equal to the acceleration at time t , since forces acting on the block vary with time. Thus, the displacements given by Eq. (3) are correct only if $\ddot{x}_i^{t+dt}$ has the correct value at this point in the solution. Step 4 is to determine the forces acting on the block corresponding to block displacements. These forces are given by the component element under consideration at the time. If normal side contact is considered, the element is a nonlinear spring, Fig. 4.

Figure 4 illustrates typical component elements for side contact forces. The problem is inherently a nonlinear one even with a linear material model, because of clearance between core blocks, Fig. 4a. In addition, a nonlinear material can be used, Fig. 4b. The information for these force-versus-relative-displacement curves can be obtained from

Fig. 4a. Typical force displacement curve for bi-linear model.

Fig. 4b. Typical force displacement curve for non-linear model.

any other desired analysis. To simulate the stiffness of the HTGR core block, a finite element model of the problem (Fig. 5) is used to establish curves of force-versus-nodal-displacement for nodes along the pressure face.

Curves for three such nodal points are shown in Fig. 6. Curve B shows the effect of the absence of a hole behind a nodal point, while Curve C illustrates the decreased stiffness from bending of the material at the edge of the loaded surface. The nodal force-versus-displacement curves are used to compute an equivalent spring constant for each node. Each node is then treated as a spring attached to a rigid boundary that kinematically remains plane as it is displaced. The springs are combined in parallel, and an equivalent single spring constant for flat-to-flat impact is derived. In this manner the distributed properties of the components are lumped. Here, the value of the equivalent spring constant is found to be 2.27×10^6 kN/m.

C. Equilibrium

The forces found corresponding to the displacements given by predicted accelerations are

Fig. 5. Finite element model for fuel block stiffness calculations.

checked by applying the equilibrium condition in Step 5, Fig. 3. Referring to Fig. 2, the acceleration of the free body is given by

$$\ddot{X}_i^{t+dt} = \frac{1}{M_i} (F_{LC} - F_{RC} - f), \quad (4)$$

where

- M_i = mass of the i^{th} block,
- F_{LC} = contact force on the left side,
- F_{RC} = contact force on the right side, and
- f = friction force.

The acceleration from Step 5, Fig. 3, is now compared to that predicted by Eq. (1) and the accuracy of predicted accelerations evaluated, based

Fig. 6. Typical nodal force displacement curves for fuel block stiffness.

on how well equilibrium is satisfied. This comparison is made on the basis that

$$\frac{\ddot{X}_{\text{computed}}^{t+dt} - \ddot{X}_{\text{predicted}}^{t+dt}}{\ddot{X}_{\text{computed}}^{t+dt}} \leq C_f, \quad (5)$$

where C_f is the convergence factor, a measure of how accurately the scheme solves the equations during a given time step. However, C_f is not a measure of over-all solution accuracy, as this is a function of time step size and integration accuracy.

If Eq. (5) is satisfied at this point in the iteration loop, time is incremented and all variables recorded as the time-history of the i^{th} block. The iteration begins again for the next time step, with the new initial conditions being those accelerations, velocities, and displacements just determined.

If Eq. (5) is not satisfied, the accelerations found in Step 5, Fig. 3, are used as new predicted accelerations, and the solution begins again with Step 3. Thus, the numerical method of solution is that of successive substitutions.

The equilibrium-iteration method resolves into finding the solution to the set of differential equations:

$$[M] \{\ddot{X}\} + [K] \{X\} = \{F(t)\}.$$

For such a system of equations describing the motion of blocks separated by gaps, the response time constants for a single block vary over a wide range. For a block not in contact with another, the time constant is very long (infinite). For blocks undergoing impact, the response time is $\sqrt{\frac{M}{K_{eq}}}$, where K_{eq} is an equivalent spring constant for impact. For example, if a block stiffness of $K_B = 1.89 \times 10^6$ kN/m is used, with a mass $M = 99.4$ kg, the time constant T is 2.3×10^{-4} s. For numerical accuracy the time step should be no larger than 1/10 the smallest natural period found from this time constant. In this case, the natural period would be 1.4×10^{-3} s/cycle, giving a time step of 1.4×10^{-4} s. This step size is the maximum one for reasonable accuracy, and 1/100 of the natural period is recommended.

This system of equations represents a very stiff system, requiring long computational time even for modest time-histories. For long time-

histories, round-off errors are significant and this may be verified from scaling law studies described in Ref. 3. In these studies a computer-built prototype and true scale model are simulated to check the scaling law formulation. In theory, the model and prototype results should be exactly the same at all homologous times. However, round-off errors accumulate after a number of impacts until phasing changes occur, and a slightly different problem is then being solved.

III. ENERGY DISSIPATION MODELS

An effort has been made to separate the system energy dissipation into two parts: frictional dissipation and material hysteretic losses.

A. The Friction Model

Two choices for representing breakaway frictional effects are shown in Fig. 7. Figure 7a represents the local stiffness type of model, where frictional force is proportional to the displacement between the base and block until a breakaway force is reached. This model is easily programmed, but has some difficult features such as defining the value of local stiffness accurately and describing the local stiffness variation as a function of a time-varying normal force. The model implemented here is based on Fig. 7b and is the type generally presented in elementary particle mechanics courses.

First, a decision is made based on the relative velocity between the base and the block. If relative velocity exists, the frictional force applied to the block opposes the velocity in sign and is $\mu_k F_N$ in magnitude, where μ_k is the kinetic coefficient of friction and F_N is the normal force.

Fig. 7a. Local stiffness friction model.

Fig. 7b. Proportional frictional force model.

If no relative velocity exists between the base and block, the no-slip condition exists, and the block and base accelerations are the same. Referring to the free body diagram in Fig. 7b, the friction force necessary for dynamic equilibrium is given by

$$f = M\ddot{X}_B + F_R - F_L, \quad (6)$$

where $\ddot{X}_B$ is the base acceleration, and F_R and F_L are right and left side contact forces. The force calculated from this expression is compared to the breakaway value given by $f_{\max} = \mu_s F_N$. For a time-varying normal force, this breakaway value is computed each time step and the current value used in the comparison. If f is greater than $f_{\max}$, then the block is breaking away and $f_{\max}$ is applied to the block for this time step; otherwise, the computed f from Eq. (6) is used. These steps are illustrated in the flow chart given in Fig. 8. This model works well and is believed to give an accurate representation of the physical process. It has been extended and used in a two-dimensional array core representation where frictional effects on top and bottom surfaces are represented in a similar manner (Appendix A).

B. The Material Hysteresis Loss Model

The use of a material hysteresis loss model is an attempt to represent the effect sometimes expressed by a coefficient of restitution, i.e., energy is lost, but momentum is conserved during impact. It is also based on the hysteresis loop obtained in the loading-unloading stress-strain test for nonlinear materials such as graphite.

Fig. 3. Flow chart illustrating the value of friction to be used for a given time step.

Figure 9 illustrates the material hysteresis loss model used in the program, with the shaded area of the hysteresis loop representing mechanical energy lost during the loading-unloading impact sequence. The sign of the relative velocity of the two impacting masses is examined to determine if a block is being loaded or unloaded. If loading is occurring, the force acting on the block is given by

Fig. 9. Hysteretic internal energy loss model under study.

the upper curve; whereas if unloading is occurring the force is given by the lower curve. The slope of the lower curve is determined by specifying the percentage energy loss during the impact sequence. Thus,

$$D = K_B - \frac{\%E \text{ loss}}{100} K_B,$$

where

D = slope of the unloading curve,
 E = elastically stored energy, and
 K_B = block stiffness.

To determine the effectiveness of the program in utilizing the model, the forces in each time step were plotted versus relative displacement, for a collision of the two interior blocks (Fig. 1). This plot is shown in Fig. 10. If we compare Figs. 9 and 10, we see that the program represents the theoretical model very well.

Fig. 10. Hysteresis model.

A more realistic nonlinear description of the hysteresis loss can be used if desired, with a mathematical description of the loading and unloading curves. The advantage of the method used here is its simplicity and ease of implementation.

IV. CAPABILITIES AND LIMITATIONS OF FYSMOD

A. Uses of This Code to Date

The FYSMOD code allows the user to vary parameters characterizing a seismic problem. This feature has been used to make studies in the following categories,^{8,9,10}

1. A computer model study was performed to check scaling laws. The use of a computer model allows a choice of material properties, including ones for which no real material can be found.
2. A parameter study was performed where model properties and the excitation function were varied to gain greater understanding of the physical system.
3. System response to various mixed frequency inputs was studied.
4. System response to random excitation was examined.
5. System damping effects were studied.

B. Execution Time

Execution time of FYSMOD varies with several factors as shown in Figs. 11-14. In these figures the following abbreviations are used:

Fig. 11. Effect of total time-history on computation time.

Fig. 12. Effect of number of blocks on computation time.

DT = integration time interval,

CF = convergence factor,

N = number of blocks, and

TMAX = duration of time-history.

Execution time is proportional to the total time-history to be obtained, Fig. 11, and to the number of blocks,

Fig. 13. Effect of time increment on computation time.

Fig. 14. Effect of convergence factor on computation time.

Fig. 12. The time step (DT) chosen has a major effect. For small DT, execution time is approximately inversely proportional to DT. As DT is increased, the number of iterations needed to achieve convergence increases, and the time saving decreases, Fig. 13. Execution time is nearly independent of convergence factor for a sufficiently small time step, Fig. 14. Choice of DT and convergence factor are also governed by accuracy considerations, as discussed in Sec. II. These execution time figures refer to the one-dimensional version.

C. Driving Options

FYSMOD program can perform dynamic analysis on blocks driven by various input motions. These driving options include the following.

1. Earthquake time-history: T , X , $\dot{X}$, and $\ddot{X}$ are read from a file. If there are not sufficient points per unit time, a linear interpolation between adjacent points is performed. The earthquake time-history may be generated by another program run as part of the same job, or from another source. See Appendix C for a brief discussion of the generation of earthquakes.^{11,12} The format under which the time-history is read is $[3(x, 8E10.4, /)]$. Subroutines DRIVE and INTERP are used by this option.
2. A simple sinusoidal input of the form

$$X = A (1 - \cos \omega t),$$

$$\dot{X} = A \omega \sin \omega t,$$

$$\ddot{X} = A \omega^2 \cos \omega t.$$

3. An amplitude modulated wave of the form

$$X = A \cos(\omega_s t) - A \cos(\omega_D t),$$

$$\dot{X} = -A \omega_s \sin \omega_s t + A \omega_D \sin \omega_D t,$$

$$\ddot{X} = -A \omega_s^2 \cos \omega_s t + A \omega_D^2 \cos \omega_D t,$$

where

$$\omega_s = \omega_1 + \omega_2 \text{ and } \omega_D = \omega_1 - \omega_2.$$

4. A summation of from 2 to 10 waves each similar to the second option above in form. That is,

$$X = \sum_{i=1}^N A_i [1 - \cos(\omega_i t)],$$

$$\dot{X} = \sum_{i=1}^N A_i \omega_i \sin \omega_i t,$$

$$\ddot{X} = \sum_{i=1}^N A_i \omega_i^2 \cos \omega_i t,$$

with

$$2 \leq N \leq 10.$$

5. A sinusoidal input of the form

$$X = A \sin \omega t,$$

$$\dot{X} = A \omega \cos \omega t,$$

$$\ddot{X} = -A \omega^2 \sin \omega t.$$

Note that in this option, the velocity at time zero does not equal zero. This physically unrealistic situation provided the motivation for using the function $(1 - \cos \omega t)$ in the other options.

6. A narrow-band frequency modulated wave of the form

$$X = A \cos \omega t - B \cos \omega_D t$$

$$+ B \cos \omega_s t - A,$$

$$\dot{X} = A \omega \sin \omega t + B \omega_D \sin \omega_D t$$

$$- B \omega_s \sin \omega_s t,$$

$$\ddot{X} = -A \omega^2 \cos \omega t + B \omega_D^2 \cos \omega_D t$$

$$- B \omega_s^2 \cos \omega_s t,$$

where

$$\omega_D = \omega - \omega_m \text{ and } \omega_s = \omega + \omega_m.$$

Other input options using up to 10 frequencies and 10 amplitudes may be readily added to the program by modifying the subroutine DRIVER.

V. INPUT INSTRUCTIONS

The variables over which the user has direct control can be divided into three categories:

- 1) those defining the physical model (number of blocks, weight of each, gap size, stiffness),
- 2) those defining the model excitation (frequencies, amplitudes, excitation function, time duration of run, time when excitation ceases), and 3) those

defining the calculation details (differential time increment, convergence factor). With the exception of earthquake time-histories discussed later, the

data are read from cards in the input file. Included in the data deck are several data comment cards as an aid to the user. The data cards are as follows.

Card	Column	Variable	Format	Comments and Definitions
1	1-80	ITITLE	8A10	title Card
2	1-5	IB	I5	last block of forcing box (normally 6)
	6-10	IFIRST	I5	first block
	11-15	I LAST	I5	last block
3A	1-80	data comment -		recommended wording-"next card contains DT, GF, TMAX 3F10.0"
3B	1-10	DT	F10.0	differential time increment for integration, seconds.
	11-20	CF	F10.0	convergence factor
	21-30	TMAX	F10.0	total time of run, seconds
4A	1-80	data comment -		"next card contains GMUS, GMIJK W, PHL 4F10."
4B	1-10	GMUS	F10.0	static coefficient of friction
	11-20	GMIJK	F10.0	kinetic coefficient of friction
	21-30	W	F10.0	weight of one block
	31-40	PHL	F10.0	percent hysteretic loss in collision (See Sec. III. B.)
5A	1-80	data comment -		"next card contains (IAOPT, NA, TSTP, (2I5, F10.0) then NA/2 cards of 2(AI, FREQ)
5B	1-5	IAOPT	I5	IAOPT defines the driving function, (See Sec. IV.D for details.)

1. Earthquake, read time-history from FSET9
NA=0
2. Sinusoidal input of the form $X = A - \cos(\omega t)$
NA=1
3. Amplitude modulated. One amplitude and two frequencies are required. The two frequencies are the sum and the difference shown in IVC3.
4. Summation of from two to ten waves each similar to that in option 2, above.
5. Sinusoidal input of the form $X = A \sin \omega t$
NA=1
6. Narrow-band frequency modulated. Two amplitudes and two frequencies are required. The two frequencies are the carrier frequency and the modulating frequency.

<u>Card</u>	<u>Column</u>	<u>Variable</u>	<u>Format</u>	<u>Comments and Definitions</u>
	6-10	NA	15	number of frequencies and amplitudes given.
	11-20	TSTP	F10.0	time that excitation is to cease. If TSTP = 0, excitation will continue to end of run.
5C	1-10	AI(I)	F10.0	amplitudes and frequencies (Hertz) required for option selected on card 5B. If IAOPT = 1 (earthquake input) card 5C is omitted. If IAOPT equals 4 and NA is greater than 2, then (NA+1)/2 cards 5C are required. For all other options, one card 5C suffices. Note that frequencies are in Hertz. Circular frequencies are calculated internally.
	11-20	FREQI(I)	F10.0	
	21-30	AI(I+1)	F10.0	
	31-40	FREQI(I+1)	F10.0	
6A	1-80	data comment -		"next card contains C, SK, F10.0, E12.4"
6B	1-10	C	F10.0	clearance gap between blocks.
	11-22	SK	E12.4	loading block stiffness.

VI. OUTPUT OPTIONS

The output from FYSMOD is available in printed and plotted forms. The plotted output is installation dependent, and it presently utilizes LASL's standardized plot package for the SC 4020.

The printed output in the normal output file consists of an echo of the input data, plus various peak values detected by the program. Included are the following:

- 1) the maximum acceleration experienced by any block,
- 2) the maximum positive and negative displacement and acceleration experienced by each block and the times that these occur,
- 3) the number of collisions experienced by each block,
- 4) the peak contact force information containing:
 - a) the peak contact force experienced by the left side of each block,
 - b) the time that the peak occurred,
 - c) the contact force on the right side of the block at that time, and
 - d) the acceleration of the block at that time.

Also available for printing are the files FSET8 and FSET11. Both contain as their first record, the total time of the run (TMAX) and the title of the run (ITITLE), in the format (F10.6, 8A10). This first line is written by the subroutine INPT.

Following this line, FSET8 gives information on the acceleration pulses experienced by each block, while FSET11 reports contact forces. Both these files may readily be used as input to plotting routines.

In FSET8 each record consists of the block number, the pulse number, the peak acceleration reached in that pulse (g's), and the time when that peak is reached. The format is (7H BLOCK, I4, 7H PULSE, I4, E15.6, 6H^^^AT, F10.6). This record is written by the subroutine CALC.

In FSET11, each record after the first consists of the block number, the pulse number, the peak contact force experienced by the left side of the block during that pulse, the time when that peak occurs, the contact force on the right side of the block at that time, and the acceleration (g's) of the block at that time. Note that block 4 is the right hand wall of the boundary. The format of this record is

```
(7H BLOCK^, I4, 7H PULSE^, I4, E15.6, 6H^^^AT,
F10.6, X, 7HF(I,J)= , E15.6, 5H ACC= , E15.6).
```

This record is written by the subroutine CF11.

A plot program that utilizes FSET8 and FSET11 to generate film plots of the acceleration and contact force time-histories for each block, is described in Appendix C.

Further plotted output from FYSMOD consists of time-history plots of the driving functions used.

REFERENCES

1. "GASSAR 6, General Atomic Standard Safety Analysis Report," General Atomic Company report GA-A13200 (no date).
2. A. J. Neylan and W. Gorholt, "Design Development of the HTGR Core and its Support Structure: Seismic Considerations," Nucl. Eng. Des. 29, 231-242 (1974).
3. T. H. Lee, "Nonlinear Dynamic Analysis of a Stacked Fuel Column Subjected to Boundary Motion," Nucl. Eng. Des. 32, 337-350 (1975).
4. L. Lasker, ed., P. Bezler, J. Curreri, B. Koplik, and M. Reich, "OSCIL: One-Dimensional Spring-Mass System Simulator for Seismic Analysis of High Temperature Gas Cooled Reactor Core," Brookhaven National Laboratory report BNL-21023 (January 1975).
5. J. G. Bennett and R. C. Dove, "Proposal for Analysis of HTGR Core Response to Seismic Input," Los Alamos Scientific Laboratory report LA-5821-MS (January 1975).
6. Robert E. Nickell, "A Survey of Direct Integration Methods in Structural Dynamics," Brown University, Providence, R. I. NTIS Report No. AD-743984 (April 1972).
7. J. F. McNamara, "Solution Schemes for Problems of Nonlinear Structural Dynamics," presented at the Pressure Vessels and Piping Conference with Nuclear Engineering and Materials Division, Miami Beach, Fla., of the American Society of Mechanical Engineers. Paper No. 74-PVP-30 (June 24-28, 1974).
8. N. M. Newmark, "A Method of Computation for Structural Dynamics," J. of the Eng. Mech. Div., ASCE, 88, EM3 (1959).
9. W. L. Kirk, "HTGR Safety Research Program, April-June 1975," Los Alamos Scientific Laboratory report LA-6054-PR (September 1975).
10. W. L. Kirk, "HTGR Safety Research Program, July-September 1975," Los Alamos Scientific Laboratory report LA-6161-PR (December 1975).
11. N. M. Newmark, J. A. Blume, and K. K. Kapur, "Seismic Design Spectra for Nuclear Power Plants," Journal of the Power Division, ASCE, 99, P02, 287-303 (1973).
12. R. H. Scanlan and K. Sachs, "Earthquake Time Histories and Response Spectra," Journal of the Engineering Mechanics Division, ASCE, 100, EM-4 635-655 (1974).

APPENDIX A

EXTENSION OF THE MODEL TO TWO DIMENSIONS

The extension of the one-dimensional model to the two-dimensional model involves the addition of pin-slot forces, upper and lower frictional forces, and upper and lower normal forces. Figure A-1 shows the free body diagram of a typical block in the two-dimensional model. Figure A-2 shows the two-dimensional model.

Fig. A-1. Free body diagram illustrating all effects in the two-dimensional model.

Fig. A-2. Two-Dimensional Model.

To extend the friction model, it is necessary to examine the relative horizontal velocities between each block and the two blocks vertically adjacent to it. The four cases that result are handled appropriately: 1) if no-slip top and bottom, compute the frictional force from dynamic equilibrium considerations, 2) and 3) slip top (bottom), no-slip bottom (top), apply the kinetic value $\mu_k F_N$ to the slipping surface and compute the value on the no-slip surface from equilibrium, and 4) slip both surfaces, apply the appropriate kinetic value to each surface.

The extension to include pin-slot forces requires only that the relative displacements between upper and lower blocks be examined for pin-gap closure. Otherwise, the pin-slot forces are handled exactly as side contact forces, i.e., through the specification of the appropriate

component element stiffness. If the pin and slot stiffnesses are different, they must be combined as springs in series to give an equivalent single value.

The extension to include upper and lower normal forces is again through the appropriate component element stiffness and inclusion of another coordinate. Otherwise, the vertical direction is handled the same as the horizontal. No vertical frictional effects are included in the model since horizontal contact times are extremely short. Vertical hysteresis is included. This means that the coupling between the two coordinates comes only through the proportional friction terms, i.e., the horizontal motion is influenced by the vertical, but not vice versa. The two-dimensional program input is similar to that for the one-dimensional program. Those input cards that differ are described below.

<u>Card</u>	<u>Column</u>	<u>Variable</u>	<u>Format</u>	<u>Comments and Definitions</u>
4A	1-80	data comment -		"next card GMUS, GMUK, W, PHL, PHLV 5F10.0."
4B	1-10	GMUS	F10.0	static coefficient of friction.
	11-20	GMUK	F10.0	kinetic coefficient of friction.
	21-30	W	F10.0	weight of one block
	31-40	PHL	F10.0	percent hysteretic loss in horizontal collision
	41-50	PHLV	F10.0	percent hysteretic loss in vertical collision
5A	1-80	data comment -		"next IAOPT, NA, TSTP, INFILE, INFILE2. 2I5, F10.0, 2I5."
5B	1-5	IAOPT	I5	IAOPT defines the driving function. For $2 \leq \text{IAOPT} \leq 6$, the amplitudes and frequencies may be different in the two directions, but the number of them and the form of the function must be the same. These options define the same functions as those in the one-dimensional version. <ul style="list-style-type: none"> 1. Horizontal earthquake read from INFILE, no vertical input. 2-6. Functions as defined for one-dimensional model, applied in both directions. 7. Vertical earthquake read from INFILE, no horizontal input. 8. Vertical earthquake read from INFILE, and horizontal sinusoidal input as defined by options.

Card	Column	Variable	Format	Comments and Definitions
				9. Horizontal earthquake read from INFILE, vertical earthquake read from INFILE2.
	6-10	NA	I5	number of frequencies and amplitudes given.
	11-20	TSTP	F10.0	time that excitation is to cease. If TSTP = 0, excitation will continue to end of run.
	21-25	INFILE	I5	designation of first earthquake input file.
	26-30	INFILE2	I5	designation of second earthquake input file.
5C	1-10	AI(I)	F10.0	amplitudes and frequencies (Hertz) required for option selected on card 5B. If IAOPT = 1 or 7, card 5C is omitted. Otherwise, the number of cards 5C is the same as in the one-dimensional version. For IAOPT = 8, card 5C is used for vertical inputs, otherwise it specifies horizontal inputs.
	11-20	FREQI(I)	F10.0	
	21-30	AI(I+1)	F10.0	
	31-40	FREQI(I+1)	F10.0	
5D	1-10	AY(I)	F10.0	vertical amplitudes and frequencies (Hertz) required for option selected on card 5B. Used only if $2 \leq \text{IAOPT} \leq 6$.
	11-20	FREQIY(I)	F10.0	
	21-30	AY(I+1)	F10.0	
	31-40	FREQIY(I+1)	F10.0	
7A	1-80	data comment -		"next card CP, SKV, SKPU, SKPL 4F10.0."
7B	1-10	CP	F10.0	pin gap vertical stiffness stiffness of pin stiffness of slot (must equal SKPU).
	11-20	SKV	F10.0	
	21-30	SKPU	F10.0	
	31-40	SKPL	F10.0	

The output for the two-dimensional program contains horizontal information similar to that given by the one-dimensional version, as well as pin contact forces, vertical accelerations and

contact forces, and block separations in the vertical direction. In addition FSET16, FSET22, and FSET33 are generated for plotting vertical accelerations, vertical contact forces, and pin forces, respectively.

APPENDIX B

LASL Identification: LP-0660

FYSMOD SOURCE LISTING

```
PROGRAM FYSMOD (INP,OUT,FSET5=INP,FSET6=OUT,FILM,FSET7,FSET8,FSET9
1,FSET11,FSET1=FILM)
```

```

C
C  VARIABLE      DEFINITION
C  COMMON /NUM/
C      I          GENERALLY IDENTIFIES A BLOCK
C      J          GENERALLY IDENTIFIES THE BLOCK TO THE LEFT OF I
C      L          GENERALLY IDENTIFIES THE BLOCK TO THE RIGHT OF I
C  COMMON /CORD/
C      X(I)       DISPLACEMENT OF BLOCK I AT TIME T
C      XTP1(I)    DISPLACEMENT OF BLOCK I AT TIME T+DT
C  COMMON /ONEDS/
C      XOTPI(I)   VELOCITY (X-DOT) OF BLOCK I AT TIME T+DT
C      XD(I)      VELOCITY OF BLOCK I AT TIME T
```



```

C      COMMON /TWODS/
C      XDDTP1(I)  ACCELERATION OF BLOCK I AT TIME T+DT
C      XDD(I)     ACCELERATION OF BLOCK I AT TIME T
C      COMMON /PDES/
C      COMMON /MASS/
C      WOG(I)     WEIGHT OF BLOCK I DIVIDED BY GRAVITY
C      COMMON /FSTUF/
C      GMUS       STATIC COEFFICIENT OF FRICTION
C      GMUK       DYNAMIC COEFFICIENT OF FRICTION
C      W          WEIGHT OF ONE BLOCK
C      COMMON /STUFF/
C
C      C          CLEARANCE BETWEEN BLOCKS
C      SKL        LOADING STIFFNESS
C      SKU        UNLOADING STIFFNESS
C      COMMON /HOWBIG/
C      IFIRST     NUMBER OF FIRST BLOCK NOT PART OF BOUNDARY
C      ILAST      NUMBER OF LAST BLOCK
C      COMMON /FSS/
C      F(I,J)     FORCE BETWEEN BLOCK I AND BLOCK J
C      FR(I,J)    FRICTION FORCE BETWEEN BLOCK I AND BLOCK J
C      COMMON /INITC/
C      INFILE     FILE FROM WHICH EARTHQUAKE IS READ
C      KPL        COUNT OF PULSES
C      APLMX(I,1) PEAK ACCELERATION OF BLOCK I IN A PULSE
C      APLMX(I,2) TIME AT WHICH APLMX(I,1) OCCURED
C      TSTART     TIME AT WHICH TO START RECORDING ACCELERATION PULSES
C      DETECT     LEVEL ABOVE WHICH TO RECORD ACCELFRACTION PULSES
C      XMAX(I,1)  MAXIMUM DISPLACEMENT OF BLOCK I
C      XMAX(I,2)  TIME AT WHICH XMAX(I,1) OCCURED
C      XMIN(I,1)  MINIMUM (MAXIMUM NEGATIVE) DISPLACEMENT OF BLOCK I
C      XMIN(I,2)  TIME AT WHICH XMIN(I,1) OCCURED
C      AMAX(I,1)  MAXIMUM ACCELERATION OF BLOCK I
C      AMAX(I,2)  TIME AT WHICH AMAX(I,1) OCCURED
C      AMIN(I,1)  MINIMUM ACCELERATION OF BLOCK I
C      AMIN(I,2)  TIME AT WHICH AMIN(I,1)OCCURED
C      ITITLE     TITLE OF RUN
C      IT
C      ICOUNT
C      ACMAx      MAXIMUM ACCELERATION EXPERIENCED BY ANY BLOCK
C      NI
C      PI         3.14159...
C      BATA       FOR USE IN NEWMARK INTEGRATION (=0.25)
C      TMAX       TOTAL TIME DURATION OF RUN
C      JT         TOTAL NUMBER OF TIME STEPS
C
C      DT         DIFFERENTIAL TIME STEP
C      SK         STIFFNESS
C      T          TIME
C      COMMON /INITC1/
C      INOPT      FLAG TO INDICATE EARTHQUAKE INPUT
C      ITER       ITERATION COUNTER
C      CF         CONVERGENCE FACTOR
C      COMMON /VIB/
C      AI         DRIVING FUNCTION AMPLITUDE COMPONENT
C      FREQI      DRIVING FUNCTION FREQUENCY COMPONENT
C      OMEGAI     CIRCULAR FREQUENCY (2*PI*FREQI)
C      IAOPT      DRIVING FUNCTION OPTION FLAG
C      TSTP       TIME AT WHICH EXCITATION IS TO STOP
C      NA         NUMBER OF AMPLITUDES AND FREQUENCIES IN DRIVING FTN
C
C      IN GENERAL, VARIABLES NOT IDENTIFIED ABOVE BUT
C      PRESENT IN COMMON BLOCKS ARE NOT USED IN THE
C      1D VERSION, BUT WERE INCLUDED FOR EASE IN
C      EXPANSION TO HIGHER DIMENSIONS
C
C      INTEGER O
C      COMMON /NUM/ I,J,K,L,M,N,O
C      COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
C      COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
C      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
C      COMMON /PDES/ P11X,P11Y,P12X,P12Y,P13X,P13Y,P21X,P21Y,P22X,P22Y,P2
C      13X,P23Y

```

```

COMMON /MASS/ WOG(70),ERT(70)
COMMON /FSTUF/ GMUS,GMUK,W
COMMON /STUFF/ C,SKL,SKU
COMMON /HOWBIG/ IFIRST,ILAST
COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
COMMON /INITC/ INFILE,KPL,APLMX,TSTART,DETECT,XMAX,XMIN,AMAX,AMIN,
1ITITLE,IT,ICOUNT,ACMAX,NI,PI,BATA,A,A2,FREQ,FREQ2,OMEGA,OMEGA2,TMA
2X,JT,DT,SK,T
COMMON /INITC1/ INOPT,ITER,CF
COMMON /VIB/ AI(10),FREQI(10),OMEGAI(10),IAOPT,TSTP,NA
DIMENSION CFMXB(22,4)
DIMENSION XMAX(22,2), XMIN(22,2), AMAX(22,2), AMIN(22,2)
DIMENSION KPL(22), APLMX(22,2), XDDSV(22)
DIMENSION ITITLE(8)
DIMENSION ABOUN(2)

```

C

```

IBUG=1
IBUG=0
IF (IBUG.EQ.1) WRITE (6,10)T
CALL INIT
IF (IBUG.EQ.1) WRITE (6,20)T
CALL INPT
IF (IBUG.EQ.1) WRITE (6,30)T
CALL CALC
IF (IBUG.EQ.1) WRITE (6,40)T
CALL OTPT
IF (IBUG.EQ.1) WRITE (6,50)T
STOP

```

C

```

10 FORMAT (* CALLING INIT,T=*,E15.6)
20 FORMAT (* CALLING INPT,T=*,E15.6)
30 FORMAT (* CALLING CALC,T=*,E15.6)
40 FORMAT (* CALLING OTPT,T=*,E15.6)
50 FORMAT (* CALLING STOP,T=*,E15.6)
END
SUBROUTINE INPT

```

C

C THIS ROUTINE READS AND PRINTS THE INPUT DATA, AND TRANSFORMS
C FREQUENCIES TO CIRCULAR FREQUENCIES.

C

C THIS ROUTINE IS CALLED BY THE MAIN PROGRAM

```

INTEGER O
COMMON /NUM/ I,J,K,L,M,N,O
COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
COMMON /TWOOS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
COMMON /PDES/ P11X,P11Y,P12X,P12Y,P13X,P13Y,P21X,P21Y,P22X,P22Y,P2
13X,P23Y
COMMON /MASS/ WOG(70),ERT(70)
COMMON /FSTUF/ GMUS,GMUK,W
COMMON /STUFF/ C,SKL,SKU
COMMON /HOWBIG/ IFIRST,ILAST
COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
COMMON /INITC/ INFILE,KPL,APLMX,TSTART,DETECT,XMAX,XMIN,AMAX,AMIN,
1ITITLE,IT,ICOUNT,ACMAX,NI,PI,BATA,A,A2,FREQ,FREQ2,OMEGA,OMEGA2,TMA
2X,JT,DT,SK,T
COMMON /VIB/ AI(10),FREQI(10),OMEGAI(10),IAOPT,TSTP,NA
COMMON /INITC1/ INOPT,ITER,CF
DIMENSION CFMXB(22,4)
DIMENSION XMAX(22,2), XMIN(22,2), AMAX(22,2), AMIN(22,2)
DIMENSION KPL(22), APLMX(22,2), XDDSV(22)
DIMENSION ITITLE(8)
DIMENSION ABOUN(2)
NAMELIST /BUG/ A,A2,OMEGA,OMEGA2,T,INOPT,INFILE,IFIRST,ILAST,X,XD,
1XDD,ITER,NI,BATA,DT,CF,TSTART,DETECT,ISUB
PI=ACOS(-1.)
ISUB=2
READ (5,170)(ITITLE(I),I=1,8)
WRITE (6,180)(ITITLE(I),I=1,8)
READ (5,160)I0,IFIRST,ILAST
READ (5,150)DT,CF,TMAX

```

```

      JT=TMAX/(100.*DT)
      READ (5,150)GMUS,GMUK,W,PHL
      WRITE (6,50)GMUS,GMUK
      WRITE (6,130)W
      WRITE (6,140)IB,IFIRST,ILAST,DT,CF,TMAX
      READ (5,60)IAOPT,NA,TSTP
      IF (TSTP.LE.0) TSTP=1.E6
C      IAOPT INDICATES THE KIND OF DRIVING FUNCTION TO BE USED BY THE
C      PROGRAM AS FOLLOWS. NA GIVES NUMBER OF ACCELERATIONS AND
C      FREQS TO BE READ, 2 SETS PER CARD, TSTP IS TIME TO STOP DRIVING
      IAOPT
      0      NOT USED
      1      EARTHQUAKE
      2      2 FREQUENCIES SUPERIMPOSED
      3      AMPLITUDE MODULATED,B,FDIFFERENCE,FSUM USED
      4      MORE THAN TWO SUPERIMPOSED
      5      1 STARTING NOT AT ZERO
      6      FREQUENCY MODULATED
      7-     FOR EXPANSION
      IF (IAOPT.NE.1) GO TO 10
      INOPT=1
      WRITE (6,70)
      GO TO 30
10  CONTINUE
      N=NA/2+.6
      DO 20 I=1,N
      J=2*I-1
      READ (5,80)AI(J),FREQI(J),AI(J+1),FREQI(J+1)
      OMEGAI(J)=2.*PI*FREQI(J)
      OMEGAI(J+1)=2.*PI*FREQI(J+1)
20  CONTINUE
      WRITE (6,90)IAOPT,(AI(I),OMEGAI(I),FREQI(I),I=1,NA)
30  CONTINUE
      READ (5,120)C,SK
      WRITE (6,110)C,SK
      SKL=SK
      SKU=SK*(1.-0.01*PHL)
      WRITE (6,100)SKL,SKU,PHL
      DO 40 I=IFIRST,ILAST
40  WOG(I)=W/386.4
      WRITE (11,190)TMAX,ITITLE
      WRITE (8,190)TMAX,ITITLE
      RETURN
C
50  FORMAT (1X,*STATIC FRICTION COEFFICIENT= *,E12.4//1X,*DYNAMIC FRICT
      ION COEFFICIENT = *,E12.4/)
60  FORMAT (/215,F10.0)
70  FORMAT (* EARTHQUAKE DRIVEN*)
80  FORMAT (8F10.0)
90  FORMAT (1X,*DRIVING OPTION = *,I5/10X,*AMPLITUDE*,5X,*FREQUENCY*,/(
      110X,F10.6,3X,E12.4,1X,*(*,E12.4,* HZ)*//))
100 FORMAT (1X,*LOADING BLOCK STIFFNESS =*,E12.5,* LB/INCH*//1X,*UNLOA
      DING BLOCK STIFFNESS *,E12.5,* LB/INCH*//1X,*PERCENTAGE HYSTERETIC
      2 ENERGY LOSS IN IMPACT*,E12.5/)
110 FORMAT (1X,*CLEARANCE GAP SIZE =*,E12.4,*INCHES*//1X,*BLOCK STIFFN
      1ESS (LINEAR) =*,F12.4,*LB/INCH*//)
120 FORMAT (/F10.0,E12.4)
130 FORMAT (1X,*WEIGHT OF EACH BLOCK*,E12.4,* LBS*//)
140 FORMAT (1X,*LAST BOUNDARY NO. = *,I5//1X,* FIRST AND LAST MASS NUM
      1BERS *,2I10//1X,*DT = *,F12.4//1X,*CONVERGENCE FACTOR *,E12.4//1X,
      2*MAXIMUM TIME HISTORY *,F12.4,* SECONDS*//)
150 FORMAT (/8F10.0)
160 FORMAT (3I5)
170 FORMAT (8A10)
180 FORMAT (/1X,10(2H* ),5X,,8A10,///)
190 FORMAT (F10.6,8A10)
      END
      SUBROUTINE DRIVER(X,XD,XDD,T,IERR,INFILL)
C
C      THIS ROUTINE IS CALLED BY CALC. IT PROVIDES CALC WITH THE DRIVING

```

```

C      DISPLACEMENT, VELOCITY, AND ACCELERATION AT EACH TIME STEP.
C      IF THE DRIVING FUNCTION IS AN EARTHQUAKE, DRIVER CALLS DRIVE TO
C      READ AND/OR INTERPOLATE THE EARTHQUAKE DATA
C
      COMMON /VIB/ AI(10),FREQ1(10),OMEGAI(10),IAOPT,TSTP,NA
      GO TO (10,20,40,20,50,60,70,70,70), IAOPT
10    CONTINUE
      CALL DRIVE (T,X,XD,XDD,IFRR,INFILE)
      GO TO 80
20    CONTINUE
      X=0
      XD=0
      XDD=0
      DO 30 I=1,NA
      X=X+(AI(I)*(1.-COS(OMEGAI(I)*T)))
      XD=XD+(AI(I)*OMEGAI(I)*SIN(OMEGAI(I)*T))
      XDD=XDD+(AI(I)*OMEGAI(I)*OMEGAI(I)*COS(OMEGAI(I)*T))
30    CONTINUE
      GO TO 80
40    CONTINUE

      X=AI(1)*(COS(OMEGAI(1)*T)-COS(OMEGAI(2)*T))
      XD=-1*AI(1)*OMEGAI(1)*SIN(OMEGAI(1)*T)+AI(1)*OMEGAI(2)*SIN(OMEGAI(
12)*T)
      XDD=-1*AI(1)*OMEGAI(1)*OMEGAI(1)*COS(OMEGAI(1)*T)+AI(1)*OMEGAI(2)*
10MEGAI(2)*COS(OMEGAI(2)*T)
      GO TO 80
50    CONTINUE
      X=AI(1)*SIN(OMEGAI(1)*T)
      XD=AI(1)*OMEGAI(1)*COS(OMEGAI(1)*T)
      XDD=-1*AI(1)*OMEGAI(1)*OMEGAI(1)*SIN(OMEGAI(1)*T)
      GO TO 80
60    CONTINUE
      FREQUENCY MODULATED -- NARROW BAND
      OMEGAD=OMEGAI(1)-OMEGAI(2)
      OMEGAS=OMEGAI(1)+OMEGAI(2)
      X=AI(1)*COS(OMEGAI(1)*T)-AI(2)*COS(OMEGAD*T)+AI(2)*COS(OMEGAS*T)-A
1I(1)
      XD=-1*AI(1)*OMEGAI(1)*SIN(OMEGAI(1)*T)+AI(2)*OMEGAD*SIN(OMEGAD*T)-
1AI(2)*OMEGAS*SIN(OMEGAS*T)
      XDD=-1*AI(1)*OMEGAI(1)*OMEGAI(1)*COS(OMEGAI(1)*T)+AI(2)*OMEGAD*OME
1GAD*COS(OMEGAD*T)-AI(2)*OMEGAS*OMEGAS*COS(OMEGAS*T)
      GO TO 80
70    CONTINUE
      WRITE (6,90)
      STOP 2
80    RETURN

C
90    FORMAT (* OPTIONS GREATER THAN 6 NOT IMPLEMENTED*)
      END
      SUBROUTINE INIT

C
C      THIS ROUTINE IS CALLED BY THE MAIN PROGRAM. IT INITIALIZES
C      VARIOUS VARIABLES
C
      INTEGER O
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
      COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
      COMMON /PDES/ P11X,P11Y,P12X,P12Y,P13X,P13Y,P21X,P21Y,P22X,P22Y,P2
13X,P23Y
      COMMON /MASS/ WUG(70),ERT(70)
      COMMON /FSTUF/ GMUS,GMUK,W
      COMMON /STUFF/ C,SKL,SKU
      COMMON /HOWBIG/ IFIRST,ILAST
      COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
      COMMON /INITC/ INFILE,KPL,APLMX,TSTART,DETCT,XMAX,XMIN,AMAX,AMIN,
1ITITLE,IT,ICOUNT,ACMAX,NI,PI,BATA,A,A2,FREQ,FREQ2,OMEGA,OMEGA2,TMA
2X,JT,DT,SK,T
      COMMON /INITC1/ INOPT,ITER,CF

```

```

COMMON /VIB/ AI(10),FREQI(10),OMEGAI(10),IAOPT,TSTP,NA
DIMENSION CFMXB(22,4)
DIMENSION XMAX(22,2), XMIN(22,2), AMAX(22,2), AMIN(22,2)
DIMENSION KPL(22), APLMX(22,2), XDDSV(22)
DIMENSION ITITLE(8)
DIMENSION ABOUN(2)
NAMELIST /BUG/ A,A2,OMEGA,OMEGA2,T,INOPT,INFILE,IFIRST,ILAST,X,XD,
1XDD,ITER,NI,BATA,DT,CF,TSTART,DETECT,ISUB
INOPT=0
INFILE=9
DO 10 I=1,ILAST
KPL(I)=0
APLMX(I,1)=0.0
APLMX(I,2)=0.0
10 CONTINUE
TSTART=0.
DETECT=80.
DO 20 I=1,2
DO 20 J=1,ILAST
XMAX(J,I)=0
XMIN(J,I)=0
AMAX(J,I)=0
AMIN(J,I)=0
20 CONTINUE
C INITIALIZE MASSES AND OTHER VARIABLES
IT=0
ICOUNT=0
ACMAX=0.
NI=4
IDEBUG=0
IDEBUG=1
PI=ACOS(-1.)
BATA=.25
T=0.
RETURN
END
SUBROUTINE CALC
C
C THIS ROUTINE CONTAINS THE MAIN CALCULATION LOOP. IT IS CALLED
C BY THE MAIN PROGRAM.
C IT CALLS DRIVER, DUM, DEFORM, FORCE, ORACC, ACCEL, AND CF11.
C
INTEGER O
COMMON /NUM/ I,J,K,L,M,N,O
COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
COMMON /TWOES/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
COMMON /PDES/ P11X,P11Y,P12X,P12Y,P13X,P13Y,P21X,P21Y,P22X,P22Y,P2
13X,P23Y
COMMON /MASS/ WOG(70),FRT(70)
COMMON /FSTJF/ GMUS,GMUK,W
COMMON /STUFF/ C,SKL,SKU
COMMON /HOWBIG/ IFIRST,ILAST
COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
COMMON /INITC/ INFILE,KPL,APLMX,TSTART,DETECT,XMAX,XMIN,AMAX,AMIN,
1ITITLE,IT,ICOUNT,ACMAX,NI,PI,BATA,A,A2,FREQ,FREQ2,OMEGA,OMEGA2,TMA
2X,JT,DT,SK,T
COMMON /INITC1/ INOPT,ITER,CF
COMMON /VIB/ AI(10),FREQI(10),OMEGAI(10),IAOPT,TSTP,NA
COMMON /OUTC/ XTEMP(70,100),PTIME(110),PTEMP(110),ATEMP(22,100),TP
1L(1001),APL(1001),VPL(1001),XPL(1001),KT,CFMXB
DIMENSION CFMXB(22,4)
DIMENSION XMAX(22,2), XMIN(22,2), AMAX(22,2), AMIN(22,2)
DIMENSION KPL(22), APLMX(22,2), XDDSV(22)
DIMENSION XDDOLD(70), YDDOLD(70), ZDDOLD(70), QDDOLD(70), RDDOLD(7
10), SDDOLD(70)
DIMENSION XBOUN(4)
DIMENSION ITITLE(8)
DIMENSION ABOUN(2)
DIMENSION XDD1(16), XDD2(16)
NAMELIST /BUG/ A,A2,OMEGA,OMEGA2,T,INOPT,INFILE,IFIRST,ILAST,X,XD,

```

```

1XDD,ITER,NI,BATA,DT,CF,TSTART,DETECT,ISUB
1JT=IMAX/(1000*DT)
C   CALCULATE FIRST BLOCK

C   *****
C   CALL DRIVER (X(2),XD(1),XDD(1),T,IERR,INFILE)
C
X(4)=X(2)
X(1)=X(2)
TPL(1)=T
XPL(1)=X(2)
VPL(1)=XD(1)
APL(1)=XDD(1)/386.4
KT=2
JCOUNT=0
C   DISPLACEMENT BOUNDARY CONDITION HERE
DO 10 I=IFIRST,ILAST
CALL DUM (I,J,K,L,M,N,O)
C   COMPUTE RELATIVE DISPLACEMENTS FROM ABSOLUTE ONES
CALL DEFORM (X,Y,Z,Q,R,S)
C   COMPUTE ALL FORCES
CALL FORCE
C   COMPUTE INITIAL ACCELERATION
10 CALL ORACC
C   BEGIN FORWARD ITERATION
C   RETURN TO THIS POINT FROM ITERATION END
20 DO 30 I=IFIRST,ILAST
XDDTP1(I)=XDD(I)
30 CONTINUE
C   SAVE OLD ACCL AND INTEGRATE FORWARD
C   QUIESCIENT INIT CONDITIONS ASSUMED
40 DO 50 I=IFIRST,ILAST
ITER=ITER+1
IDEBUG=1
IF (ITER.EQ.NI.AND.IDEBUG.EQ.1) WRITE (6,270)ITER,T
IF (ITER.EQ.NI.AND.IDEBUG.EQ.1) NI=NI+4
IF (ITER.GT.1000) STOP 2
XDDOLD(I)=XDDTP1(I)
XDTP1(I)=XD(I)+DT*.5*(XDDTP1(I)+XDD(I))
XTP1(I)=X(I)+XD(I)*DT+(.5-BATA)*XDD(I)*DT*DT+BATA*XDDTP1(I)*DT*DT
50 CONTINUE
C   BASED ON NEW DISPLACEMENTS, CALCULATE CORRES. ACCEL. FROM EQUIL.
CALL DRIVER (XTP1(2),XDTP1(1),XDDTP1(1),T+DT,IERR,INFILE)
IF (T+DT.LT.TSTP) GO TO 60
XDDTP1(1)=0
XDTP1(1)=0
XTP1(2)=X(2)
60 CONTINUE
U=T+DT
TGET=1.F6
IF (T.GT.TGET) WRITE (6,280)U,XTP1(2),XDTP1(1),XDDTP1(1)
C
XTP1(4)=XTP1(2)
XTP1(1)=XTP1(2)
DO 70 I=IFIRST,ILAST
CALL DUM (I,J,K,L,M,N,O)
CALL DEFORM (XTP1,YTP1,ZTP1,QTP1,RTP1,STP1)
CALL FORCE
CALL ACCEL
70 CONTINUE
C   CHECK FOR CONVERGENCE
NBLOCK=ILAST-IFIRST+1
DO 80 I5=IFIRST,ILAST
I1=I5-6
IF (ITER.EQ.11*NBLOCK) XDD1(I1)=XDDTP1(I5)
IF (ITER.EQ.12*NBLOCK) XDD2(I1)=XDDTP1(I5)

```

```

80 CONTINUE
  IF (ITER.EQ.12*NBLOCK) GO TO 90
  GO TO 110
90 DO 100 IJ=IFIRST,ILAST
  II=IJ+6
100 XDDTP1(IJ)=.5*(XDD1(II)+XDD2(II))
  NI=4
  GO TO 130
110 CONTINUE
  DO 120 I=IFIRST,ILAST
  IF (ABS(XDDTP1(I)-XDDOLD(I)).GT.ABS(CF*XDDOLD(I))) GO TO 40
120 CONTINUE
130 CONTINUE
C   UPDATE AND SAVE STEP
C   STEP TIME
  CALL CF11 (T,CFMXB)
  T=T+DT
  ITER=0
C   UPDATE
  DO 240 I=1,ILAST
  XDDSV(I)=XDD(I)
  X(I)=XTP1(I)
  XD(I)=XDTPI(I)
  XDD(I)=XDDTP1(I)
  ALLACC=1.
  ALLACC=0
  IF (ALLACC.EQ.0) GO TO 140
  IF (T.LE.TSTART) GO TO 140
  IF (I.LT.7) GO TO 140
  IF (XDD(I).GE.DETECT.OR.XDD(I).LE.-1*DETECT) WRITE (6,290)I,XD(I),
1XDD(I),T,X(I)
140 CONTINUE
  DETECT=1000
  IF (I.LT.7) GO TO 170
  IF (XDD(I).GT.DETECT.OR.XDD(I).LT.-1*DETECT) GO TO 150
  IF (XDDSV(I).LE.DETECT.AND.XDDSV(I).GE.-1*DETECT) GO TO 170
C   END OF A PULSE, OUTPUT AND REINITIALIZE
  APLMX(I,1)=APLMX(I,1)/386.4
  WRITE (6,300)I,KPL(I),APLMX(I,1),APLMX(I,2)
  APLMX(I,1)=0.0
  APLMX(I,2)=0.0
  GO TO 170
150 CONTINUE
C   IN A PULSE
C   IF IT IS A NEW ONE, INCREMENT THE COUNTER
  IF (XDDSV(I).LE.DETECT.AND.XDDSV(I).GE.-1*DETECT) KPL(I)=KPL(I)+1
  IF (ABS(XDD(I)).LE.ABS(APLMX(I,1))) GO TO 160
  APLMX(I,1)=XDD(I)
  APLMX(I,2)=T
160 CONTINUE
170 CONTINUE
  IF (X(I).LE.XMAX(I,1)) GO TO 180
  XMAX(I,1)=X(I)
  XMAX(I,2)=T
180 IF (X(I).GE.XMIN(I,1)) GO TO 190
  XMIN(I,1)=X(I)
  XMIN(I,2)=T
190 IF (XDD(I).LE.AMAX(I,1)) GO TO 200
  AMAX(I,1)=XDD(I)
  AMAX(I,2)=T
200 IF (XDD(I).GE.AMIN(I,1)) GO TO 210
  AMIN(I,1)=XDD(I)
  AMIN(I,2)=T
210 CONTINUE
  IF (ABS(XDD(I)).GT.ABS(ACMAX)) ACMAX=XDD(I)
  IF (ICOUNT-JT) 230,220,230
  SAVE HERE - SEE INIT
C   220 IF (I.EQ.1) IT=IT+1
  IF (I.LT.1.OR.IT.LT.1) WRITE (6,310)IT,IT,IT,1,1,I
  IF (I.GT.20.OR.IT.GT.100) WRITE (6,310)I,IT
  ATEMP(I,IT)=XDD(I)/386.4

```

```

      XTEMP(I,IT)=X(I)
      IF (I,GE,7) XTEMP(I,IT)=X(I)-X(2)
      PTIME(IT)=T
      IF (I,EQ,ILAST) ICOUNT=-1
230  IF (I,EQ,ILAST) ICOUNT=ICOUNT+1
240  CONTINUE
      IF (JCOUNT-IJT) 260,250,260
250  XPL(KT)=X(2)
      VPL(KT)=XDD(1)
      APL(KT)=XDD(1)/386.4
      TPL(KT)=T
      KT=KT+1
      JCOUNT=-1
260  JCOUNT=JCOUNT+1
      IF (T,LE,TMAX) GO TO 20
      RETURN
C
270  FORMAT (1X,*      ITER = *,I5,*      T= *,F14,5)
280  FORMAT (* IN CALC,T,XTP1(2),XDTP1(1),XDDTP1(1)=*,4E15,6)
290  FORMAT (* BLOCK,VEL,ACC,TIME,DIS = *,I5,4E17,8)
300  FORMAT (* BLOCK *,I4,* PULSE *,I4,E15,6,*      AT*,F10,6)
310  FORMAT (315)
      END
      SUBROUTINE OTPT
C      THIS ROUTINE HANDLES MOST OF THE OUTPUT FOR THE PROGRAM.
C      OTHER OUTPUT IS GENERATED IN CALC AND IN CF11.
C      INPT ECHOS THE INPUT DATA
      COMMON /HOWBIG/ IFIRST,ILAST
      COMMON /INITC/ INFILE,KPL,APLMX,TSTART,DETECT,XMAX,XMIN,AMAX,AMIN,
1ITITLE,IT,ICOUNT,ACMAX,NI,PI,BATA,A,A2,FREQ,FREQ2,OMEGA,OMEGA2,TMA
2X,JT,DT,SK,T
      COMMON /OUTC/ XTEMP(70,100),PTIME(110),PTEMP(110),ATEMP(22,100),TP
3L(1001),APL(1001),VPL(1001),XPL(1001),KT,CFMXB
      DIMENSION XBOUN(4)
      DIMENSION ITITLE(8)
      DIMENSION ABOUN(2)
      DIMENSION CFMXB(22,4)
      DIMENSION XMAX(22,2), XMIN(22,2), AMAX(22,2), AMIN(22,2)
      DIMENSION KPL(22), APLMX(22,2), XDDSV(22)
      NODISP=1
C      IF NODISP IS SET EQUAL TO ZERO HERE, A SET OF DISPLACEMENTS AND
C      ACCELERATIONS OF THE BLOCKS AND THE EQUALLY SPACED TIMES AT WHICH
C      THEY OCCURED WILL BE PRINTED AND PLOTTED.
      IF (NODISP,EQ,1) GO TO 90
      WRITE (6,120)IT,(XTEMP(2,11),I1=1,IT)
      WRITE (6,110)(PTIME(I1),I1=1,IT)
      DO 30 I=IFIRST,ILAST
      WRITE (6,120)I,(XTEMP(I,I1),I1=1,IT)
      DO 10 J=1,IT
10  PTEMP(J)=XTEMP(I,J)
      IF (I,NE,IFIRST) GO TO 20
      CALL MINV (PTIME,1,IT,IDUM,XBOUN(1))
      CALL MAXV (PTIME,1,IT,IDUM,XBOUN(2))
      CALL MINV (PTEMP,1,IT,IDUM,XBOUN(3))
      CALL MAXV (PTEMP,1,IT,IDUM,XBOUN(4))
20  CALL SPLOT (1,2,XBOUN(1),XBOUN(3),48,0)
      CALL PLOT (IT,PTIME,1,PTEMP,1,48,1)
30  CONTINUE
      DO 40 J=1,IT
40  PTEMP(J)=XTEMP(2,J)
      CALL SPLOT (1,IT,PTIME,PTEMP,48,1)
      DO 70 I=IFIRST,ILAST
      WRITE (6,130)I,(ATEMP(I,I1),I1=1,IT)
      DO 50 J=1,IT
50  PTEMP(J)=ATEMP(I,J)
      IF (I,NE,IFIRST) GO TO 60
      CALL MINV (PTEMP,1,IT,IDUM,XBOUN(3))
      CALL MAXV (PTEMP,1,IT,IDUM,XBOUN(4))
60  CALL SPLOT (1,2,XBOUN(1),XBOUN(3),48,0)
      CALL PLOT (IT,PTIME,1,PTEMP,1,48,1)

```



```

70 CONTINUE
DO 80 J=1,IT
PTMP(J)=ATEMP(1,J)
80 CONTINUE
I=1
WRITE (6,130)I,(ATEMP(1,I),I=1,IT)
ABOUN(1)=-2.
ABOUN(2)=2.
CALL SPLOT (1,2,XBOUN(1),ABOUN(1),48,0)
CALL PLOT (IT,PTIME,1,PTMP,1,48,1)
90 CONTINUE
CALL MINV (TPL,1,KT-1,IDUM,XBOUN(1))
CALL MAXV (TPL,1,KT-1,IDUM,XBOUN(2))
CALL MINV (XPL,1,KT-1,IDUM,XBOUN(3))
CALL MAXV (XPL,1,KT-1,IDUM,XBOUN(4))
CALL SPLOT (1,2,XBOUN(1),XBOUN(3),48,0)
WRITE (1,140)
CALL PLOT (KT-1,TPL,1,XPL,1,48,1)
CALL MINV (VPL,1,KT-1,IDUM,XBOUN(3))
CALL MAXV (VPL,1,KT-1,IDUM,XBOUN(4))
CALL SPLOT (1,2,XBOUN(1),XBOUN(3),48,0)
WRITE (1,150)
CALL PLOT (KT-1,TPL,1,VPL,1,48,1)
CALL MINV (APL,1,KT-1,IDUM,XBOUN(3))
CALL MAXV (APL,1,KT-1,IDUM,XBOUN(4))
XBOUN(3)=-1.
XBOUN(4)=1.
CALL SPLOT (1,2,XBOUN(1),XBOUN(3),48,0)
WRITE (1,160)
CALL PLOT (KT-1,TPL,1,APL,1,48,1)
ACMAX=ACMAX/386.4
WRITE (6,170)ACMAX
WRITE (6,180)
DO 100 I=7,ILAST
AMIN(I,1)=AMIN(I,1)/386.4
AMAX(I,1)=AMAX(I,1)/386.4
WRITE (6,190)I,(XMIN(I,J),J=1,2),(XMAX(I,J),J=1,2),(AMIN(I,J),J=1,
12),(AMAX(I,J),J=1,2)
100 CONTINUE
WRITE (6,200)(I,KPL(I),I=IFIRST,ILAST)
WRITE (6,210)(I,(CFMXB(I,J),J=1,4),I=1,ILAST)
RETURN
C
110 FORMAT (/1X,*TIMES*/(1X,10E12.5))
120 FORMAT (/1X,*MASS NO.*/15/(1X,10E12.4))
130 FORMAT (/1X,*MASS NO.*/15/(1X,10E12.4))
140 FORMAT (* DRIVING DISPLACEMENTS*)
150 FORMAT (* DRIVING VELOCITIES*)
160 FORMAT (* DRIVING ACCELERATIONS*)
170 FORMAT (/1X,*MAXIMUM RECORDED ACCELERATION **E12.4,3H G )
180 FORMAT (/4X,*I*,7X,*XMIN*,13X,*T*,14X,*XMAX*,13X,*T*,14X,*AMIN*,13
1X,*T*,14X,*AMAX*,13X,*T*)
190 FORMAT (X,I4,4(E15.6,E15.6,1X))
200 FORMAT (* BLOCK COLLISION*/(X,I4,2X,I10/))
210 FORMAT (* BLOCK*,4X,*PEAK C F*,9X,*TIME*,9X,*C F ON RIGHT*,9X,*ACC
1*/(X,I4,4E15.6))
END
SUBROUTINE CF11(T,CFMXB)
C THIS ROUTINE DETECTS AND RECORDS PEAK CONTACT FORCES BETWEEN
C BLOCKS.
INTEGER O
COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
COMMON /TWOOS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
COMMON /HOWBIG/ IFIRST,ILAST
DIMENSION CFMX(22,4), CF(22,2), CFSV(22), CFMXB(22,4), KPL(22)
DATA IFF/1/
DATA FTECT/1.F-6/
IBUG=1
IBUG=0

```

```

      IF (IFF.EQ.0) GO TO 20
      DO 10 I=1,ILAST
      CFMX(I,1)=0
      CFSV(I)=0
      KPL(I)=0
      CFMXB(I,1)=0
10    CONTINUE
      IFF=0
20    CONTINUE
      DO 30 I=JFIRST,ILAST
      IF (IBUG.EQ.1) WRITE (6,100)I
      CALL DUM (I,J,K,L,M,N,O)
      CFSV(I)=CF(I,1)
      CF(I,1)=F(I,L)
      CF(I,2)=F(I,J)
30    CONTINUE
      I=4
      CFSV(I)=CF(I,1)
      CF(I,1)=-1*F(I,4)
      CF(I,2)=0
      DO 90 I=1,ILAST
      IF (ABS(CF(I,1)).GT.FTECT) GO TO 50
      IF (ABS(CFSV(I)).LE.FTECT) GO TO 80
C    END OF A PULSE, OUTPUT AND REINITIALIZE
      CFMX(I,4)=CFMX(I,4)/386.4
      WRITE (11,110)I,KPL(I),CFMX(I,1),CFMX(I,2),CFMX(I,3),CFMX(I,4)
      DO 40 J=1,4
      CFMX(I,J)=0
40    CONTINUE
      GO TO 80
50    CONTINUE
C    IN A PULSE
C    IF IT IS A NEW ONE, INCREMENT THE COUNTER
      IF (CFSV(I).LE.FTECT.AND.CFSV(I).GE.-1*FTECT) KPL(I)=KPL(I)+1
      IF (ABS(CF(I,1)).LE.ABS(CFMXB(I,1))) GO TO 70
      CFMX(I,1)=CF(I,1)
      CFMX(I,2)=I
      CFMX(I,3)=CF(I,2)
      CFMX(I,4)=XDDTP1(I)
      IF (ABS(CF(I,1)).LE.ABS(CFMXB(I,1))) GO TO 60
      CFMXB(I,1)=CF(I,1)
      CFMXB(I,2)=I
      CFMXB(I,3)=CF(I,2)
      CFMXB(I,4)=XDDTP1(I)
60    CONTINUE
70    CONTINUE
80    CONTINUE
90    CONTINUE
      RETURN

C
100  FORMAT (* CALLING DUM WITH I=*,I10)
110  FORMAT (* BLOCK *,I4,* PULSE *,I4,E15.6,* AT*,F10.6,X,*F(1,J)=*
      1,E15.6,* ACC=*,E15.6)
      END
      SUBROUTINE DUM(I,J,K,L,M,N,O)
C    THIS ROUTINE IDENTIFIES THE BLOCKS ADJACENT TO A GIVEN BLOCK I
C    BLOCK L IS TO THE LEFT OF I, AND K IS TO ITS RIGHT
      INTEGER O
      COMMON /HOWBIG/ IFIRST,ILAST
      K=23
      N=23
      O=23
      M=1
      J=I+1
      L=I-1
      IF (I.EQ.IFIRST) L=2
      IF (I.EQ.ILAST) J=4
      RETURN
      END
      SUBROUTINE DEFORM(X,Y,Z,Q,R,S)
C    THIS ROUTINE CALCULATES DEFORMATIONS FROM DISPLACEMENTS

```

```

      INTEGER O
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /NDES/ DIJ,DIK,DIL,DIM,DIN,DIO
      DIMENSION X(70), Y(70), Z(70), Q(70), R(70), S(70)
      DIJ=X(J)-X(I)
      DIL=X(L)-X(I)
      RETURN
      END
      SUBROUTINE FORCE
C      THIS ROUTINE CALCULATES FORCES ON BLOCKS
      INTEGER O
      COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
      COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
      COMMON /NDES/ DIJ,DIK,DIL,DIM,DIN,DIO
      COMMON /STUFF/ C,SKL,SKU
      SK=SKL
      VRELIJ=XDTP1(I)-XDTP1(J)
      VRELIL=XDTP1(I)-XDTP1(L)
      IF (DIJ+C) 20,10,10
10  F(I,J)=0.
      GO TO 30
20  IF (VRELIJ.LT.0.) SK=SKU
      F(I,J)=SK*(DIJ+C)
30  IF (DIL-C) 40,40,50
40  F(I,L)=0.
      CALL CLOMB
      RETURN
50  IF (VRELIL.GT.0.) SK=SKU
      F(I,L)=SK*(DIL-C)
      CALL CLOMB
      RETURN
      END
      SUBROUTINE ORACC
C      THIS ROUTINE CALCULATES THE ORIGINAL ACCELERATIONS
      INTEGER O
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /MASS/ WOG(70),ERT(70)
      COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
      COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
      COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
      XDD(I)=1./WOG(I)*(F(I,J)+F(I,L)+FR(I,M))
      RETURN
      END
      SUBROUTINE ACCEL
C      THIS ROUTINE CALCULATES THE ACCELERATIONS USING F=MA
      INTEGER O
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /MASS/ WOG(70),ERT(70)
      COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)

      XDDTP1(I)=1./WOG(I)*(F(I,J)+F(I,L)+FR(I,M))
      RETURN
      END
      SUBROUTINE CLOMB
C      THIS ROUTINE CALCULATES THE FRICTION FORCES
      INTEGER O
      COMMON /CORD/ X(70),Y(70),XTP1(70),YTP1(70)
      COMMON /ONEDS/ XDTP1(70),YDTP1(70),XD(70),YD(70)
      COMMON /TWODS/ XDDTP1(70),YDDTP1(70),XDD(70),YDD(70)
      COMMON /FSS/ F(70,70),FR(70,70),PF1(70,70),PF2(70,70)
      COMMON /NUM/ I,J,K,L,M,N,O
      COMMON /MASS/ WOG(70),ERT(70)
      COMMON /FSTUF/ GMUS,GMUK,W
      VCHEK=1,E=3
      FKIN=GMUK*W
      FSTAT=GMUS*W

```

```

      VRELXB=XD(M)-XD(I)
      IF (ABS(VRELXB)-VCHEK) 10,10,40
10  FR(I,M)=-(F(I,J)+F(I,L)-WOG(I)*XDDTP1(M))
      IF (ABS(FR(I,M))-FSTAT) 30,30,20
20  FR(I,M)=SIGN(FSTAT,FR(I,M))
30  RETURN
40  FR(I,M)=SIGN(FKIN,VRELXB)
      RETURN
      END
      SUBROUTINE DRIVE(T,X,V,A,IERR,INFILE)
C     THIS ROUTINE PROVIDES EARTHQUAKE DRIVING DATA FOR THE PROGRAM,
C     IT READS AN EARTHQUAKE TIME HISTORY FROM THE FILE INFILE AS NEEDED
C     AND USES A LINEAR INTERPOLATION IN ROUTINE INTERP
C     IT IS CALLED BY DRIVER
      DIMENSION TI(6), XI(6), VI(6), AI(6)
      COMMON /JC/ JERK
      DATA IFIRST/1/
      TON1=.518
      TON1=1.E6
      TOF1=1.E6
      TON2=1.E6
      TOF2=1.E6
      JERK=0
      IF ((T.GE.TON1.AND.T.LE.TO1).OR.(T.GE.TON2.AND.T.LE.TO2)) JERK=1
      IF (JERK.EQ.1) WRITE (6,150)T
      IF (JERK.EQ.1) WRITE (6,160)(TI(I),XI(I),VI(I),AI(I),I=1,6)
      IBUG=1
      IBUG=0
      IF (IBUG.EQ.1) WRITE (6,150)T
      IERR=0
      IF (IFIRST.EQ.0) GO TO 80
      READ (INFILE,160)(TI(I),XI(I),VI(I),AI(I),I=1,6)
      IF (EOF,INFILE) 10,20
10  CONTINUE
      WRITE (6,170)
      IERR=4
      GO TO 140
20  CONTINUE
      IF (T.GE.TI(1)) GO TO 30
      T=TI(1)
      X=XI(1)
      V=VI(1)
      A=AI(1)
      WRITE (6,180)
      GO TO 140
30  IF (T.LE.TI(6)) GO TO 40
C     NEED LATER DATA, SAVE LATEST AND GO READ MORE
      TSV=TI(6)
      XSV=XI(6)
      VSV=VI(6)
      ASV=AI(6)
      IFIRST=0
      GO TO 90
40  CONTINUE
      DO 50 I=1,6
      J=I
      IF (T.NE.TI(J)) GO TO 50
      X=XI(J)
      V=VI(J)
      A=AI(J)
      IFIRST=0
      IF (JERK.EQ.1) WRITE (6,190)T,X,V,A,J
      GO TO 140
50  CONTINUE
C     T DOES NOT EXACTLY EQUAL ANY DATA POINT, FIND WHICH IT IS BETWEEN
      DO 60 I=1,5
      K=I
      IF (T.GT.TI(K).AND.T.LT.TI(K+1)) GO TO 70
60  CONTINUE
      WRITE (6,200)
      IERR=1

```

```

      GO TO 140
70 CONTINUE
  IF (JERK,EQ,1) WRITE (6,210)K,TI(K),TI(K+1)
C   T IS BETWEEN TI(K) AND TI(K+1)
  CALL INTERP (T,TI(K),TI(K+1),XI(K),XI(K+1),VI(K),VI(K+1),AI(K),AI(
1K+1),X,V,A)
  IFIRST=0
  GO TO 140
80 CONTINUE
  IF (T,LE,TI(6)) GO TO 120
  TSV=TI(6)
  XSV=XI(6)
  VSV=VI(6)
  ASV=AI(6)
90 READ (INFILE,160)(TI(I),XI(I),VI(I),AI(I),I=1,6)
  IF (EOF,INFILE) 100,110
100 CONTINUE
  WRITE (6,170)
  IERR=5
  GO TO 140
110 CONTINUE
  IF (JERK,EQ,1) WRITE (6,220)
  IF (JERK,EQ,1) WRITE (6,160)(TI(I),XI(I),VI(I),AI(I),I=1,6)
  GO TO 80
120 CONTINUE
  IF (T,GE,TI(1)) GO TO 40
  IF (T,GT,TSV) GO TO 130
  WRITE (6,230)
  IERR=2
  GO TO 140
130 CONTINUE
  IF (JERK,EQ,1) WRITE (6,240)T,TSV,TI(1)
  CALL INTERP (T,TSV,TI(1),XSV,XI(1),VSV,VI(1),ASV,AI(1),X,V,A)
  GO TO 140
140 CONTINUE
  A=A*386,4
  RETURN
C
150 FORMAT (* REACHED DRIVE,T=*,E15,6)
160 FORMAT (3(X,8E10,4,/))
170 FORMAT (* RAN OUT OF TIME HISTORY DATA*)
180 FORMAT (* STARTING TIME WAS LESS THAN START OF EARTHQUAKE*)
190 FORMAT (* FOUND T EXACTLY,T,X,V,A,J=*,4E15,6,I2)
200 FORMAT (* ERROR, T SHOULD BE IN RANGE AND WAS NOT*)
210 FORMAT (* CALLING INTERP WITH K,TI(K),TI(K+1)=*,I4,2F10,6)
220 FORMAT (* READ MORE DATA*)
230 FORMAT (* ERROR, T IS .LE,TSV*)
240 FORMAT (* CALLING INTERP WITH T,TSV,TI(1)=*,3F10,6)
  END
  SUBROUTINE INTERP(T,T1,T2,X1,X2,V1,V2,A1,A2,X,V,A)
C   THIS ROUTINE IS CALLED BY DRIVE TO INTERPOLATE AN EARTHQUAKE
  COMMON /JC/ JERK
  TF=(T-T1)/(T2-T1)
  X=(TF*(X2-X1))+X1
  V=(TF*(V2-V1))+V1
  A=(TF*(A2-A1))+A1
  IF (JERK,EQ,1) WRITE (6,10)T1,T,T2,TF,X1,X,X2,V1,V,V2,A1,A,A2
C
  RETURN
C
10 FORMAT (* T1,T,T2,TF=*,4E15,6,/* X1,X,X2=*,3E15,6,/* V1,V,V2=*,3E1
15,6,/*A1,A,A2=*,3E15,6)
  END

```

APPENDIX C SIMULATED EARTHQUAKES

Earthquakes may be thought of in either the time or frequency domain. Generally, earthquakes are compared and specified for use in safety analyses in the frequency domain,¹¹ i.e., their spectra are given. For the purposes of this analytical method, a time-history is required. This can be obtained in several ways. An actual earthquake accelerogram can be integrated to give velocities and displacements, or a computer program may be used to generate a simulated earthquake time-history. One such program is SIMEAR developed by M. Watabe and available through the National Information Service, Earthquake Engineering, University of California, Berkeley. It allows specifying a required spectrum by several methods. The intensity function is described in the form of a parabolic buildup, followed by a period of

approximately constant intensity, and an exponential decay to a cutoff time. The technique of random phase angles is then used to generate a simulated earthquake time-history, which may be scaled to a peak value specified by the user.

The program described above is only one of many such methods. Several references to other earthquake simulation methods are given in a paper by Scanlan and Sachs.¹² They describe an iterative method to generate a time-history with a desired spectrum. The method is based on the assumption that the zero-damping spectrum is similar to the Fourier spectrum of an earthquake. It can therefore be used as a first approximation in generating simulated earthquakes that envelope the response spectra for other damping ratios.

APPENDIX D PLOT8

The PLOT8 program was designed to display a time-history of accelerations and contact forces on each block in the FYSMOD model. The scale of each graph is specified by input, as are the data file and title. Data are read from files generated by Zeros are inserted in the array on either side of each data point to form peaks of very short duration simulating those that actually occur. The actual plotting is done by library routines local to the LASL system. An example of output is shown in Fig. D-1. A source listing follows.

Fig. D-1. Example of film output.

```

PROGRAM PLOT8 (INP,OUT,FILM,FSET8,FSET1=FILM,FSET5=INP,FSET6=OUT,F
1SET11,FSET22,FSF133,FSET16)
C
C   A ROUTINE TO PLOT ACCELERATIONS AND CONTACT FORCES FROM FYSMOD
C
  DIMENSION ITITLE(8)
  IDID=0
10 READ (5,40)INFILE,ITITLE
  IF (EOF,5) 30,20
20 CONTINUE
  WRITE (6,50)INFILE,ITITLE
  CALL PLOTSUB (INFILE,ITITLE)
  GO TO 10
30 STOP
C
40 FORMAT (I5,8A10)
50 FORMAT (* PLOTTING FROM *,I5,5X,8A10)
  END
  SUBROUTINE PLOTSUB(INFILE,ITITLE)
  DIMENSION ACC(20000), TIME(20000), ASCALE(2), TSCALE(2), ITITLE(8)
  DIMENSION JTITLE(8)
  COMMON IL(1000),KPLL(1000),AL(1000),TL(1000)
  IHUG=0
  REWIND INFILE
C
C   READ HALF-WIDTH OF PULSE, AND NUMBER OF FIRST AND LAST BLOCKS
C   READ INTERVAL BETWEEN PULSES TO BE PLOTTED
C
  READ(5,90) DT,IFIRST,ILAST,INT,INCR
  WRITE (6,80)INT
C
C   READ SCALE TO BE USED FOR ORDINATE
C
  READ (5,100)(ASCALE(I),I=1,2)
  IF (IHUG.EQ.1) WRITE (6,110)(ASCALE(I),I=1,2),DT,IFIRST,ILAST
C
C   READ TOTAL TIME AND TITLE AS GENERATED BY FYSMOD
C
  READ (INFILE,120)TMAX,JTITLE
10 CONTINUE
  DO 60 K=IFIRST,ILAST,INCR
    CALL SETPLT (1)
    J=2
20 CONTINUE
C
C   READ THE DATA
C
  READ(INFILE,130)(IL(LL),KPLL(LL),AL(LL),TL(LL),LL=1,INT)
  IHUG=0
  IF (EOF,INFILE) 50,30
30 IF (EOI(INFILE)) 50,40,50
40 CONTINUE
  I=IL(INT)
  A=AL(INT)
  T=TL(INT)
  IF (IHUG.EQ.1) WRITE (6,140)I,KPL,A,T,K
C
C   REJECT DATA IF IT IS FOR THE WRONG BLOCK OR IF IT IS NOT AT
C   THE SPECIFIED INTERVAL
C
  IF (I.NE.K) GO TO 20
C
C   PLACE ZEROS ON EITHER SIDE OF PULSE
C

```

```

      ACC(J)=A
      TIME(J)=T
      ACC(J+1)=0
      TIME(J+1)=T+DT
      ACC(J-1)=0
      TIME(J-1)=T-DT
      KOUNT=J+1
      J=J+3
      GO TO 20
50  CONTINUE
      IF (IBUG.EQ.1) WRITE (6,150)K,KOUNT
C    REACHED END OF INFO OR END OF FILE--PLOT THEN REWIND
      TSCALE(1)=3
      TSCALE(2)=TMAX
      CALL SPLOT (1,2,TSCALE,ASCALE,48,0)
      WRITE (1,160)JTITLE
      WRITE (1,170)ITITLE,K
      WRITE (6,160)JTITLE
      WRITE (6,170)ITITLE,K
      CALL PLOT (KOUNT,TIME,1,ACC,1,48,1)
      KOUNT=0
      REWIND INFILE
      READ (INFILE,120)TMAX,ITITLE
60  CONTINUE
      IF (INFILE.NE.11.OR.IDID.EQ.1) GO TO 70
C
C    PICK UP CONTACT FORCE ON RIGHT WALL
C
      IDID=1
      IFIRST=1
      ILAST=4
      GO TO 10
70  RETURN
C
80  FORMAT (* DATA PLOTTED EVERY *,I5,*POINTS*)
90  FORMAT(F10.6,4I5)
100 FORMAT (2E15.6)
110 FORMAT (* ASCALE,DT,IFIRST,ILAST=*,2F15.6,5X,F10.6,2I5)
120 FORMAT (F10.6,8A10)
130 FORMAT ((7X,I4,6X,I7,E15.6,4X,F10.6/))
140 FORMAT (* READ I,KPL,A,T=*,I4,X,I4,E15.6,X,F10.6,* WITH K=*,I4)
150 FORMAT (* REACHED END OF FILE,K,KOUNT=*,2I10)
160 FORMAT (X,8A10)
170 FORMAT (X,8A10,* BLOCK *,I5)
      END

```