BIBLIOGRAPHIC INFORMATION

PB93-216000

Report Nos: REPT-3.1(B)-1

Title: Summary on Pseudodynamic Testing.

Date: Feb 87

Authors: F. Seible, and H. L. LaRovere.

Performing Organization: United States-Japan Coordinated Program for Masonry Building Research.**California Univ., San Diego, La Jolla. Dept. of Applied Mechanics and Engineering Sciences.

Sponsoring Organization: *National Science Foundation, Washington, DC.

Grant Nos: NSF-ECE-85-52672

Supplementary Notes: See also PB84-119270 and PB84-190644. Prepared in cooperation with California Univ., San Diego, La Jolla. Dept. of Applied Mechanics and Engineering Sciences. Sponsored by National Science Foundation, Washington, DC.

NTIS Field/Group Codes: 89D

Price: PC A04/MF A01

Availability: Available from the National Technical Information Service, Springfield, VA. 22161

Number of Pages: 53p

Keywords: *Dynamic tests, *Computerized simulation, *Dynamic structural analysis, *Earthquake engineering, Test methods, Structural vibration, User manuals(Computer programs), Vibration damping, Mathematical models, Matrices(Mathematics), Computer programs, PSDYN Computer program.

Abstract: The analytical basis for interactive or on-line experimental testing under simulated dynamic loads, often referred to as Pseudodynamic Testing, is summarized in the report and implemented in a computer code PSDYN. The code is based on the CALSD series of structural analysis programs and incorporates a set of commands allowing not only the computation of the displacement response in a Pseudodynamic Method but also linear dynamic analysis of structures using step-by-step methods, solution of eigenvalue problems, and general matrix manipulations. PSDYN is written in FORTRAN77 with a free-field type of input and can be used either in batch or interactive mode. The program utilizes runtime libraries interconnected by a common database, which allows applications on large mainframes as well as on microcomputer systems. Examples described in the report have been run in a VAX/VMS environment and on the CRAY X-MP. Chapter 2 describes briefly the Pseudodynamic Test Method and summarizes the background necessary to utilize it. A complete user's manual for all the PSDYN commands is given in Chapter 3. In Chapter 4 two examples of a three Degrees of Freedom (DOF) structure subjected to a blast

Continued on next page

BIBLIOGRAPHIC INFORMATION

Continued... PB93-216000

loading and to the El Centro 1940 (NS) earthquake are shown.

日米

U.S. - JAPAN COORDINATED PROGRAM

FOR

MASONRY BUILDING RESEARCH

REPORT 3.1b-1

SUMMARY ON PSEUDO DYNAMIC TESTING

by

FRIEDER SEIBLE
HERIETTE L. LaROVERE

FEBRUARY 1987

supported by:

NATIONAL SCIENCE FOUNDATION

GRANT NO. ECE 85-52672

DEPARTMENT OF APPLIED MECHANICS & ENGINEERING SCIENCES
UNIVERSITY OF CALIFORNIA, SAN DIEGO, CA

PREFACE

This report was prepared in support of research Category 3.0 of the U.S. Coordinated Program for Masonry Building Research. The work was supported by NSF Grant ECE-8552672.

J. L. Noland

SUMMARY ON PSEUDODYNAMIC TESTING

by

FRIEDER SEIBLE
Assistant Professor of Structural Engineering

HENRIETTE L. LA ROVERE
Research Assistant

Department of Applied Mechanics and Engineering Sciences
University of California, San Diego
La Jolla, California

February, 1987

Table of Contents

1. SUMMARY

The analytical basis for interactive or on-line experimental testing under simulated dynamic loads, often referred to as Pseudodynamic Testing, is summarized in this report and implemented in a computer code PSDYN.

The code is based on the CALSD series of structural analysis programs [1] and incorporates a set of commands allowing not only the computation of the displacement response in a Pseudodynamic Method but also linear dynamic analysis of structures using step-by-step methods, solution of eigenvalue problems, and general matrix manipulations.

PSDYN is written in FORTRAN77 with a free-field type of input and can be used either in batch or interactive mode. The program utilizes runtime libraries interconnected by a common database which allows applications on large mainframes as well as on microcomputer systems. Examples described in this report have been run in a VAX/VMS environment and on the CRAY X-MP.

Chapter 2 describes briefly the Pseudodynamic Test Method and summarizes the background necessary to utilize it. A complete user's manual for all the PSDYN commands is given in Chapter 3. In Chapter 4 two examples of a three Degrees of Freedom (DOF) structure subjected to a blast loading and to the El Centro 1940 (NS) earthquake are shown.

2. THE PSEUDODYNAMIC TEST METHOD

2.1 INTRODUCTION

In seismically active regions, buildings are usually designed to deform inelastically under severe earthquakes. Currently, analytical methods are unable to fully predict the complex inelastic behavior exhibited by most structural systems under seismic loading conditions. Therefore experimental testing remains the most reliable means to evaluate the inelastic response of structures under critical earthquake loads.

Recently, a new experimental method has been developed which attempts to combine the economy and flexibility of quasi-static tests with the realism of shaking table tests [2,3]. In this method, a computer is used on-line to determine the displacement history to be imposed on a test specimen. Conventional step-by-step integration methods are used to calculate these displacements based on the equations of motion formulated for the specimen. The inertial and damping characteristics of the test structure as well as the earthquake accelerogram are numerically prescribed by the user. Since the structure's restoring forces are likely to vary significantly during a test, they are measured experimentally from the deformed specimen at each step in the test.

By using a direct integration method, the displacement response in each step of a test is computed based on the measured restoring forces from the previous step and on the prescribed inertial and damping characteristics, and then imposed on the specimen using servo controlled hydraulic actuators.

Previous studies have shown that Pseudodynamic Testing (PDT) can be very reliable if appropriate test equipment and techniques are used. It should be enphasized that the accuracy of test results depends largely on the selection of appropriate test specimens, the determination of realistic discrete-parameter models and the use of reliable numerical methods. While lumped mass structural models are most convenient to formulate and test, it may be difficult to apply the method to structures with significant distributed masses. Additionally, viscous damping, strain-rate effects and the performance of loading apparatus may all affect the results of PDT.

Also certain mechanical and numerical problems have been experienced in testing stiff systems which have a large number of DOF. The major problems in these tests were caused by limitations in resolution of the displacement control in servo controlled multi-actuator systems and the sensitivity of numerical computations to pertubations in the experimental data.

Improved actuator-control techniques are under development at the University of Michigan, Ann Arbor [4] to reduce displacement control errors. The propagation of experimental errors in numerical computation has been investigated and methods for artificially supressing the error effects have been developed at U.C. Berkeley [5].

As part of the U.S.-Japan Coordinated Project on Masonry Research for Earthquake Resistant Buildings [6], efforts are under way at the University of California, San Diego, to develop an on-line computer controlled experimental testing procedure suitable for testing geometrically complex shear wall type structures with openings under simulated seismic loads.

In the following a state of the art summary is given on PDT for discrete parameter systems which will form the basis for new developments in on-line testing procedures of full scale structural systems under critical earthquake loads.

2.2 DISCRETE-PARAMETER STRUCTURAL MODEL

The equations of motion for a discrete-parameter system can be represented by a family of second-order ordinary differential equations which can be expressed in a matrix form as:

$$\mathbf{m}\,\mathbf{a}(t) + \mathbf{c}\,\mathbf{v}(t) + \mathbf{k}\,\mathbf{d}(t) = \mathbf{f}(t) \tag{2.1}$$

In this report, vector and matrix quantities are always represented by bold-faced variables as in the equation above, where **m**, **c**, **k**, are the mass, damping and stiffness matrices of the structure and **a**, **v**, **d**, **f**, are the acceleration, velocity, displacement and loading vectors respectively. These equations of motion can be formulated for discrete coordinates by the finite element method.

In the PDT the mass and damping matrices of a test structure which are assumed to be invariable are analytically constructed, while the restoring forces developed by structural deformations are experimentally measured. Thus, the formulation of the stiffness matrix for the discrete structural system is not required. The viscous damping is usually determined based on some idealized modal damping properties of a system, like for instance the Rayleigh damping assumption (see Ref. [7]). A lumped-mass idealization can tremendously simplify the experimental setup and the numerical formulation, and is usually employed in the PDT.

The results of PDT should closely represent the actual dynamic behavior of the test structure as long as the higher frequency responses neglected by the analytical model are insignificant.

2.3 STEP-BY-STEP INTEGRATION METHODS

Dividing the duration T of the structural response into n time steps ($\Delta t = T/n$), a step-by-step integration method is used to transform the set of differential equations of motion (2.1) into n sets of algebraic equations. The solution in each step is a function of the structural response in the previous step or steps. If the displacement solution in each step is a function of previous step solutions only the method is considered explicit, if information from the current time step is utilized then the method is considered implicit. Many implicit methods are unconditionally stable while generally the explicit methods are only conditionally stable.

One of the most general integration method in structural dynamics is the Newmark algorithm :

$$\mathbf{m}\,\mathbf{a}_2 + \mathbf{c}\,\mathbf{v}_2 + \mathbf{k}\,\mathbf{d}_2 = \mathbf{f}_2 \qquad (2.2)$$

$$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\,\mathbf{v}_1 + \Delta t^2 \left\{ \left[\frac{1}{2} - \beta\right] \mathbf{a}_1 + \beta\,\mathbf{a}_2 \right\} \qquad (2.3)$$

$$\mathbf{v}_2 = \mathbf{v}_1 + \Delta t\,[(1-\gamma)\,\mathbf{a}_1 + \gamma\,\mathbf{a}_2] \qquad (2.4)$$

Here the indices 1 and 2 represent respectively the solutions at the beginning and at the end of a certain step i (or the solutions at the time $i\Delta t$ and $(i+1)\Delta t$). The parameters β and γ are selected by the user to achieve desirable stability and accuracy. For $\beta = 1/4$ and $\gamma = 1/2$ the method is called constant average acceleration which is an implicit and unconditionally stable method. When $\beta = 0$ and $\gamma = 1/2$ the method is explicit and conditionally stable.

Since in pseudodynamic testing only the product $\mathbf{k}\,\mathbf{d}_2$ can be measured experimentally, explicit methods are recommended. Although the restoring forces $\mathbf{r}_2 = \mathbf{k}\,\mathbf{d}_2$ and the displacements $\mathbf{d}_2$ are known in each step of a test there is not sufficient information to compute the instantaneous tangent stiffness $\mathbf{k}^t$ for a highly coupled MDOF nonlinear system. Even if a method for determining the tangent stiffness could be devised, the resulting values may be overly sensitive to tolerances in experimental measurements. Furthermore the solution of nonlinear differential equations by an implicit method usually requires an iterative procedure which can be another source of error accumulation.

Japanese researches [2,8] have found the Central Difference Method (CDM) which is an explicit integration method the most suitable one for the PDT method. However, Shing and Mahin [5] have shown that the basic CDM is more sensitive to experimental errors than the Newmark explicit method or than the summed-form of the CDM. They have also recommended a modified Newmark method which has a numerical dissipation property and the numerical damping is approximately frequency-proportional. Numerical dissipation is very useful for supressing the spurious growth of high frequency responses encountered in pseudodynamic testing of MDOF systems. Only these two methods, the Newmark explicit and the modified Newmark, were implemented in the PSDYN program and are presented next.

(i) The Newmark Explicit Method

$$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1 \tag{2.5}$$

$$\mathbf{v}_2 = \mathbf{v}_1 + \frac{\Delta t}{2} [\mathbf{a}_1 + \mathbf{a}_2] \tag{2.6}$$

$$\left[\mathbf{m} + \frac{\Delta t}{2} \mathbf{c}\right] \mathbf{a}_2 = \mathbf{f}_2 - \mathbf{k}\, \mathbf{d}_2 - \mathbf{c}\, \mathbf{v}_1 - \frac{\Delta t}{2} \mathbf{c}\, \mathbf{a}_1 \tag{2.7}$$

$$\mathbf{r}_2 = \mathbf{k}\, \mathbf{d}_2$$

(ii) The Modified Newmark Explicit Method

$$\mathbf{m}\, \mathbf{a}_2 + \left\{[1 + \alpha]\, \mathbf{k} + \frac{\rho}{\Delta t^2} \mathbf{m}\right\} \mathbf{d}_2 = \mathbf{f}_2 + \left[\alpha\, \mathbf{k} + \frac{\rho}{\Delta t^2} \mathbf{m}\right] \mathbf{d}_1 \tag{2.8}$$

$$\mathbf{r}_2 = \mathbf{k}\,\mathbf{d}_2$$
$$\mathbf{r}_1 = \mathbf{k}\,\mathbf{d}_1$$

$$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\,\mathbf{v}_1 + \frac{\Delta t^2}{2}\mathbf{a}_1 \tag{2.9}$$

$$\mathbf{v}_2 = \mathbf{v}_1 + \frac{\Delta t}{2}[\mathbf{a}_1 + \mathbf{a}_2] \tag{2.10}$$

2.3.1 STABILITY AND ACCURACY

A linear elastic SDOF system is considered in the following analysis. However, the results obtained herein are applicable to linear MDOF systems in general by means of modal superposition. In solving the equation of motion for a free-vibration response of a linear elastic SDOF system, a step-by-step integration algorithm can be written in a recursive matrix form as:

$$\mathbf{x}_2^{(i)} = \mathbf{A}\,\mathbf{x}_1^{(i)} \tag{2.11}$$

where $\mathbf{x}_1^{(i)}$ is a solution vector (kx1) which contains the displacement, velocity and/or acceleration terms at the beginning of step i and $\mathbf{x}_2^{(i)}$ at the end of step i; **A** is called the amplification matrix.

The numerical properties of a step-by-step algorithm can be obtained from its corresponding amplification matrix **A**.

The solution at step n is obtained by applying (2.11) recursively:

$$\mathbf{x}_2^{(n)} = \mathbf{A}\,\mathbf{x}_1^{(n)} = \mathbf{A}^2\,\mathbf{x}_1^{(n-1)} = \ldots = \mathbf{A}^n\,\mathbf{x}_1^{(1)} \tag{2.12}$$

where $\mathbf{x}_1^{(1)}$ is the initial conditions vector.

The spectral decomposition of the amplification matrix **A** (k x k) gives:

$$\mathbf{A}^n = \mathbf{\Phi}\,\mathbf{J}^n\,\mathbf{\Phi}^{-1} \tag{2.13}$$

where

$$\mathbf{\Phi} = [\phi_1\,\phi_2\ldots\phi_k]$$

and

$$\mathbf{J} = \mathrm{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_k) \quad ;$$

the vectors ϕ_i are the eigenvectors corresponding to the eigenvalues λ_i of the matrix **A**.

From Eqs. (2.12) and (2.13) the numerical solution of the free-vibration response, the displacement at the end of step n, can be written as:

$$d_2^{(n)} = c_1\,\lambda_1^n + c_2\,\lambda_2^n + \ldots + c_k^n\,\lambda_k^n \tag{2.14}$$

where $c_1, c_2, \ldots, c_k$ are constants determined from initial conditions. For most algorithms **A** can be formulated as a 2 x 2 or 3 x 3 matrix.

Calling $\rho\,(\mathbf{A}) = \max\,|\lambda_i|$ the spectral radius of **A**, $\mathbf{J}^n$ is said to be bounded when $n \to \infty$ if and only if $\rho\,(\mathbf{A}) \leq 1$. Moreover, in order to Eq. (2.14) represent an oscillatory response, two of the eigenvalues of **A**, $\lambda_{1,2}$ should be complex conjugates. The third eigenvalue λ_3 (if it exists) is called the spurious root since it does not have a physical meaning. In summary, the stability conditions for an integration algorithm are:

$$|\lambda_3| < |\lambda_{1,2}| \leq 1$$

The eigenvalues λ_1 and λ_2 can be represented as:

$$\lambda_{1,2} = A \pm i\,B = e^{(-\bar{\xi} \pm i)\bar{\Omega}} \tag{2.15}$$

where $i = \sqrt{-1}$ and A, B are real numbers such that $(A^2 + B^2) \leq 1$ and

$$\bar{\xi} = -\frac{\ln\,(A^2 + B^2)}{2\bar{\Omega}} \tag{2.16}$$

$$\bar{\Omega} = \arctan\left(\frac{B}{A}\right) \tag{2.17}$$

Calling $\bar{\Omega} = \bar{\omega}\Delta t$ and substituting Eq. (2.15) into Eq. (2.14), can get:

$$d^{(n)} = c\;e^{(-\bar{\xi} \pm i)n\bar{\omega}\Delta t} \tag{2.18}$$

The exact solution of an undamped free-vibration response is given by:

$$d(t) = c\,e^{\pm i\omega t} \tag{2.19}$$

where ω is the natural response frequency.

By comparing Eqs. (2.18) and (2.19) it can be induced that physically $\overline{\xi}$ represents the numerical damping and $\overline{\omega}$ the frequency of the approximated response. Therefore, the numerical inaccuracies of an algorithm can be measured by its numerical damping $\overline{\xi}$ and by the percentage of frequency distortion $(\overline{\omega} - \omega)/\omega$ or, more commonly by the percentage of period distortion $(T - \overline{T})/T$, where $T = 2\pi/\omega$. These numerical properties and the stability of both algorithms described in the previous sections are considered next:

(i) The Newmark Explicit Method

The Newmark explicit method has the following solution scheme for a SDOF free-vibiation response:

$$d_2 = d_1 + \Delta t\, v_1 + \frac{\Delta t^2}{2} a_1$$

$$v_2 = v_1 + \frac{\Delta t}{2}(a_1 + a_2)$$

$$a_2 = -\omega^2 d_2$$

which can be written in a recursive form as Eq. (2.11) with:

$$\mathbf{x}_1 = \begin{Bmatrix} d_1 \\ v_1 \\ a_1 \end{Bmatrix} \quad ; \quad \mathbf{x}_2 = \begin{Bmatrix} d_2 \\ v_2 \\ a_2 \end{Bmatrix}$$

and

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \dfrac{\Delta t^2}{2} \\ -\dfrac{\omega^2 \Delta t}{2} & 1-\dfrac{\omega^2 \Delta t^2}{2} & \dfrac{\Delta t - \omega^2 \Delta t^3}{2} \\ -\omega^2 & -\omega^2 \Delta t & -\dfrac{\omega^2 \Delta t^2}{2} \end{bmatrix} \tag{2.20}$$

The eigenvalues of matrix **A** are:

$$\lambda_{1,2} = A \pm i B \quad \text{and} \quad \lambda_3 = 0$$

where

$$A = 1 - \frac{\omega^2 \Delta t^2}{2} \tag{2.21}$$

$$B = \frac{\sqrt{4 - (\omega^2 \Delta t^2 - 2)^2}}{2} \tag{2.22}$$

Stability :

The stability conditions require that $(A^2 + B^2) \leqslant 1$ and that B is a real number. Since $(A^2 + B^2)$ is always equal to 1 here, the numerical stability is governed by the condition that:

$$(\omega^2 \Delta t^2 - 2)^2 \leqslant 4 \quad \rightarrow$$

$$\omega \Delta t \leqslant 2 \quad \text{or} \quad \frac{\Delta t}{T} \leqslant \frac{1}{\pi} \tag{2.23}$$

Accuracy :

Substituting (2.21) and (2.22) into (2.16) and (2.17) yields:

$$\overline{\xi} = 0 \tag{2.24}$$

and

$$\overline{\omega} = \frac{1}{\Delta t} \arctan \left[\frac{\sqrt{4 - (\omega^2 \Delta t^2 - 2)^2}}{2 - \omega^2 \Delta t^2} \right] \tag{2.25}$$

Therefore the Newmark explicit method has no numerical damping, the only source of innacuracy is the frequency or period distortion which is a function of $\omega \Delta t$. In Fig. (2.1), the percentage of period distortion $(T - \overline{T})/T$ against $\Delta t/T$ is plotted. It can be observed that period distortion will approach zero as $\Delta t/T$ goes to zero and that a reasonably accurate solution can be obtained (less than 1% distortion) when $\Delta t/T$ is less than 0.05 (Δt = T/20). This level of accuracy is comparable to those of the most reliable implicit methods (see Ref. [9]).

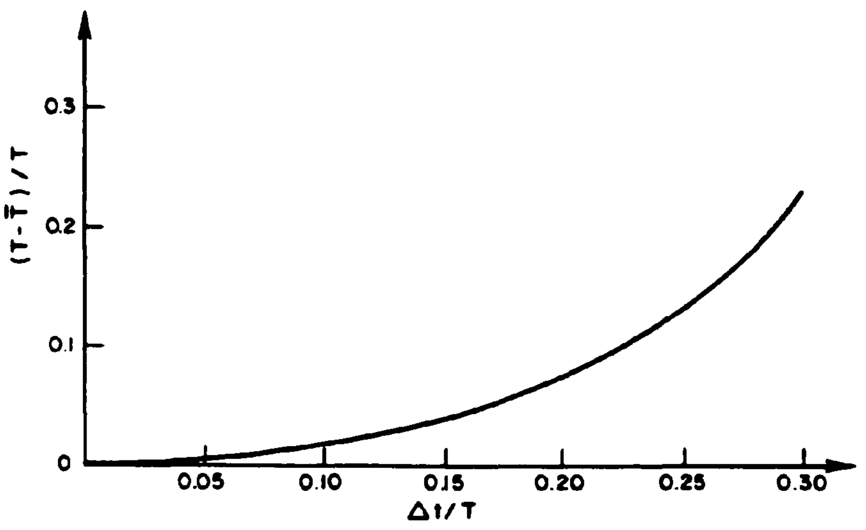


Fig. 2.1 - Period Shrinkage by the Newmark Explicit Method

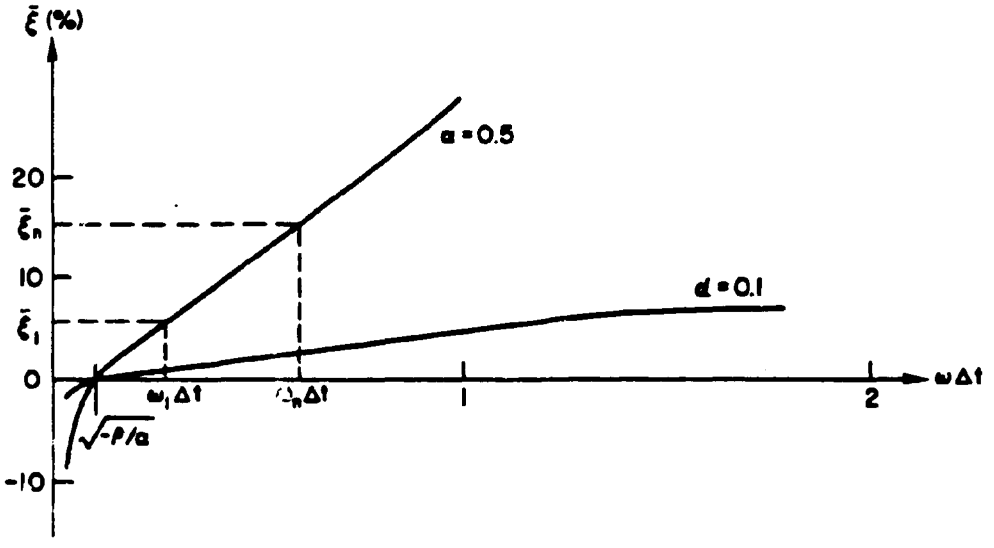


Fig. 2.2 - Numerical Damping by the Modified Newmark Explicit Method

(ii) The Modified Newmark Explicit Method

Based on the equations (2.8) to (2.10) the modified Newmark algorithm can be written in a recursive form as (2.11) with:

$$\mathbf{x}_1 = \begin{Bmatrix} d_1 \\ \Delta t\ v_1 \\ \Delta t^2\ a_1 \end{Bmatrix} \quad ; \quad \mathbf{x}_2 = \begin{Bmatrix} d_2 \\ \Delta t\ v_2 \\ \Delta t^2\ a_2 \end{Bmatrix}$$

and

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \frac{1}{2} \\ -\frac{\Omega^2}{2} & 1 - \left(1 + \alpha\right)\frac{\Omega^2}{2} - \frac{\rho}{2} & \frac{1}{2} - \left(1 + \alpha\right)\frac{\Omega^2}{4} - \frac{\rho}{4} \\ -\Omega^2 & -\left(1 + \alpha\right)\Omega^2 - \rho & -\left(1 + \alpha\right)\frac{\Omega^2}{2} - \frac{\rho}{2} \end{bmatrix} \qquad (2.26)$$

in which $\Omega = \omega\Delta t$.

Stability :

Matrix **A** has eigenvalues:

$$\lambda_{1,2} = A_1 \pm (A_1^2 - A_2)^{1/2} \quad \text{and} \quad \lambda_3 = 0$$

where

$$A_1 = 1 - (1 + \alpha)\frac{\Omega^2}{2} - \frac{\rho}{2} \qquad (2.27)$$

$$A_2 = 1 - \alpha\Omega^2 - \rho \qquad (2.28)$$

In order for $\lambda_{1,2}$ to be complex conjugates and $|\lambda_{1,2}| \leqslant 1$, A_1 , A_2 must satisfy $A_1^2 < A_2 \leqslant 1$. When $A_1^2 = A_2$ the algorithm will have a non-oscillatory solution but the solution will remain stable as long as $A_2 \leqslant 1$. The condition $A_1^2 \leqslant A_2$ implies that:

$$\frac{-1 + \sqrt{1 - (1 + \alpha)\rho}}{1 + \alpha} \leqslant \Omega \leqslant \frac{1 + \sqrt{1 - (1 + \alpha)\rho}}{1 + \alpha}$$

and $A_2 \leqslant 1$ gives:

$$\Omega \geqslant \sqrt{-\frac{\rho}{\alpha}}$$

To obtain approximately frequency-proportional numerical damping, ρ should be selected negative and α positive. Under these assumptions the stability conditions are equivalent to:

$$\sqrt{-\frac{\rho}{\alpha}} \leqslant \Omega \leqslant \frac{1 + \sqrt{1 - (1 + \alpha)\rho}}{1 + \alpha} \tag{2.29}$$

Accuracy :

According to Eqs. (2.16) and (2.17) the eigenvalues of **A** from Eqs. (2.27) and (2.28) give:

$$\bar{\xi} = -\frac{\ln (1 - \alpha\Omega^2 - \rho)}{2\Omega} \tag{2.30}$$

and

$$\bar{\omega} = \frac{1}{\Delta t} \arctan \left[\frac{B}{A}\right] \tag{2.31}$$

where

$$A = 1 - (1 + \alpha)\frac{\Omega^2}{2} - \frac{\rho}{2}$$

and

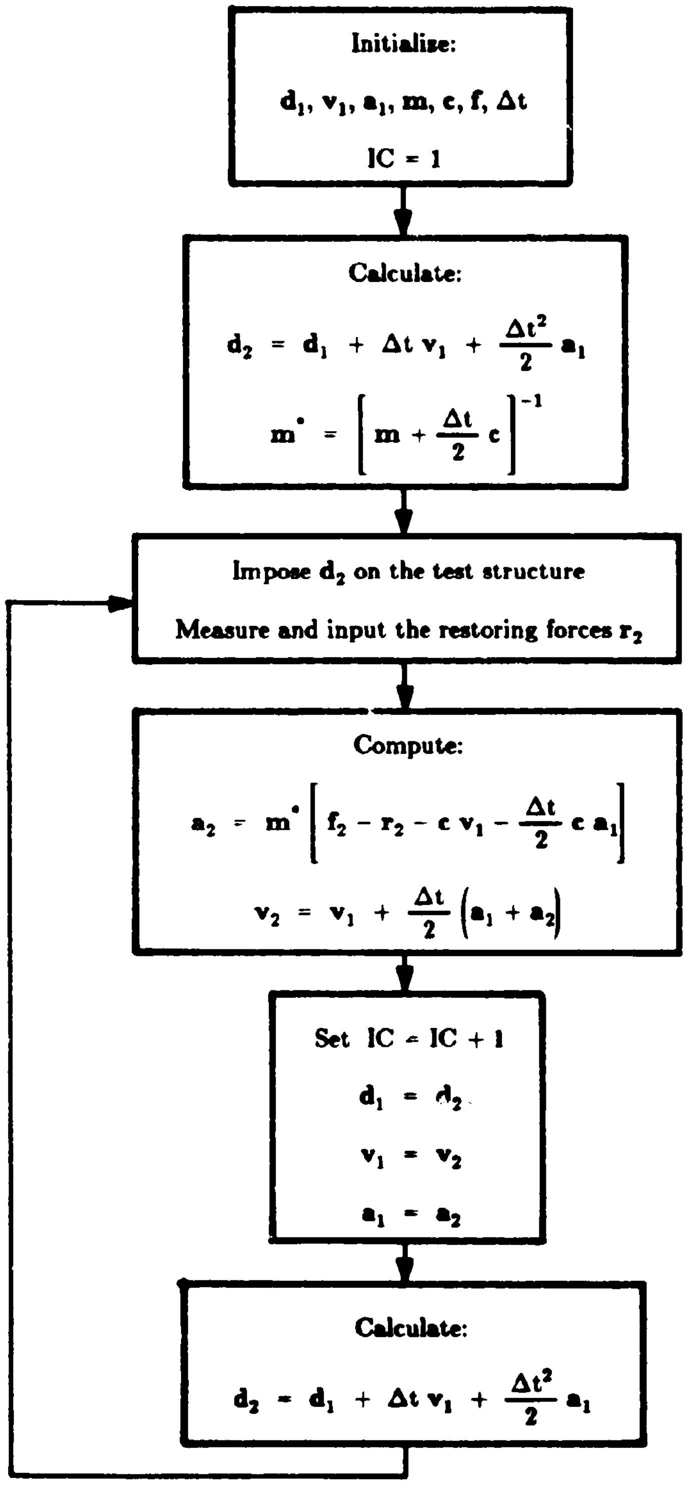
$$B = \left\{\Omega^2 - \left[(1 + \alpha)\frac{\Omega^2}{2} + \frac{\rho}{2}\right]^2\right\}^{1/2}$$

From Eq. (2.30) it can be seen that $\bar{\xi} = 0$ when $\Omega = \sqrt{-\frac{\rho}{\alpha}}$. For $\Omega < \sqrt{-\frac{\rho}{\alpha}}$, damping is negative and the solution becomes unstable, i.e., energy is added into the numerical solution. For α equal to 0.1 and 0.5 , $\bar{\xi}$ against $\omega\Delta t$ is plotted in Fig. (2.2) where for both curves $\sqrt{-\frac{\rho}{\alpha}}$ is 0.1. It can be observed that damping increases with increasing α. Consequently, by an appropriate combination of α and ρ, one can have a zero or small damping for the fundamental mode while having a significantly greater damping for a higher frequency. This characteristic is very useful for supressing the spurious growth of higher mode responses encountered in pseudodynamic testing of MDOF systems.

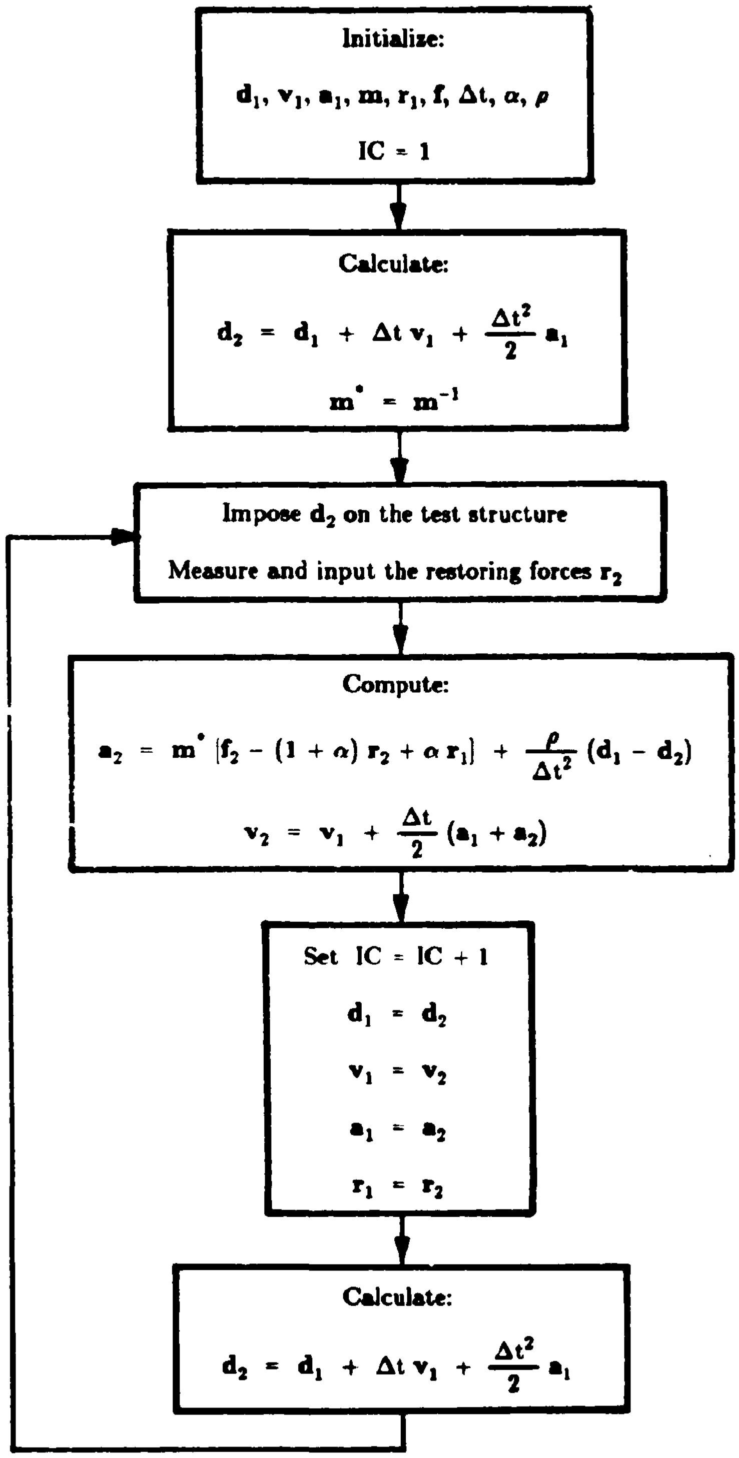
NOTE : The stability and accuracy properties of numerical integration methods in solving *nonlinear* differential equations are not very well understood due to the lack of an analytical evaluation technique. It can be shown that unconditionally stable implicit methods can become unstable when applied to nonlinear problems with large integration time intervals. This is caused mainly by the fact that implicit integration methods usually require an approximate solution procedure such as iterative correction, tangent modulus or pseudo-force approximation when solving nonlinear equations. This problem does not occur when explicit integration methods are employed since a direct solution for a nonlinear equation is utilized. However, for nonlinear systems, the integration time interval selected should always be sufficiently small so that the nonlinear behavior can be accurately traced with the discretized displacements increments. The Δt selected for a linear system will remain conservative if the nonlinearity is of the softening type. This is because the effective $\Delta t/T$ ratio will be smaller as a system becomes less stiff. The opposite will be true for a hardening system.

2.3.2 IMPLEMENTATION

(i) Newmark Explicit Method


Initialize:
$\mathbf{d}_1, \mathbf{v}_1, \mathbf{a}_1, \mathbf{m}, \mathbf{c}, \mathbf{f}, \Delta t$
IC = 1
Calculate:
$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$
$\mathbf{m}^* = \left[\mathbf{m} + \frac{\Delta t}{2} \mathbf{c} \right]^{-1}$
Impose $\mathbf{d}_2$ on the test structure
Measure and input the restoring forces $\mathbf{r}_2$
Compute:
$\mathbf{a}_2 = \mathbf{m}^* \left[\mathbf{f}_2 - \mathbf{r}_2 - \mathbf{c}\, \mathbf{v}_1 - \frac{\Delta t}{2} \mathbf{c}\, \mathbf{a}_1 \right]$
$\mathbf{v}_2 = \mathbf{v}_1 + \frac{\Delta t}{2} \left(\mathbf{a}_1 + \mathbf{a}_2 \right)$
Set IC = IC + 1
$\mathbf{d}_1 = \mathbf{d}_2$
$\mathbf{v}_1 = \mathbf{v}_2$
$\mathbf{a}_1 = \mathbf{a}_2$
Calculate:
$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$

(ii) Modified Newmark Explicit Method


Initialize:
$\mathbf{d}_1, \mathbf{v}_1, \mathbf{a}_1, \mathbf{m}, \mathbf{r}_1, \mathbf{f}, \Delta t, \alpha, \rho$
IC = 1
Calculate:
$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$
$\mathbf{m}^* = \mathbf{m}^{-1}$
Impose $\mathbf{d}_2$ on the test structure
Measure and input the restoring forces $\mathbf{r}_2$
Compute:
$\mathbf{a}_2 = \mathbf{m}^* [\mathbf{f}_2 - (1+\alpha)\, \mathbf{r}_2 + \alpha\, \mathbf{r}_1] + \frac{\rho}{\Delta t^2} (\mathbf{d}_1 - \mathbf{d}_2)$
$\mathbf{v}_2 = \mathbf{v}_1 + \frac{\Delta t}{2} (\mathbf{a}_1 + \mathbf{a}_2)$
Set IC = IC + 1
$\mathbf{d}_1 = \mathbf{d}_2$
$\mathbf{v}_1 = \mathbf{v}_2$
$\mathbf{a}_1 = \mathbf{a}_2$
$\mathbf{r}_1 = \mathbf{r}_2$
Calculate:
$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$

NOTE : In the flow-charts presented in the previous pages, the first two boxes correspond to the INIT command and the last three boxes correspond to the PSEUDO command of the PSDYN program. These commands are described in the next chapter, in section 3.2.4

3. USER INFORMATION

3.1 PSDYN PROGRAM AND INTERACTION

PSDYN incorporates a set of commands which are described in the next section separated into four groups according to the operation type. Each command can perform operations on matrices formed by other commands by using the common database.

The program is designed to operate in either an interactive mode by reading input from the terminal or in a batch mode by reading input lines (commands marked by separators) from a file either specified by the user or from the default file INPUT. Interactive mode operation will prompt for all missing data not supplied with the original command line. When in "interactive mode" all commands typed in at the screen are recorded in the file OUTPUT. In addition, the output from a PRINT command is also written to this file. When in "batch mode" the commands being read from the input file are echoed to the screen and the results of those commands are written into the file OUTPUT. In this way a complete record of a PSDYN run may be kept.

3.2 PSDYN COMMANDS

There are a few general rules that must be followed for execution of any PSDYN command. All commands are of the form:

OP M1+ M2- N=N1,N2,..

where

OP	is the operation to be executed. This can be 1 to 6 characters.
Mi	is the name of the array or separator to be used for that operation. Only the first four characters are read by the program.
Ni	is a set of i additional parameters.

Some of the operations have single character abbreviations. They are designated by the single character enclosed in parenthesis. A command may require none or up to eight matrices. The '+' or '-' after the Mi designates the condition of the matrix after the command is executed. A '+' means the array will be created by the command and therefore such an array must not exist or it will be deleted. A '-' means the matrix will be changed in some manner by the command. If neither '+' or '-' follows the Mi then the matrix is unchanged after the operation. A command may or may not require any additional parameter lists.

General input conventions are:

-A "C" in column 1 of any line denotes a comment line.

-A backslash "\" at the end of information on an input line will allow a second line extension to a total of 160 characters.

-A colon ":" indicates the end of information on a line. Information to the right of the colon is ignored by the program.

-Requested data not supplied by the user will be automatically set to sero or blank depending on the routine used.

-Real numbers do not require decimal points. E format is accepted.

-Arithmetic statements (+,-,*,/) may be used within the data input. The order of evaluation is sequential.

-Input data must be separated by a "blank" or "comma".

3.2.1 PROGRAM AND DATABASE MANAGEMENT MACRO COMMANDS

DELETE (D) Mi-

This command deletes one or more matrices named Mi, where i=1,2,..,6.

DUPDG M1 M2+

This command forms the row matrix M2 which consists of the diagonal values of the matrix M1.

DUPSM M1 M2+ R=R1 C=C1 L=L1,L2

This command forms the matrix M2 which is a submatrix of M1. The submatrix is R1 rows by C1 columns. The submatrix starts at the location L1,L2 of the matrix M1.

HELP (H)

This command lists PSDYN's operations.

LIST (L)

This command lists the directory of all files known to the database.

LOAD M1 R=R1 C=C1

This command loads the real matrix M1 with R1 rows and C1 columns. The matrix is entered row-wise, one row per line.

MODIFY M1-

This command modifies any individual term of the matrix M1. This command can only be used interactively.

RETURN

This is the last command in a separator group of a submit file. This command returns control to the interactive mode after executing the commands following a separator.

START

This command reinitializes the database by deleting all files.

STODG M1- M2

This command stores the row or column matrix M2 on the diagonal of matrix M1.

STOP (S)

This command stops execution of the current segment and saves the database.

STOSM M1- M2 L=L1,L2

This command stores the matrix M2 as a submatrix in matrix M1. The submatrix starts at the location L1,L2 of matrix M1.

ZERO M1+ R=R1 C=C1

This command forms the real matrix M1 which is R1 rows by C1 columns. The values of all terms in the matrix are zero.

3.2.2 MATRIX OPERATION MACRO COMMANDS

ADD M1- M2

This command adds the matrices M1 and M2 and stores the result in M1.

DUP M1 M2+

This command forms the matrix M2 which is the duplicate of M1.

INVERT M1-

This command inverts the matrix M1. The inverse is stored in M1. M1 must be symmetric and positive definite.

MULT M1 M2 M3+

This command multiplies matrix M1 times M2 and stores the result in M3.

SCALE M1- M2

This command scales all the terms of matrix M1 by the (1,1) term of matrix M2.

SOLVE M1- M2- S=S1

This command solves the set of equations M1 x = M2. The result is written back into M1 and/or M2.

where	S1	= 1 complete solution of M1 x = M2. x is returned in M2.
		= 2 triangularization of M1 only. The result is stored in M1.
		= 3 forward and back substitution of M2 only. The result is stored in M2.

SQREL M1-

This command replaces each term of the matrix M1 with the square root of the term.

SUB M1- M2

This command subtracts the matrix M2 from M1 and stores the results in M1.

TRAN M1 M2+

This command forms the matrix M2 which is the transpose of matrix M1.

3.2.3 I/O OPERATION MACRO COMMANDS

DEFINE M1 M2 R=R1 C=C1

This command defines the directory for a matrix stored on a disk file. This command is only needed if the database does not contain the matrix directory.

where

M1	is the matrix for which the directory is to be created.
M2	is the matrix type. M2=R if the matrix is real. M2=I if the matrix is integer.
R1	is the number of rows in M1.
C1	is the number of columns in M1.

FILE M1

This command saves the matrix M1 in a disc file with the name "INPUT.EXT".

where INPUT	is the name of the current input file. The default is INPUT.
EXT	is the three character extension made up of the first three characters of the name of the matrix M1.

INPUT

This command is used to change the input file for batch operation from the default file INPUT to any other file. It can only be used from the interactive mode. It prompts for the input file name.

PLOT M1 N=N1 R=R1,R2,... S=S1,S2,...

This command generates a screen plot of the rows of matrix M1.

where

M1	is the matrix to be plotted.
N1	is the number of rows of N1 that are to be plotted.
Ri	is the row number to be plotted from M1.
Si	is the symbol to be used for plotting row i.

There *must* be N1 entries for Ri and Si.

PRINT (P) M1

This command prints the matrix named M1.

RFILE M1

This command reads into core the matrix M1 which is stored on a disk file. The database directory must contain an entry for the matrix M1. If no entry exists, one must be created by the DEFINE command.

SUBMIT M1

This command starts the batch execution. It executes the commands following the M1 separator in the input file. The input file name is defaulted to the name "INPUT".

3.2.4 DYNAMIC OPERATION MACRO COMMANDS

EIGEN M1- M2+ M3- T=T1

This command solves the following eigenvalue problem;

$$K \Phi = \lambda M \Phi$$

where

M1	is the N by N symmetric, positive definite matrix K.
M2	is the N by N matrix containing the eigenvectors Φ.
M3	is a row or column vector containing the diagonal terms of the matrix M. The values for the mass matrix MUST be positive. After the operation is completed , it contains the eigenvalues, λ.
T1	is the approximate number of significant digits for the eigenvalues. The default for T1 is 4.

The program reduces the problem to the standard form by the following transformation;

$$K' = m^T K m$$

where

$$I = m^T M m$$

in which

$$m_i = \frac{1}{\sqrt{M_{ii}}}$$

The calculated mode shapes Φ, are normalized by;

$$\Phi^T M \Phi = I$$
$$\Phi^T K \Phi = \lambda$$

The operation uses the standard Jacobi method to solve for all eigenvalues and eigenvectors.

FUNG M1 M2+ T=T1 L=L1,L2

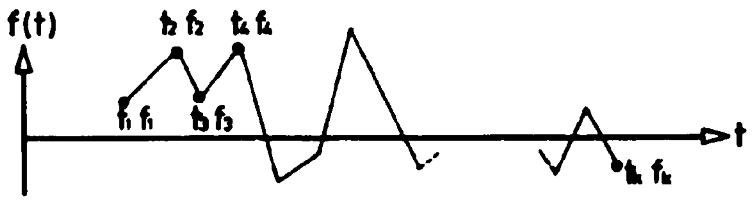
This operation forms the matrix M2, which contains values, at equal intervals, of the function specified by the matrix M1.

where

M1 is the input function, M1 is 2 x K values of the form;

$$M1 = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & \dots\dots & t_k \\ f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & \dots\dots & f_k \end{bmatrix}$$

which represents a function of the form;



M2	is the output time function at equal intervals in time.
T1	is the delta time increment to be used in the matrix M2.
L1	is the number of function values to be formed in M2. This should be $\leqslant (t_k - t_1)/\Delta t + 1$.
L2	is the number of rows to be generated for the matrix M2. L2 = 1 for just the function values. L2 = 2 for both the function and time values.

The matrix M2 can have two forms. If L2 = 1, M2 will be a 1 x L1 row matrix containing the function values at time increment T1. If L2 = 2, M2 will be a 2 x L1 matrix in which the first row contains the time value and the second row contains the corresponding function value.

INIT I=I1 M1 M2 M3 M4+ M5 M6- M7 M8 T=T1 (P=P1,P2)

This is the initial command for the Pseudodynamic Method and it *must* be executed only once. It initializes and prepares the arrays and parameters necessary to execute the step command PSEUDO. The array names (M1 to M8) and parameters (I,T,P) are written to a file named INITIAL. Another file named COUNTER which contains the step ID number (IC=1 at this stage) is created in this command. This step number will be updated at the end of each PSEUDO operation.

The parameter I defines which step-by-step method will be used by the PSEUDO command:

I1=1	Newmark Explicit Method
I1=2	Modified Newmark Explicit Method

when I1=2, P1 and P2 must be specified, where:

P1	is the parameter α and
P2	is the parameter ρ (see section 2.3).

This command calculates the inverse of the effective mass matrix and also the displacement vector at the end of the first step:

$$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$$

where

M1	is the name of the N by 1 displacement vector at the beginning of step 1, $\mathbf{d}_1$ (default is the null vector).
M2	is the name of the N by 1 velocity vector at the beginning of step 1, $\mathbf{v}_1$ (default is the null vector).
M3	is the name of the N by 1 acceleration vector at the beginning of step 1, $\mathbf{a}_1$ (default is the null vector).
M4	is the name of the N by 1 calculated displacement vector at the end of step 1, $\mathbf{d}_2$.
M5	for I1=1 $\rightarrow$ is the name of the N by N damping matrix, $\mathbf{c}$ (default is the null matrix). for I1=2 $\rightarrow$ is the name of the N by 1 restoring force vector at the beginning of step 1, $\mathbf{r}_1$ (default is the null vector).
M6	is the name of the N by N mass matrix, $\mathbf{m}$. After the operation is completed it gives $\mathbf{m}^*$ where: for I1=1 $\rightarrow \mathbf{m}^* = \left[\mathbf{m} + \frac{\Delta t}{2}\mathbf{c}\right]^{-1}$. for I1=2 $\rightarrow \mathbf{m}^* = \mathbf{m}^{-1}$.
M7	is the name of the N by 1 load distribution vector, $\mathbf{p}$.
M8	is the name of the 1 by (n + 1) load multipliers matrix, $\mathbf{f}_m$, at equal increments Δt , ($\mathbf{f} = \mathbf{p}\, \mathbf{f}_m$).
T1	is the time interval Δt.

NOTE : Here N represents the number of DOF and n the number of steps or time intervals which divide the duration T of the response, $\Delta t = T/n$.

PSEUDO M1-

This is the step command for the Pseudodynamic Method, where:

M1	is the name of the N by 1 restoring force vector at the end of the current step, $\mathbf{r}_2$. This vector will be deleted at the end of this operation.

In this command the array names and parameters defined in the command INIT are read from the file INITIAL. Also the step number (IC) is read from the file COUNTER.

By using the arrays from the previous step $\mathbf{d}_1$, $\mathbf{v}_1$, $\mathbf{a}_1$, [$\mathbf{r}_1$] and the arrays defined in INIT $\mathbf{c}$, $\mathbf{m}^*$, $\mathbf{p}$, $\mathbf{f}$, (all kept in the DATABASE) the external load forces, $\mathbf{f}_2$ are formed and the accelerations and velocities at the end of the current step, $\mathbf{v}_2$ and $\mathbf{a}_2$ respectively, computed.

Next a new step is set (IC=IC+1) and the displacements, velocities, accelerations, [restoring forces] updated:

$$\begin{aligned} \mathbf{d}_1 &= \mathbf{d}_2 \\ \mathbf{v}_1 &= \mathbf{v}_2 \\ \mathbf{a}_1 &= \mathbf{a}_2 \\ [\mathbf{r}_1 &= \mathbf{r}_2] \end{aligned}$$

Finally the displacements at the end of this new step, $\mathbf{d}_2$, are calculated:

$$\mathbf{d}_2 = \mathbf{d}_1 + \Delta t\, \mathbf{v}_1 + \frac{\Delta t^2}{2} \mathbf{a}_1$$

These displacements $\mathbf{d}_2$ will be imposed on the test structure and a new vector of restoring forces $\mathbf{r}_2$ will be measured and loaded into the program. PSEUDO can then be executed again and so on.

NOTES :

1) Here [ ] means just to consider what is inside the brackets for I1=2.

2) For backup the arrays $\mathbf{d}_1$, $\mathbf{v}_1$, $\mathbf{a}_1$, $\mathbf{d}_2$, $\mathbf{c}$ or $\mathbf{r}_1$, $\mathbf{m}^*$, $\mathbf{p}$, $\mathbf{f}$ are also saved on a disk file (out of core) with the names respectively: INPUT.DI1, INPUT.VE1, INPUT.AC1, INPUT.DI2, INPUT.DAM or INPUT.RS1, INPUT.EMI, INPUT.ELD and INPUT.FOR where INPUT is the name of the current input file (the default is INPUT).

In case the DATABASE file is lost or deleted by mistake, the arrays can be retrieved by using the commands DEFINE and RFILE (see section 3.2.3). Then the DUP command (see section 3.2.2) can be used to change the array names to those specified in the INIT command.

STEP M1- M2 M3 M4- M5+ M6 M7 T=T1 L=L1,L2 P=P1,P2,P3

This command calculates the dynamic response of a structural system using direct step by step integration of the following linear matrix equations of motion;

$$M \ddot{U} + C \dot{U} + K U = R(t) = p F(t)$$

where

M1	is the name of the N by N stiffness matrix K.
M2	is the name of the N by N mass matrix M.
M3	is the name of the N by N damping matrix C.
M4	is the name of the N by 3 initial condition matrix U_0. $U_0(1,1)$ is a vector of the initial displacements U_0. $U_0(1,2)$ is a vector of the initial velocities $\dot{U}_0$. $U_0(1,3)$ is a vector of the initial accelerations $\ddot{U}_0$.
M5	is the name of the N by L2 matrix of calculated displacements. Column i represents the displacement at time $i \times L1 \times \Delta$ t.
M6	is the name of the N by 1 load distribution matrix p.
M7	is the name of the 1 by K row matrix representing the load multipliers f at equal time increments Δ t, where K=L1*L2 + 1.
T1	is the time increment Δ t.
L1	is the output interval for the displacements. That is, displacements will be output at each L1 time step.
L2	is the total number of displacement vectors to be output. Therefore, the total time for which results will be calculated will be $L1 \times L2 \times \Delta$ t.
P1	is the γ value for the integration type. (default = 1/2)
P2	is the β value for the integration type.
P3	is the θ value for the integration type. (default = 1.0)

The use of different values of γ, β and θ allows the user to select different methods of step by step integration. The following table lists possible values.

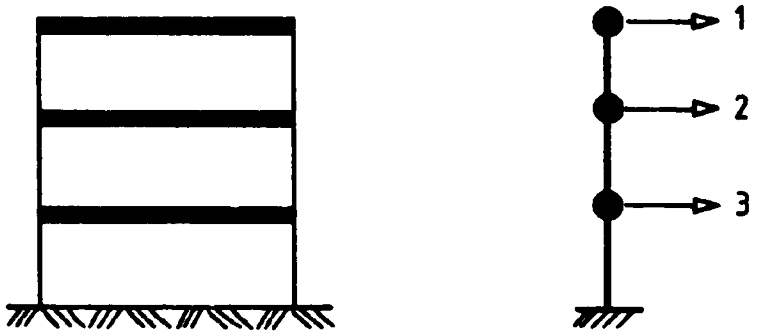
	γ	β	θ
Newmark Average Acceleration	1/2	1/4	1.00
Newmark Explicit	1/2	0.00	1.00
Linear Acceleration	1/2	1/6	1.00

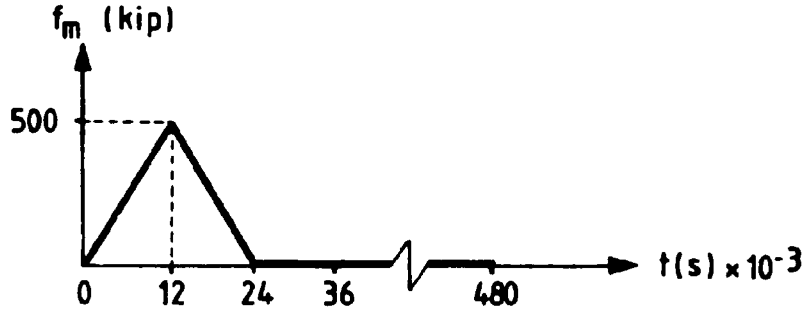
Wilson - Theta Method (Low Damping)	**1/2**	**1/6**	**1.42**
Wilson - Theta Method (High Damping)	**1/2**	**1/6**	**2.00**

4. EXAMPLES

4.1 EXAMPLE 1

A three-story frame structure, modelled by a lumped-mass system with 3 translational DOF, is subjected to a triangular pulse blast-pressure as shown below:





$$\mathbf{f}(t) = \mathbf{p}\, f_m(t) \quad ; \quad \mathbf{p} = \begin{Bmatrix} 1 \\ 2 \\ 2 \end{Bmatrix}$$

The mass and the linear elastic stiffness matrices for this frame are:

$$\mathbf{m} = (1\ \text{kip}\cdot\text{s}^2/\text{in}) \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.5 & 0.0 \\ 0.0 & 0.0 & 2.0 \end{bmatrix}$$

$$\mathbf{k} = (600\ \text{kip}\ /\text{in}) \begin{bmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 5 \end{bmatrix}$$

A dynamic analysis of this structure using the PDT method by means of the PSDYN program is shown in this example (just the first steps are shown). Both explicit methods described previously, the Newmark and the modified Newmark method are utilized.

The INPUT file and the OUTPUT file produced by the program are listed after each example. The INPUT file contains the commands to be executed in the batch mode by means of the INPUT and SUBMIT commands. In the OUTPUT file all the commands following "**" were typed to the screen by the user in the interactive mode.

To select the time interval Δt the natural frequencies of the structure have to be determined which can be done using the command EIGEN; the result is:

$$\begin{Bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{Bmatrix} = \begin{Bmatrix} 14.5 \\ 31.1 \\ 46.1 \end{Bmatrix} \text{rad/s}$$

Hence,

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix} = \begin{Bmatrix} 0.4333 \\ 0.2020 \\ 0.1363 \end{Bmatrix} \text{s}$$

a) Using the Newmark explicit method

- *Selection of the time interval* Δt

Selecting

$$\Delta t = 0.012\text{s} = \frac{T_1}{36.1} = \frac{T_2}{16.8} = \frac{T_3}{11.4}$$

the stability condition (2.23) governed in this case by the third natural period ($\Delta t \leqslant T_3/\pi$) is satisfied and also a good accuracy can be achieved (see Fig. 2.1).

- *Viscous damping*

Considering the Rayleigh damping assumption (see Ref. [7]) : $\mathbf{c} = \mathbf{a}_0\,\mathbf{m} + \mathbf{a}_1\,\mathbf{k}$ and selecting the damping ratio in the first and second modes to be 3% of critical, the constants a_0 and a_1 are:

$$a_0 = 0.59335 \quad \text{and} \quad a_1 = 0.00132.$$

The resulting damping matrix is :

$$\mathbf{c} = \begin{bmatrix} 1.383 & -0.790 & 0.000 \\ -0.790 & 3.259 & -1.579 \\ 0.000 & -1.579 & 5.135 \end{bmatrix}$$

and the resulting damping ratio in the third mode is 3.68% .

b) Using the modified Newmark explicit method

- *Selection of* Δt and parameters α and ρ :

Selecting $\Delta t = 0.012$s, $\alpha = 0.5$ and $\rho = -0.005$ the stability condition (2.29) is satisfied:

$$\sqrt{-\frac{\rho}{\alpha}} = 0.1 \quad ; \quad \frac{1 + \sqrt{1 - (1+\alpha)\rho}}{1+\alpha} = 1.336$$

$$\begin{aligned} \omega_1 \Delta t &= \Omega_1 = 0.1740 \\ \omega_2 \Delta t &= \Omega_2 = 0.3732 \\ \omega_3 \Delta t &= \Omega_3 = 0.5532 \end{aligned}$$

Hence,

$$0.1 \leqslant \begin{Bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{Bmatrix} \leqslant 1.336$$

- *Numerical damping*

For $\alpha = 0.5$ and $\rho = -0.005$ the resulting damping ratios given by Eqs. (2.30) and (2.31) are :

i) $\Omega_1 = 0.1740$

$$\begin{array}{l} A = 0.9798 \\ B = 0.1728 \end{array} \rightarrow \bar{\Omega}_1 = 0.1746$$
$$\bar{\xi}_1 = 2.92\%$$

ii) $\Omega_2 = 0.3732$

$$\begin{array}{l} A = 0.8980 \\ B = 0.3590 \end{array} \rightarrow \bar{\Omega}_2 = 0.3803$$
$$\bar{\xi}_2 = 8.79\%$$

iii) $\Omega_3 = 0.5532$

$$\begin{array}{l} A = 0.7730 \\ B = 0.5045 \end{array} \rightarrow \bar{\Omega}_3 = 0.5782$$
$$\bar{\xi}_3 = 13.85\%$$

An analytic linear dynamic analysis can be perfomed by the PSEUDO command if instead of the measured restoring forces $\mathbf{r}_2$ the product of the linear elastic stiffness $\mathbf{k}$ by the displacements $\mathbf{d}_2$ is input at each step. The same response can be obtained by the STEP command if the same algorithm, the Newmark explicit is employed. A comparison of the two procedures is shown at the end and the plot of the first DOF displacement response is also included.

INPUT

```
omega
start
load k nr=3 nc=3
800.   -800.   0.
-800.  1800.  -1200.
0.    -1200.   3000.
load m r=1 c=3
1.0  1.5  2.0
eigen k phi m t=4
sqrel m
print m
return
```

OUTPUT

```
**input
**freq
**submit omega
start
load k nr=3 nc=3
matrix name = k       number of rows =    3 number of columns =    3
load m r=1 c=3
matrix name = m       number of rows =    1 number of columns =    3
eigen k phi m t=4
  4 figure tolerance specified
     8 rotations performed
sqrel m
print m

print of array named "m      "

col# =          1           2           3
row   1    14.52167    31.04770    46.09948


return
**stop
```

INPUT

```
begin
start
load m nr=3 nc=3
1.0  0.0  0.0
0.0  1.5  0.0
0.0  0.0  2.0
load c nr=3 nc=3
1.38298    -0.78960    0.000
-0.78960    3.25883   -1.57920
0.000      -1.57920    5.13471
load pp r=3 c=1
1.
2.
2.
load f r=1 c=41
0.  500.  0.  0.  0.  0.  0.
init i=1 d1,v1,a1,d2,c,m,pp,f t=0.012
return
```

OUTPUT

```
**input
**exis
**submit begin
start
load m nr=3 nc=3
matrix name = m       number of rows =    3 number of columns =    3
load c nr=3 nc=3
matrix name = c       number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp      number of rows =    3 number of columns =    1
load f r=1 c=41
matrix name = f       number of rows =    1 number of columns =   41
init i=1 d1,v1,a1,d2,c,m,pp,f t=0.012

 newmark explicit method for pseudodynamic test

return
**p d2

print of array named "d2      "

col# =             1
row   1      0.00000
row   2      0.00000
row   3      0.00000

**zero r2 r=3 c=1
**pseudo r2
**p d2

print of array named "d2      "

col# =             1
row   1 0.71856e-01
row   2 0.95434e-01
row   3 0.71353e-01

**load r2 r=3 c=1
**-14.55
**42.61
**99.04
**pseudo r2
**p d2

print of array named "d2      "

col# =             1
row   1      0.14548
row   2      0.18566
row   3      0.13438

**load r2 r=3 c=1
**-24.51
**84.15
**178.43
**pseudo r2
**p d2

print of array named "d2      "

col# =             1
row   1      0.22220
row   2      0.26677
row   3      0.18365

**load r2 r=3 c=1
**-27.17
**124.81
**228.62
**pseudo r2
**p d2

print of array named "d2      "

col# =             1
row   1      0.30223
row   2      0.33495
row   3      0.21521

**stop
```

INPUT

```
begin
start
load m nr=3 nc=3
1.0  0.0  0.0
0.0  1.5  0.0
0.0  0.0  2.0
load pp r=3 c=1
1.
2.
2.
load f r=1 c=41
0.  500.  0.  0.  0.  0.  0.
init i=2 d1,v1,a1,d2,r1,m,pp,f  t=0.012 p=0.5,-0.005
return
```

OUTPUT

```
**input
**exib
**submit begin
start
load m nr=3 nc=3
matrix name = m        number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp       number of rows =    3 number of columns =    1
load f r=1 c=41
matrix name = f        number of rows =    1 number of columns =   41
init i=2 d1,v1,a1,d2,r1,m,pp,f  t=0.012 p=0.5,-0.005

 modified newmark explicit method for pseudodynamic test
 alfha=     0.5000 ro=    -0.0050

return
**p d2

print of array named "d2     "

col# =           1
row    1     0.00000
row    2     0.00000
row    3     0.00000


**zero r2 r=3 c=1
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row    1 0.72000e-01
row    2 0.96000e-01
row    3 0.72000e-01


**load r2 r=3 c=1
**-14.6
**41.8
**98.9
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row    1     0.14751
row    2     0.18646
row    3     0.13368

**load r2 r=3 c=1
**-24.17
**85.16
**175.27
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row    1     0.22757
row    2     0.26712
row    3     0.18030


**load r2 r=3 c=1
**-24.75
**126.11
**217.36
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row    1     0.31184
row    2     0.33410
row    3     0.20998


**stop
```

INPUT

```
1step
start
load m nr=3 nc=3
1.0  0.0  0 0
0.0  1.5  0.0
0.0  0.0  2.0
load c nr=3 nc=3
1.38296    -0.78960   0.000
-0.78960    3.25883  -1.57920
0.000      -1.57920   5.13471
load k nr=3 nc=3
600.   -600.   0.
-600.  1800.   -1200.
0.     -1200.   3000.
load pp r=3 c=1
1.
2.
2.
load f r=1 c=41
0.  500.  0.  0.  0.  0.  0.
init i=1 d1,v1,a1,d2,c,m,pp,f t=0.012
p d2
return
c****************************************
2step
mult k d2 r2
pseudo r2
p d2
return
```

OUTPUT

```
**input
**exic
**submit 1step
start
load m nr=3 nc=3
matrix name = m       number of rows =    3 number of columns =    3
load c nr=3 nc=3
matrix name = c       number of rows =    3 number of columns =    3
load k nr=3 nc=3
matrix name = k       number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp      number of rows =    3 number of columns =    1
load f r=1 c=41
matrix name = f       number of rows =    1 number of columns =   41
init i=1 d1,v1,a1,d2,c,m,pp,f t=0.012

 newmark explicit method for pseudodynamic test

p d2

print of array named "d2     "

col# =          1
row    1     0.00000
row    2     0.00000
row    3     0.00000


return
**submit 2step
mult k d2 r2
pseudo r2

pseudodynamic test - step:     1

p d2

print of array named "d2     "

col# =          1
row    1 0.71856e-01
row    2 0.95434e-01
row    3 0.71353e-01


return
**submit 2step
mult k d2 r2
pseudo r2

pseudodynamic test - step:     2

p d2

print of array named "d2     "

col# =          1
row    1     0.14542
row    2     0.18562
row    3     0.13435


return
**submit 2step
mult k d2 r2
pseudo r2

pseudodynamic test - step:     3

p d2

print of array named "d2     "

col# =          1
row    1     0.22203
row    2     0.26654
row    3     0.18345


return
**stop
```

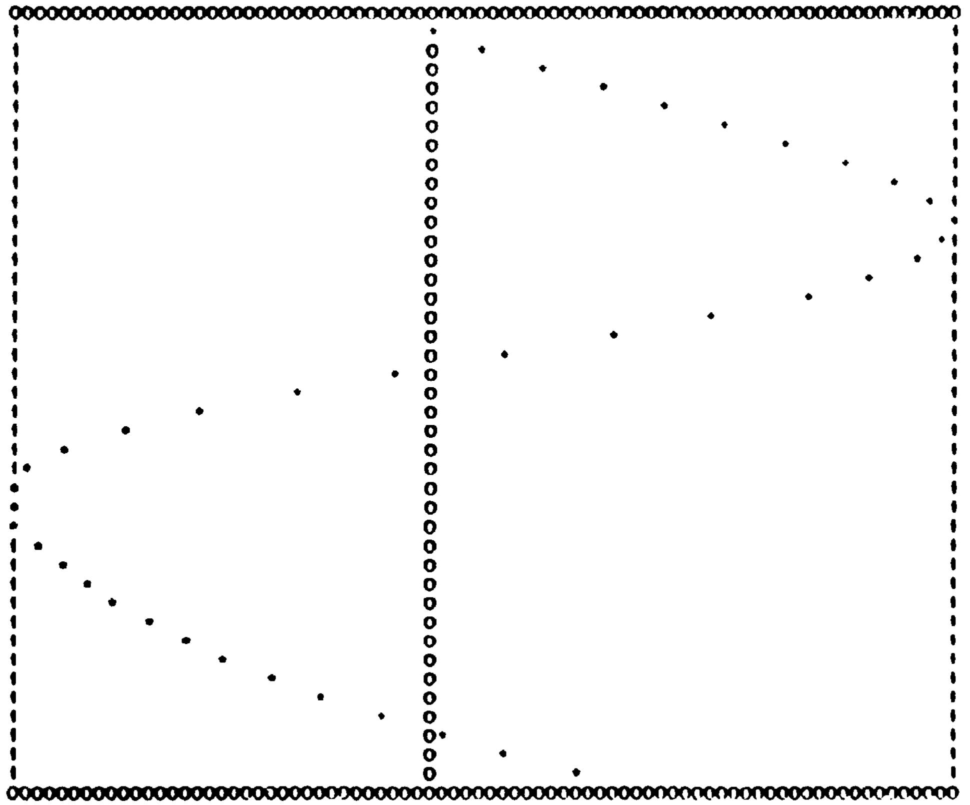
INPUT

```
1step
start
load m nr=3 nc=3
1.0  0.0  0.0
0.0  1.5  0.0
0.0  0.0  2.0
load c nr=3 nc=3
1.38296   -0.78960   0.000
-0.78960   3.25883  -1.57920
0.000     -1.57920   5.13471
load k nr=3 nc=3
600.   -600.   0.
-600.  1800.  -1200.
0.    -1200.   3000.
load pp r=3 c=1
1.
2.
2.
load f r=1 c=41
0.  500.  0.  0.  0.  0.  0.
zero u0 r=3 c=3
step k m c u0 dis pp f t=0.012 1=1,40 p=0.5,0.,1.
p dis
return
c*********************************************
2step
plot dis n=1 r=1 s=*
return
```

OUTPUT

```
**input
**exid
**submit 1step
start
load m nr=3 nc=3
matrix name = m       number of rows =    3 number of columns =    3
load c nr=3 nc=3
matrix name = c       number of rows =    3 numbdr of columns =    3
load k nr=3 nc=3
matrix name = k       number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp      number of rows =    3 number of columns =    1
load f r=1 c=41
matrix name = f       number of rows =    1 number of columns =   41
zero u0 r=3 c=',
step k m c u0 dis pp f t=0.012 1=1,40 p=0.5,0.,1.

 integration parameters for step by step integration
gamma =     0.5000 beta =     0.0000theta =     1.0000

p dis

print of array named "dis    "

col# =        1         2         3         4         5         6
row   1   0.00000   0.07186   0.14542   0.22203   0.30188   0.38377
row   2   0.00000   0.09543   0.18562   0.26654   0.33439   0.38591
row   3   0.00000   0.07135   0.13435   0.18345   0.21541   0.22958

col# =        7         8         9        10        11        12
row   1   0.46489   0.54094   0.60645   0.65534   0.68169   0.68054
row   2   0.41879   0.43210   0.42659   0.40469   0.37020   0.32770
row   3   0.22772   0.21333   0.19079   0.16452   0.13823   0.11447

col# =       13        14        15        16        17        18
row   1   0.64877   0.58563   0.49316   0.37608   0.24140   0.09768
row   2   0.28173   0.23600   0.19278   0.15256   0.11420   0.07543
row   3   0.09455   0.07866   0.06619   0.05604   0.04688   0.03739

col# =       19        20        21        22        23        24
row   1  -0.04596  -0.18085  -0.29965  -0.39710  -0.47032  -0.51883
row   2   0.03356  -0.01360  -0.06710  -0.12635  -0.18909  -0.25168
row   3   0.02627   0.01235  -0.00541  -0.02771  -0.05477  -0.08602

col# =       25        26        27        28        29        30
row   1  -0.54421  -0.54958  -0.53887  -0.51618  -0.48516  -0.44858
row   2  -0.30965  -0.35841  -0.39397  -0.41340  -0.41515  -0.39907
row   3  -0.12001  -0.15428  -0.18558  --0.21024  -0.22470  -0.22618

col# =       31        32        33        34        35        36
row   1  -0.40813  -0.36438  -0.31694  -0.26473  -0.20639  -0.14066
row   2  -0.36624  -0.31872  -0.25925  -0.19095  -0.11717  -0.04138
row   3  -0.21323  -0.18610  -0.14683  -0.09898  -0.04711   0.00394

col# =       37        38        39        40
row   1  -0.06681   0.01507   0.10372   0.19669
row   2   0.03302   0.10273   0.16482   0.21692
row   3   0.04978   0.08712   0.11415   0.13064
return
**stop
```

```
**input
**ex1d
**submit 2step
plot dis n:1 r:1 s:*
symbol row
    *    1
one space=    0.153904e-01
minimum=   -0.549576e+00
maximum=    0.881687e+00
```



```
return
**stop
```

4.2 EXAMPLE 2

The same structure described in the first example is subjected now to the El Centro 1940 (NS) earthquake.

For seismic loading the equations of motion (2.1) become :

$$\mathbf{m}\,\mathbf{a}(t) + \mathbf{c}\,\mathbf{v}(t) + \mathbf{k}\,\mathbf{d}(t) = -\mathbf{m}\,\mathbf{p}\,\mathbf{a}_g(t)$$

where $\mathbf{a}_g(t)$ is the ground acceleration time history from the El Centro earthquake and p is the load distribution vector which in this case is equal to the unity vector. Hence,

$$\mathbf{f}(t) = -\mathbf{m}\,\mathbf{a}_g(t)$$

A dynamic analysis of the 3 DOF structure for the first 10 seconds of seismic excitation is shown in this example. The physical properties of the frame including the viscous damping are the same adopted in the first example.

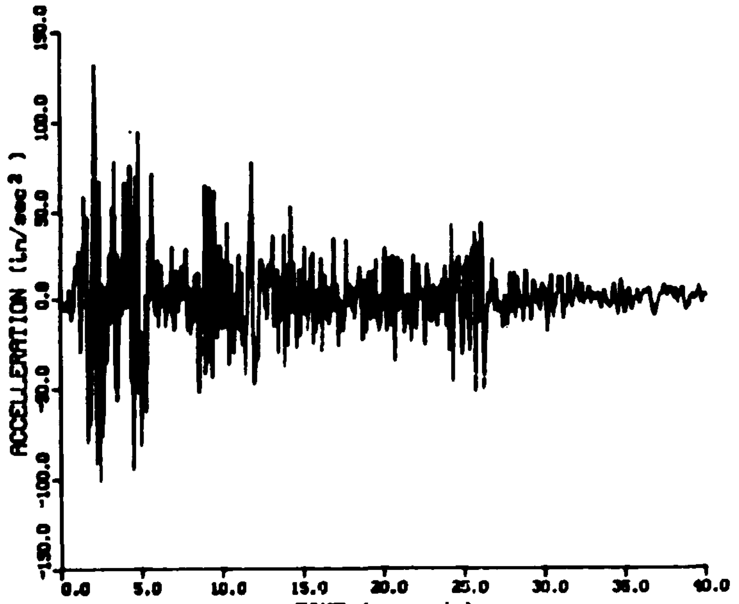
First a nonlinear analysis is performed using the PDT method by means of the INIT and PSEUDO commands. The algorithm chosen is the Newmark explicit and the time interval Δt selected is 0.01s.

Next an analytic linear analysis using the Newmark explicit method by means of the STEP command is shown. The plot of the first DOF displacement response for the first 2.5s using the PLOT command is included at the end.

It should be pointed out that in this example the disk file EC40.ACC contains a (2 x 2001) matrix which has on the first row the time values at equal intervals 0.02s ranging from 0 to 40 s and on the second row the El Centro accelerogram record $\left(\frac{a_g}{g}\times 10^3\right)$ at the corresponding time values. The commands DEFINE and RFILE are used to put the matrix ACC in the DATABASE file. Next the command FUNG is utilized to form the accelerations array at equal intervals 0.01s (ΔT chosen). The accelerations are transformed to in./s^2 by the command SCALE so that the displacement response will be given in inches.

INPUT

```
first
start
load m nr=3 nc=3
1.0  0.0  0.0
0.0  1.5  0.0
0.0  0.0  2.0
load c nr=3 nc=3
1.38296    -0.78980    0.000
-0.78980    3.25883   -1.57920
0.000      -1.57920    5.13471
load pp r=3 c=1
-1.
-1.
-1.
mult m pp p
define acc r nr=2 nc=2001
rfile acc
fung acc f t=0.01 1=1001.1
d acc
load x r=1 c=1
0.386
scale f x
init 1=1 d1,v1,a1,d2,c,m,p,f t=0.01
return
```



Accelerogram from El Centro Earthquake, May 18, 1940
(NS Component)

OUTPUT

```
**input
**ec40
**submit first
start
load m nr=3 nc=3
matrix name = m        number of rows =    3 number of columns =    3
load c nr=3 nc=3
matrix name = c        number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp       number of rows =    3 number of columns =    1
mult m pp p
define acc r nr=2 nc=2001
rfile acc
fung acc f t=0.01 1=1001.1
d acc
load x r=1 c=1
matrix name = x        number of rows =    1 number of columns =    1
scale f x
init 1=1 d1,v1,a1,d2,c,m,p,f t=0.01

 newmark explicit method for pseudodynamic test

return
**p d2

print of array named "d2    "

col# =          1
row    1    0.00000
row    2    0.00000
row    3    0.00000


**zero r2 r=3 c=1
**pseudo r2
**p d2

print of array named "d2    "

col# =          1
row    1 0.24131e-03
row    2 0.24130e-03
row    3 0.23990e-03


**load r2 r=3 c=1
**0.
**0.0015
**0.413
**pseudo r2
**p d2

print of array named "d2    "

col# =          1
row    1 0.98380e-03
row    2 0.98353e-03
row    3 0.93499e-03
```

OUTPUT

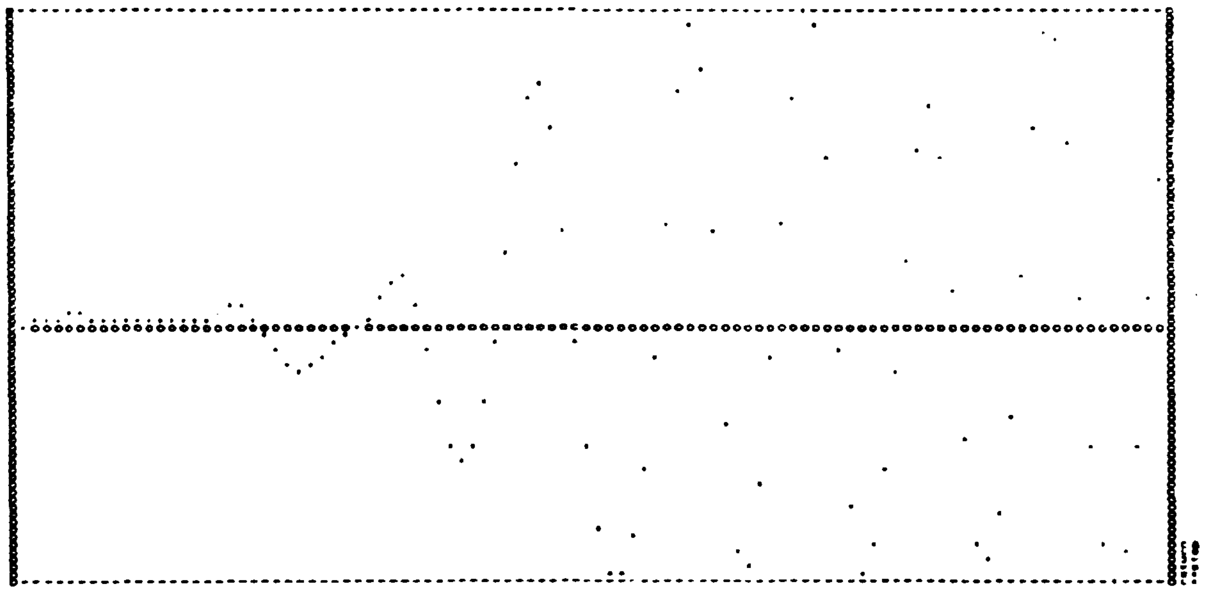
```
**load r2 r=3 c=1
**o.
**0.032
**1.426
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row   1 0.21316e-02
row   2 0.21283e-02
row   3 0.19946e-02


**load r2 r=3 c=1
**0.002
**0.157
**3.192
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row   1 0.37087e-02
row   2 0.36903e-02
row   3 0.32929e-02


**load r2 r=3 c=1
**0.01
**0.432
**5.15
**pseudo r2
**p d2

print of array named "d2     "

col# =           1
row   1 0.56782e-02
row   2 0.56135e-02
row   3 0.47166e-02


**stop
```

INPUT

```
1stp
start
load m nr=3 nc=3
1.0  0.0  0.0
0.0  1.5  0.0
0.0  0.0  2.0
load c nr=3 nc=3
1.38296   -0.78960   0.000
-0.78960   3.25883  -1.57920
0.000     -1.57920   5.13471
load k r=3 c=3
600.   -600.    0.
-600.  1800.  -1200.
0.    -1200.   3000.
load pp r=3 c=1
-1.
-1.
-1.
mult m pp p
define acc r nr=2 nc=2001
rfile acc
fung acc f t=0.01 1=1001,1
d acc
load x r=1 c=1
0.386
scale f x
zero u0 r=3 c=3
step k,m,c,u0,dis,p,f t=0.01 1=4.250 p=0.5,0.0.1.0
dupsm dis di nr=1 nc=100 1=1.1
d dis
return
c*****************************************************
2stp
plot di n=1 r=1 s=*
return
```

OUTPUT

```
**input
**ec40
**submit 1stp
start
load m nr=3 nc=3
matrix name = m        number of rows =    3 number of columns =    3
load c nr=3 nc=3
matrix name = c        number of rows =    3 number of columns =    3
load k r=3 c=3
matrix name = k        number of rows =    3 number of columns =    3
load pp r=3 c=1
matrix name = pp       number of rows =    3 number of columns =    1
mult m pp p
define acc r nr=2 nc=2001
rfile acc
fung acc f t=0.01 1=1001,1
d acc
load x r=1 c=1
matrix name = x        number of rows =    1 number of columns =    1
scale f x
zero u0 r=3 c=3
step k,m,c,u0,dis,p,f t=0.01 1=4.250 p=0.5,0.0.1.0

 integration parameters for step by step integration
gamma =     0.5000 beta =     0.0000theta =     1.0000

dupsm dis di nr=1 nc=100 1=1.1
d dis
return
**stop


**input
**ec40
**submit 2stp
plot di n=1 r=1 s=*
symbol row
   *    1
one space=    0.426389e-01
min'mum=   -0.148819e+01
maximum=    0.179501e+01
```



List of References

[1] Seible, F., Latham, C.T. and Kurkchubasche, A., "CALSD - Instructional Computer Programs for Structural Engineering - User Informatior Manual", Department of Applied Mechanics and Engineering Sciences, University of California, San Diego, February 1987

[2] Okamoto, S. et al, "Techniques for Large Scale Testing at BRI Large Scale Structure Test Laboratory", Building Research Institute, Ministry of Construction, May 1983

[3] Shing, P.B. and Mahin, S.A., "Pseudodynamic Test Method for Seismic Performance Evaluation : Theory and Implementation", Earthquake Engineering Research Center, University of California, Berkeley Report No. UCB/EERC-84/01, January 1984

[4] Mc Clamroch, N.H., Serakos, J. and Hanson, R.D., "Design and Analysis of the Pseudodynamic Test Method", UMEE81R3, University of Michigan, Ann Arbor, 1981

[5] Shing, P.B. and Mahin, S.A., "Experimental Error Propagation in Pseudodynamic Testing", Earthquake Engineering Research Center, University of California, Berkeley Report No. UCB/EERC-83/12, June 1983

[6] U.S./Japan Coordinated Project on Masonry Research, U.S. Research Plan, TCCMAR (Technical Coordinating Committee on Masonry Research), August 1984

[7] Clough, R.W. and Penzien, J., "Dynamics of Structures", Mc Graw Hill, 1975

[8] Nakashima, M., "Stability and Accuracy of Integration Techniques in Pseudodynamic Testing", Building Research Institute, Ministry of Construction, March 1984

[9] Bathe, K. and Wilson, E.L., "Numerical Methods in Finite Element Analysis", Prentice Hall, 1976