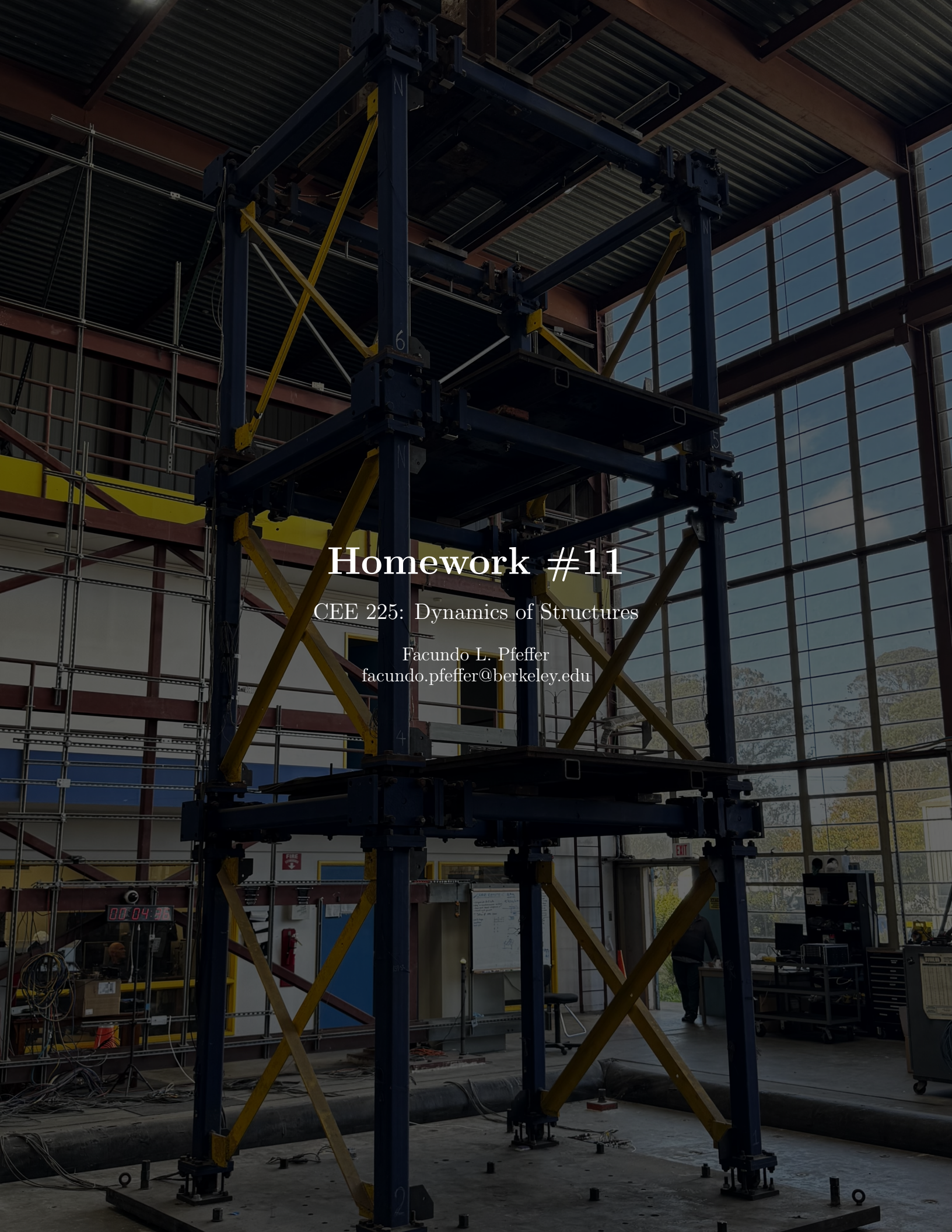


Homework #11

CEE 225: Dynamics of Structures

Facundo L. Pfeffer
facundo.pfeffer@berkeley.edu

Contents

0 Introduction

This report documents the dynamic characterization and seismic response analysis of the three–story shear building tested on the UC Berkeley Richmond Field Station facilities [1, 2, 3] as part of the final coursework of CEE225–Dynamics of Structures [4, 5].[1] The overall objective is twofold: first, to identify and interpret the fundamental modal properties of the structure from the measured responses; and second, to use these properties in a modal framework to predict and rationalize its behavior under earthquake excitation.

The work begins by extracting the modal parameters of the system from free–vibration tests. Natural frequencies, mode shapes, and modal damping ratios are obtained from the roof and floor acceleration records and then processed into a mass–orthonormal modal matrix consistent with the known floor masses. The resulting modal basis provides the starting point for all subsequent analyses, including the construction of effective modal masses and heights and the definition of modal participation factors under uniform ground motion.

Building on this experimental identification, the equations of motion of the 3DOF shear building subjected to horizontal ground acceleration are formulated in modal coordinates. The effective inertia forces, floor displacements, story shears, and base overturning moments are expressed in terms of the modal generalized coordinates and their pseudo–accelerations. These expressions make explicit how each physical response quantity decomposes into contributions from individual modes.

The third part of the report focuses on the history response to the 100% Loma Prieta at Palo Alto record. The uncoupled modal equations are integrated in time using a Python implementation of Newmark's method for SDOF systems, and the resulting modal responses are recombined to obtain floor displacements, floor accelerations, base shear, and base moment. These computed responses are then compared directly with the measured floor accelerations from the shake–table experiment, providing a detailed check on the adequacy of the identified modal model.

Finally, the peak structural response is estimated using modal response spectrum analysis (RSA). Spectrum–interpolated pseudo–accelerations are obtained for each mode at its identified period and damping ratio, and are combined with the modal static base shears through the SRSS rule to estimate the total peak base shear. The comparison with the peak base shear from the direct time–history analysis shows a modest discrepancy of about five percent, consistent with the approximate nature of SRSS and the assumption of statistically independent modal peaks for a system with well–separated natural frequencies. The appendices collect the theoretical background on MDOF modal analysis, the detailed procedures for mode–shape and damping identification, and the Python scripts used to perform the history and spectral response calculations.

Records of the experiments can be found in the subsequent pages:

[1] All numerical values, figures, and scripts used in this report are based on the experimental data set and auxiliary files distributed for Homework #11, including the free–vibration records, the 100% Loma Prieta at Palo Alto ground motion, and the associated design response spectrum.

Figure 0.1: Three–story steel test frame used for dynamic testing at the Richmond Field Station. Photograph taken by the author.

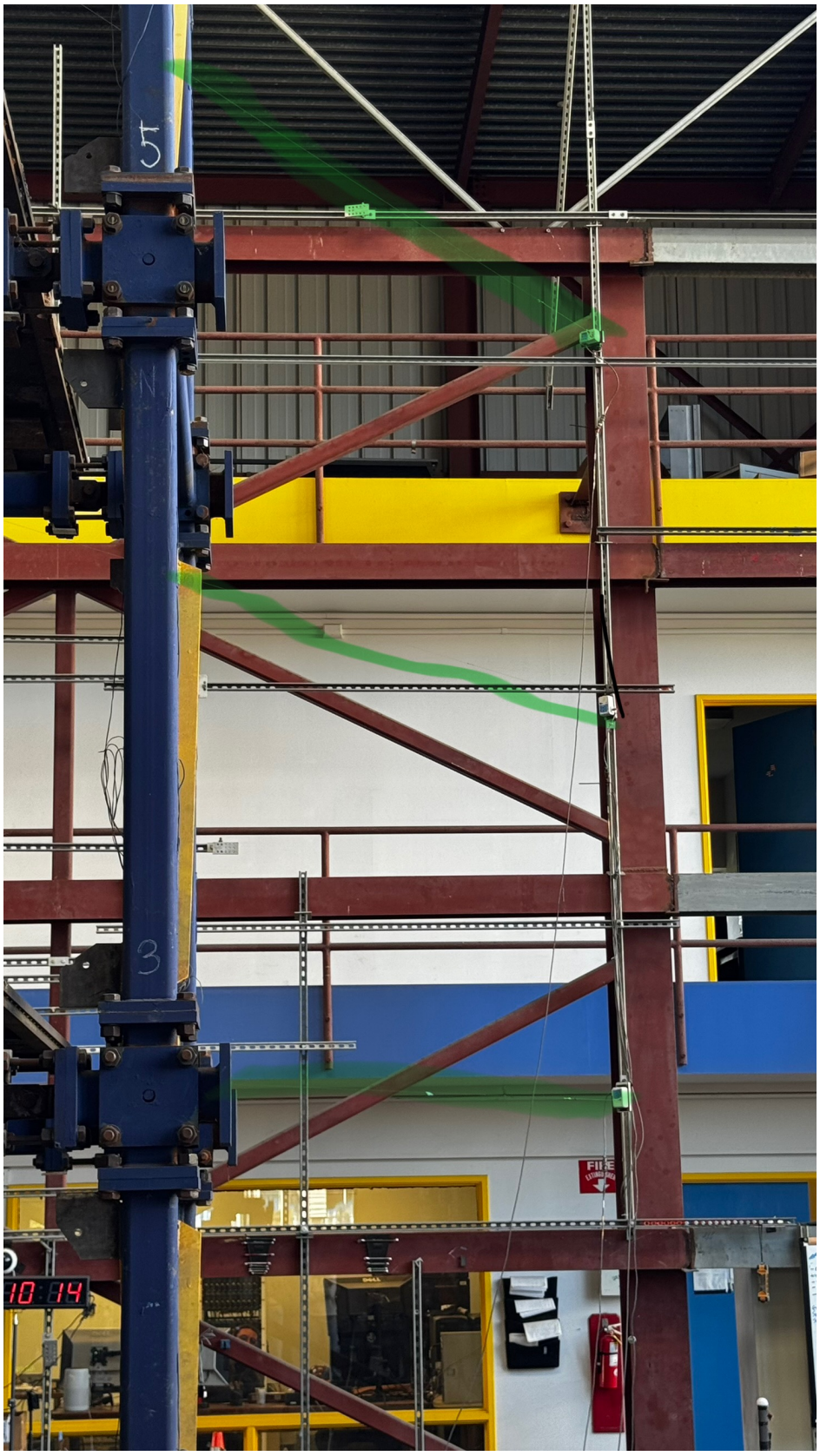
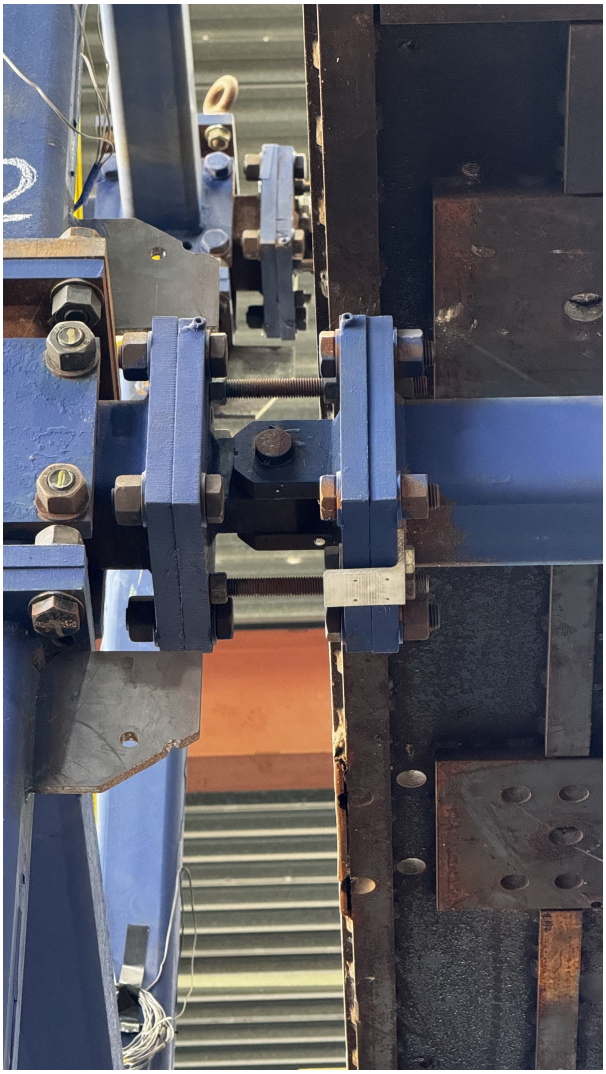


Figure 0.2: Accelerometers (highlighted in green) attached to each story to measure total accelerations.

Figure 0.3: Close–up view of a floor accelerometer and its mounting arrangement.



(a) Beam–column connection detail at an interior joint.

(b) Beam–column connection detail at an exterior joint.

Figure 0.4: Details of the steel beam–column connections for the three–story test frame.

1 P1: Modal Properties of the System

Assignment 1

Using items (i) through (iv), determine:

(a) The frequency of each mode.

(b) The mode shapes. (Hint: the mode shapes can be approximated by using the relative amplitude of the free vibration responses of each floor for each mode from items (ii) through (iv). Try plotting them on the same axes.)

(c) The modal damping in each mode.

1.1 Natural Frequencies of the Structure

The Fourier transform of the roof acceleration record is provided in **??**. The modal natural frequencies are identified by locating the dominant acceleration peaks in the spectrum shown in **Fig. 1.1**.

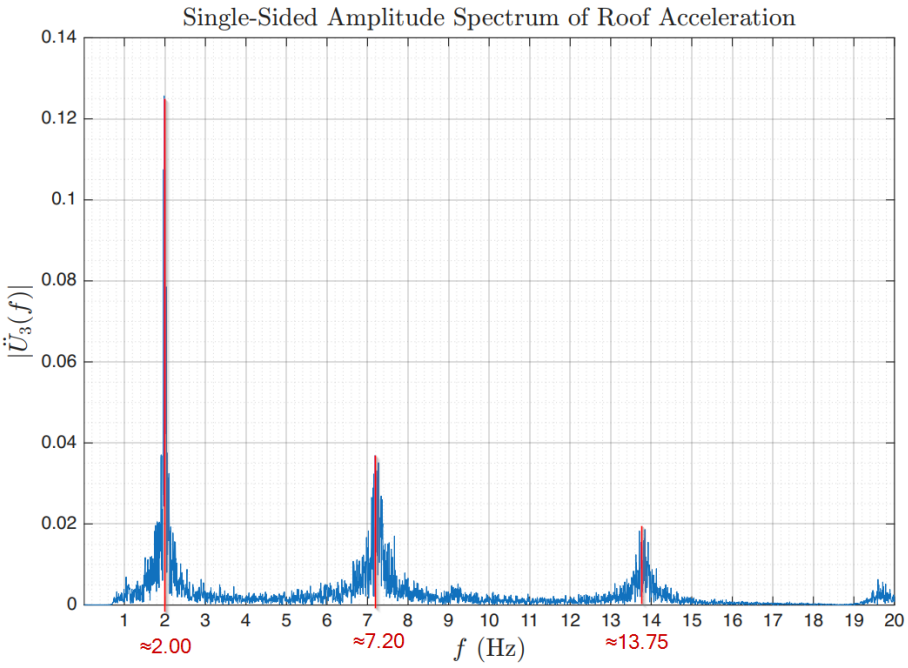


Figure 1.1: Single–sided amplitude spectrum of roof acceleration.

The observed frequencies are:

$$f_1 = 2.00\,\text{Hz}, \qquad f_2 = 7.20\,\text{Hz}, \qquad f_3 = 13.75\,\text{Hz},$$

with corresponding natural circular frequencies $\omega_n = 2\pi f_n$:

$$\omega_{n1} = 4\pi\,\frac{\text{rad}}{\text{s}}, \qquad \omega_{n2} = 14.4\pi\,\frac{\text{rad}}{\text{s}}, \qquad \omega_{n3} = 27.50\pi\,\frac{\text{rad}}{\text{s}}.$$

1.2 Mass Matrix

According to the provided information, and noting that each degree of freedom is located at the mass center of each floor, the mass matrix of the three–story shear building takes the form[6, 4, 5]:

$$\boldsymbol{m} = \begin{bmatrix} 1180 & 0 & 0 \\ 0 & 1180 & 0 \\ 0 & 0 & 910 \end{bmatrix} \text{kg} = \begin{bmatrix} 2.600 & 0 & 0 \\ 0 & 2.600 & 0 \\ 0 & 0 & 2.062 \end{bmatrix} \text{klb},$$

where each diagonal entry is the tributary mass associated with the corresponding floor.

1.3 Mode Shape Extraction

The mode shapes of the three–story MDOF structure are obtained from the measured floor accelerations by treating the response as single–mode– dominated free vibration in narrow frequency bands, as discussed in Section B. In a time interval where mode n dominates the response, the floor accelerations satisfy the approximate relation:

$$a_j^{(n)}(t) \approx \phi_j^{(n)}\, \ddot{q}_n(t), \quad j = 1, 2, 3, \tag{1.1}$$

where $a_j^{(n)}(t)$ is the acceleration at floor j, $\phi_j^{(n)}$ is the jth component of the nth mode shape, and $\ddot{q}_n(t)$ is the corresponding modal acceleration.[2]

Directly from (1.1), the experimental mode shapes identified from accelerations determine only *relative* amplitudes across floors; any common scaling of a mode shape and the associated modal coordinate leaves the physical response unchanged. Moreover, measurement noise and slight departures from ideal single–mode behavior introduce small inconsistencies, so that the resulting modal matrix Φ_{exp} does not automatically satisfy the mass–orthogonality and normalization conditions assumed in the theoretical development (Sections A.1 and A.2), namely

$$\Phi^{\mathsf{T}} \boldsymbol{m} \Phi = \boldsymbol{M} = \text{diag}(M_1, \ldots, M_N), \qquad (\boldsymbol{\phi}^{(n)})^{\mathsf{T}} \boldsymbol{m}\, \boldsymbol{\phi}^{(n)} = M_n.$$

In this experiment, the floor masses are known from the physical configuration and are regarded as less uncertain than the measured accelerations. The experimental mode shapes are therefore first extracted from the measured accelerations in a purely relative sense (essentially "shape only") and then brought into consistency with the known mass matrix by a dedicated post–processing step. This post–processing has two goals:

1. Removing the arbitrary scaling of each experimental mode by enforcing a consistent mass–based normalization, typically $(\boldsymbol{\phi}^{(n)})^{\mathsf{T}} \boldsymbol{m}\, \boldsymbol{\phi}^{(n)} = 1$.
2. Reducing residual lack of orthogonality between modes so that $\Phi^{\mathsf{T}} \boldsymbol{m} \Phi$ is as close to the identity as possible, in line with the modal framework used in Section A.2.

[2] The interpretation in (1.1) assumes that, in the frequency band of interest, the response is governed by a single mode and that the system behaves approximately linearly during the measurement interval. Under these conditions, the measured accelerations at each floor differ only by the components of the underlying mode shape, up to a scalar modal factor.

In practice, the experimentally identified modes are first assembled into a roof–normalized modal matrix:

$$\boldsymbol{\Phi}_{\text{exp}} = \begin{bmatrix} 0.3089 & -0.9694 & 1.0000 \\ 0.7034 & -0.6746 & -0.9225 \\ 1.0000 & 1.0000 & 0.4605 \end{bmatrix},$$

where each column corresponds to Modes 1, 2, and 3, respectively. Using the known floor masses, these shapes are then mass–normalized and mass–orthogonalized so that the final modal matrix satisfies, within numerical tolerance,

$$\Phi^{\mathsf{T}} \boldsymbol{m} \Phi \approx I, \qquad (\boldsymbol{\phi}^{(n)})^{\mathsf{T}} \boldsymbol{m}\, \boldsymbol{\phi}^{(n)} \approx 1.$$

The resulting mass–orthonormal modal matrix used in all subsequent analyses is

$$\boldsymbol{\Phi} = \boldsymbol{\Phi}_{\text{mass-norm}} = \begin{bmatrix} 0.771 & -1.916 & 2.051 \\ 1.755 & -1.331 & -1.903 \\ 2.495 & 1.982 & 0.914 \end{bmatrix} \times 10^{-2}.$$

For brevity, the subscript "mass-norm" is dropped in the remainder of the report and $\boldsymbol{\Phi}$ always denotes this mass–orthonormal version of the experimental mode shapes. A detailed description of the extraction, filtering, and mass–based post–processing is provided in Section B.

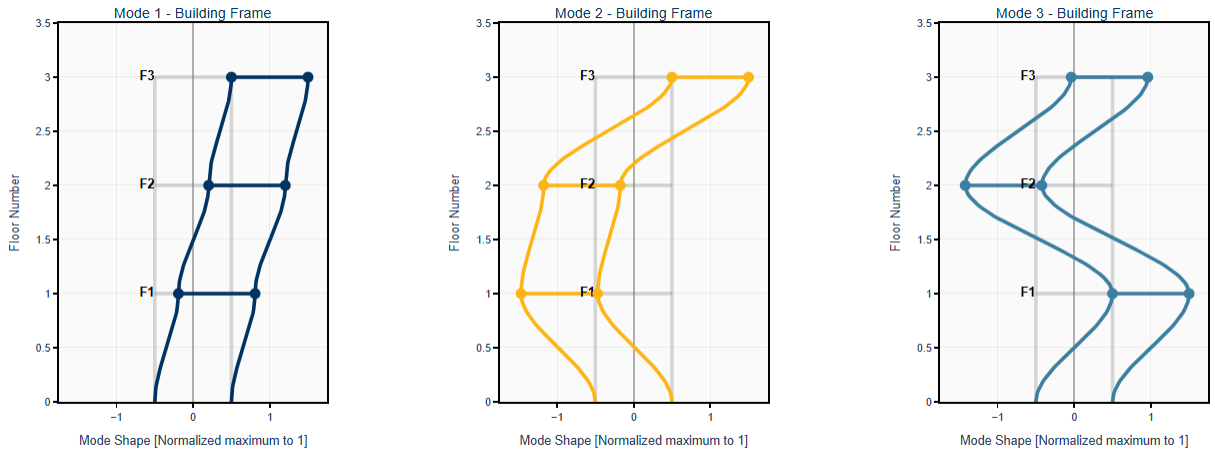


Figure 1.2: Identified mode shapes for the three–story structure. An interactive, Python–generated visualization of these mode shapes is available at https://facundo-pfeffer.github.io/UCBerkeley-SEMM-MS-Codebook/CEE225_Dynamics/highlighted_htmls/step1_mode_shapes.html.

1.4 Modal Damping Ratios

The procedure for obtaining the damping ratios of each mode is detailed in Section C. For a given mode n, the procedure first produces one damping estimate $\zeta_{n,j}$ per floor j by:

- Detecting a common decay window from the combined–modal signal $a^{(n)}_{\text{comb}}(t)$.
- Extracting peaks for each floor within that same window and computing multiple logarithmic decrements δ_i over several cycles.

- Averaging the valid δ_i (cf. **Eq. (C.3)**) and converting the result to a damping ratio via **Eq. (C.4)**.

The quality checks built into the algorithm act only at the *time–sample* level: noisy segments or peaks that do not follow an approximately exponential decay are discarded, but all instrumented floors remain in the analysis. Thus, for each mode n and each floor j, a single floor–wise estimate $\zeta_{n,j}$ is obtained from the peaks that pass these checks.

Let N_f denote the number of instrumented floors ($N_f = 3$ in this experiment). The *modal* damping ratio for mode n is taken as the arithmetic mean of the per–floor estimates:

$$\bar{\zeta}_n = \frac{1}{N_f} \sum_{j=1}^{N_f} \zeta_{n,j},$$

and the dispersion across floors is quantified by the standard deviation and the coefficient of variation:

$$\sigma_{\zeta,n} = \sqrt{\frac{1}{N_f - 1} \sum_{j=1}^{N_f} \left(\zeta_{n,j} - \bar{\zeta}_n\right)^2}, \qquad \mathrm{CoV}_n = \frac{\sigma_{\zeta,n}}{\bar{\zeta}_n}.$$

A low value of CoV_n (typically below 0.3) indicates that the per–floor estimates are mutually consistent and supports the interpretation of $\bar{\zeta}_n$ as a global, mode–dependent structural damping ratio.

The resulting modal damping ratios for the three identified modes are:

$$\zeta_1 = 1.13\%, \qquad \zeta_2 = 1.57\%, \qquad \zeta_3 = 0.93\%.$$

The statistics from which these damping ratios were obtained are summarized in **Fig. 1.3**–**Fig. 1.5**. Interactive plots illustrating the peak detection and decay fitting used in this damping identification are available at https://facundo-pfeffer.github.io/UCBerkeley-SEMM-MS-Codebook/CEE225_Dynamics/highlighted_htmls/step2_damping.html.

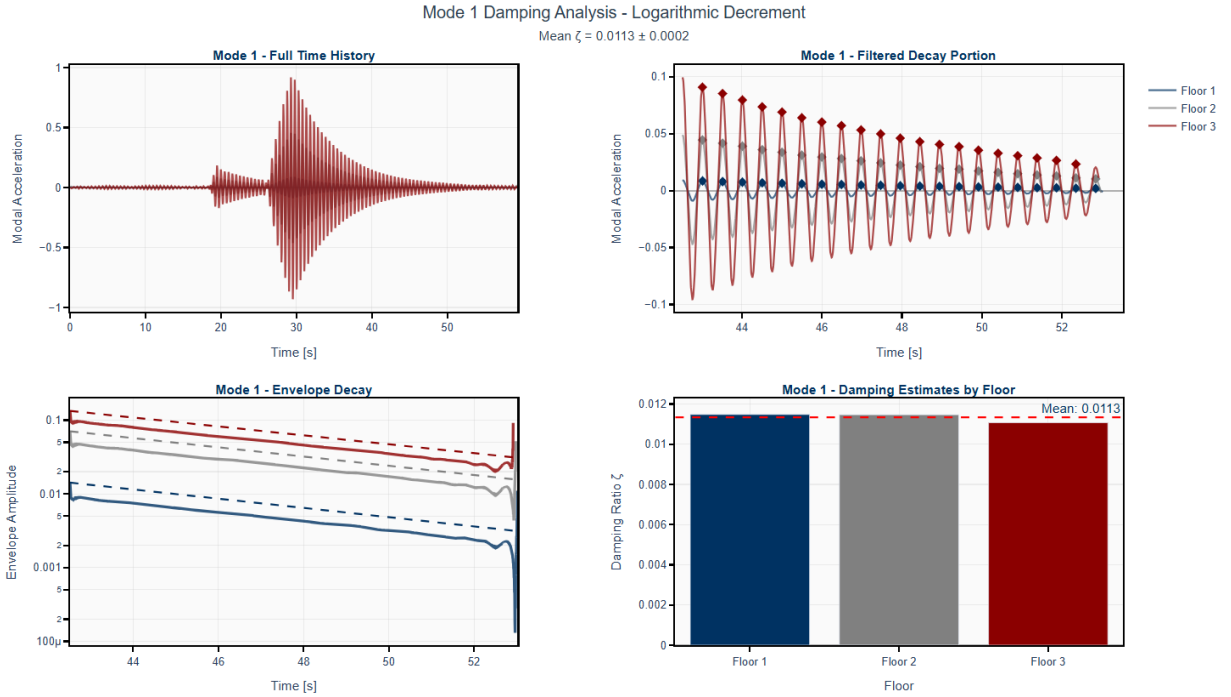


Figure 1.3: Statistics for determining the Mode 1 damping ratio.

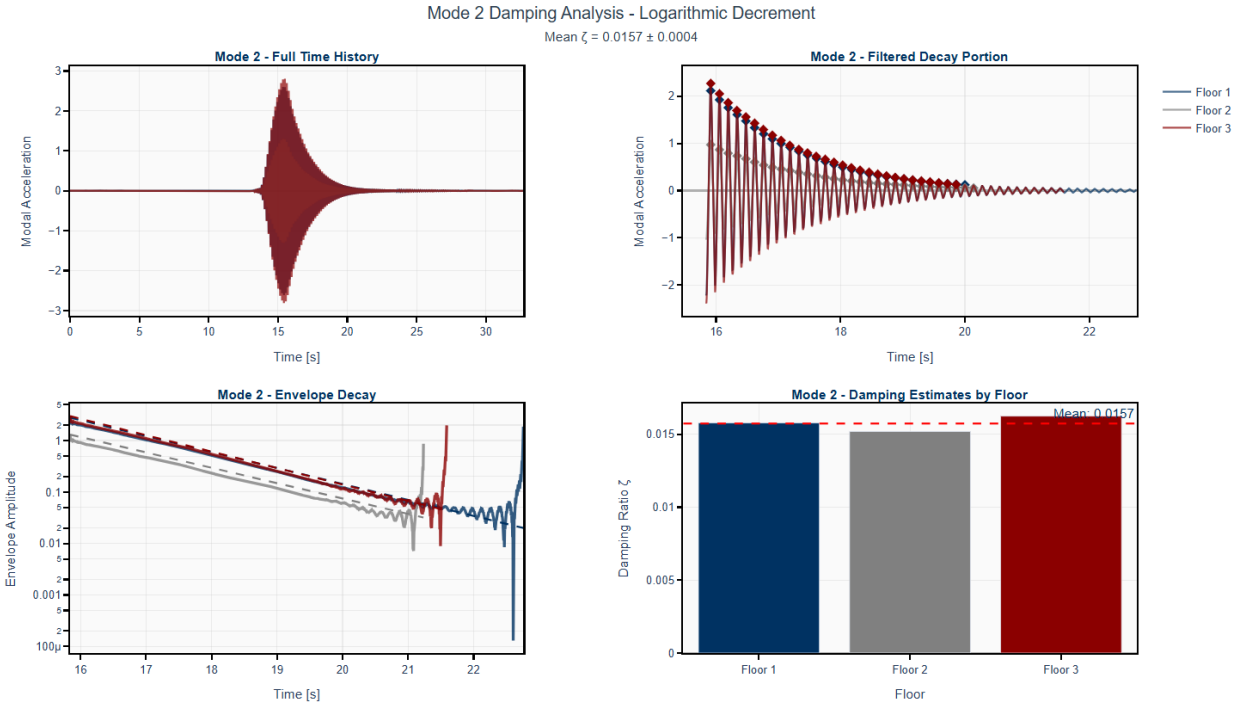


Figure 1.4: Statistics for determining the Mode 2 damping ratio.

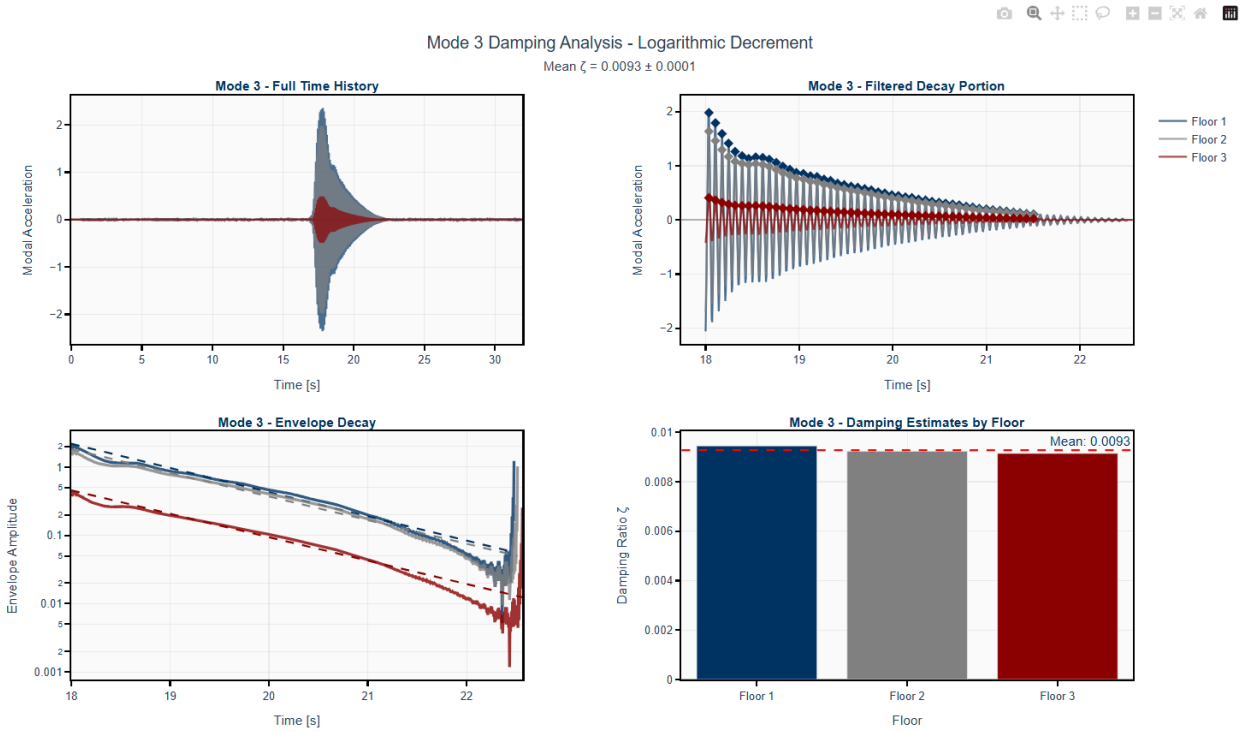


Figure 1.5: Statistics for determining the Mode 3 damping ratio.

2 P2: Modal Expansion of Ground Motion

Assignment 2

For the 3DOF system excited by an *arbitrary* horizontal ground motion $\ddot{u}_g(t)$, determine:

(a) the modal expansion of the effective earthquake forces.

(b) the floor displacements in terms of $D_n(t)$.

(c) the story shear response in terms of $A_n(t)$.

(d) the base overturning moment in terms of $A_n(t)$.

(e) the effective modal masses and effective modal heights.

Note: Leave the answers in terms of $A_n(t)$ and $D_n(t)$. Do not use the actual ground motion provided. Specify units clearly.

Following the theoretical introduction detailed in Section A.6, the 3DOF shear building subjected to spatially uniform ground acceleration $\ddot{u}_g(t)$ is governed by:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{c}\,\dot{\boldsymbol{u}}(t) + \boldsymbol{k}\,\boldsymbol{u}(t) = \boldsymbol{p}_{\text{eff}}(t) = -\,\boldsymbol{m}\,\boldsymbol{\iota}\,\ddot{u}_g(t), \qquad \ddot{u}_g(t)\,[\text{m/s}^2],\ \boldsymbol{u}(t)\,[\text{m}],\ \boldsymbol{p}_{\text{eff}}(t)\,[\text{N}].$$

Here $\boldsymbol{\iota} = [1\ 1\ 1]^{\mathsf{T}}$ is the influence vector, $\boldsymbol{m}\,[\text{kg}]$ is the mass matrix, and $\boldsymbol{k}\,[\text{N/m}]$ is the lateral stiffness matrix.

The mode shapes are collected as $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_3]$ and are taken mass–orthonormal so that $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi} = I$ and $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi} = \boldsymbol{\Lambda} = \text{diag}(\omega_1^2, \omega_2^2, \omega_3^2)$ with $\omega_n\,[\text{rad/s}]$. For each mode n,

$$L_n = \boldsymbol{\phi}_n^{\mathsf{T}}\boldsymbol{m}\,\boldsymbol{\iota}\,[\text{kg}], \qquad M_n = \boldsymbol{\phi}_n^{\mathsf{T}}\boldsymbol{m}\,\boldsymbol{\phi}_n\ = \mathbf{1}[\text{kg}], \qquad \Gamma_n = \frac{L_n}{M_n}\,[-],$$

are the participation numerator, the modal mass, and the participation factor, respectively (Section A.6). For the particular structure, the modal expansions for each mode are:

$$\Gamma_1 = 52.5113 \qquad \Gamma_2 = -20.2784 \qquad \Gamma_3 = 10.0638$$

2.1 Modal expansion of effective earthquake forces

Introducing the modal inertia–force patterns

$$\boldsymbol{s}_n = \Gamma_n\,\boldsymbol{m}\,\boldsymbol{\phi}_n, \qquad \boldsymbol{s}_n\,[\text{kg}] \equiv \frac{\text{N}}{\text{m/s}^2}, \qquad n = 1, 2, 3,$$

the effective earthquake forces expand as

$$\boldsymbol{m}\,\boldsymbol{\iota} = \sum_{n=1}^{3} \boldsymbol{s}_n, \qquad \boldsymbol{p}_{\text{eff}}(t) = -\,\boldsymbol{m}\,\boldsymbol{\iota}\,\ddot{u}_g(t) = -\sum_{n=1}^{3} \boldsymbol{s}_n\,\ddot{u}_g(t), \qquad \boldsymbol{p}_{\text{eff}}(t)\,[\text{N}].$$

2.2 Floor displacements in terms of $D_n(t)$

Each mode behaves as an equivalent SDF system with generalized coordinate $D_n(t)$ [m]:

$$\ddot{D}_n(t) + 2\zeta_n\omega_n\dot{D}_n(t) + \omega_n^2 D_n(t) = -\ddot{u}_g(t), \qquad \zeta_n\,[-],\ \omega_n\,[\text{rad/s}].$$

The modal coordinate in structural space is

$$q_n(t) = \Gamma_n D_n(t)\,[\text{m}],$$

and the relative floor displacements follow from

$$\boldsymbol{u}(t) = \sum_{n=1}^{3} \Gamma_n \boldsymbol{\phi}_n D_n(t) \qquad [\text{m}],$$

so that, for floor j,

$$u_j(t) = \sum_{n=1}^{3} \Gamma_n \phi_{jn} D_n(t) \qquad [\text{m}], \quad j = 1, 2, 3.$$

Evaluating the expression:

$$\boldsymbol{u} = \begin{bmatrix} 0.4049 \\ 0.9215 \\ 1.3101 \end{bmatrix} D_1(t) + \begin{bmatrix} 0.9885 \\ 0.2699 \\ -0.40119 \end{bmatrix} D_2(t) + \begin{bmatrix} 0.2064 \\ -0.1915 \\ 0.092 \end{bmatrix} D_3(t) \qquad [\text{m}]$$

2.3 Story shear response in terms of $A_n(t)$

Let $V_r(t)$ denote the shear force in story r [N]. For mode n, a static analysis of the frame subjected to the lateral force pattern $\boldsymbol{s}_n$ gives the modal static story shears $V_{r,n}^{\text{st}}$ [N] (one value per story and mode). Using the general combination rule from *Section A.6*:

$$r(t) = \sum_{n=1}^{3} r_n^{\text{st}} A_n(t), \qquad A_n(t) = \omega_n^2 D_n(t)\,[\text{m/s}^2],$$

Where the modal static response r_n^{st} is determined by static analysis of the building due to external forces $\boldsymbol{s_n}$, which are obtained using **Eq. (A.29)**. The computed values are:

$$\boldsymbol{s_1} = \begin{bmatrix} 477.737 \\ 1087.456 \\ 1192.2428 \end{bmatrix} \text{kg} \qquad \boldsymbol{s_2} = \begin{bmatrix} 458.470 \\ 318.489 \\ -365.745 \end{bmatrix} \text{kg} \qquad \boldsymbol{s_3} = \begin{bmatrix} 243.562 \\ -225.987 \\ 83.705 \end{bmatrix} \text{kg} \tag{2.1}$$

Equilibrium dictates that the shear for story j (from bottom to top) on this structure for mode n:

$$V_{3,n} = s_3 A_n(t) \qquad V_{2,n} = (s_3 + s_2) A_n(t) \qquad V_{3,n} = (s_3 + s_2 + s_1) A_n(t)$$

Grouping this results in a single vector and combining the contributions of each mode gives:

$$V_{story} = \begin{bmatrix} 2757.5258 \\ 2279.7888 \\ 1192.2428 \end{bmatrix} A_1(t) + \begin{bmatrix} 411.214 \\ -47.256 \\ -365.745 \end{bmatrix} A_2(t) + \begin{bmatrix} 101.28 \\ -142.282 \\ 83.705 \end{bmatrix} A_3(t) \qquad [\text{N}] \tag{2.2}$$

2.4 Base overturning moment in terms of $A_n(t)$

Let $M_b(t)\,[\mathrm{N\,m}]$ denote the base overturning moment, and $M_{b,n}^{\mathrm{st}}\,[\mathrm{N\,m}]$ the static base moment obtained from the same modal lateral pattern $\boldsymbol{s}_n$. Applying the same combination rule to $r(t) = M_b(t)$ produces

$$M_b(t) = \sum_{n=1}^{3} M_{b,n}^{\mathrm{st}}\, A_n(t) \qquad [\mathrm{N\,m}], \qquad A_n(t)\,[\mathrm{m/s^2}].$$

Equilibrium dictates that the base overturning moment can be obtained by multypling the values of $\boldsymbol{s_n}$ obtained in **Eq. (2.1)** by their respective lever arm. In matricial form:

$$M_{b,n}^{\mathrm{st}} = -\boldsymbol{h_n} \cdot \boldsymbol{s_n} \qquad [\mathrm{kg\,m}]$$

Where the minus sign is introduced to preseve counter-clock-wise postive moment direction and $\boldsymbol{h_n^T} = \begin{bmatrix} 2.0828 & 4.1656 & 6.2484 \end{bmatrix}$ m is the corresponding lever arm for each s_n. The resulting base overturning moment is:

$$M_b(t) = -12974.55\, A_1(t) + 3.7252\, A_2(t) - 88.94\, A_3(t) \qquad [\mathrm{Nm}]$$

2.5 Effective modal masses and effective modal heights

2.5.1 Effective Modal Mass

The effective modal mass $M_n^*\,[\mathrm{kg}]$ associated with mode n is

$$M_n^* = \Gamma_n L_n = \Gamma_n^2 M_n = \Gamma_n^2\, \boldsymbol{\phi}_n^{\mathsf{T}} \boldsymbol{m} \boldsymbol{\phi}_n \qquad [\mathrm{kg}],$$

representing the portion of the total mass that participates in mode n under uniform ground motion. Therefore:

$$M_1^* = 2757.436\,\mathrm{kg} \qquad M_2^* = 411.213\,\mathrm{kg} \qquad M_3^* = 101.28\,\mathrm{kg}$$

As expected, the contribution of the first mode of vibration is the most predominant and is expected to govern the earthquake response. Moreover, the sum of the effective modal masses shall satisfy:

$$\sum_{n}^{3} M_n^* - tr(\boldsymbol{m}) = -6.97\,10^{-2}\,\mathrm{kg} \approx 0$$

Which is satisfied with sufficient accuracy.

2.5.2 Effective modal heights

For each mode n, the effective modal height h_n^* is defined so that the static base overturning moment produced by the modal force pattern $\boldsymbol{s}_n$ satisfies

$$M_{b,n}^{\mathrm{st}} = M_n^* h_n^*,$$

where M_n^* is the effective modal mass. With floor heights h_j [m] measured from the base, this gives:

$$M_n^* = \sum_{j=1}^{3} s_{jn}, \qquad M_{b,n}^{\mathrm{st}} = \sum_{j=1}^{3} h_j s_{jn}, \qquad h_n^* = \frac{M_{b,n}^{\mathrm{st}}}{M_n^*}.$$

Using the modal force patterns in **Eq. (2.1)** and $h_1 = 2.0828\,\text{m}$, $h_2 = 4.1656\,\text{m}$, $h_3 = 6.2484\,\text{m}$:

$$M_1^* = 2757.44\,\text{kg}, \quad M_{b,1}^{\text{st}} = 1.2975 \times 10^4\,\text{kg m}, \quad h_1^* = 4.71\,\text{m},$$

$$M_2^* = 411.21\,\text{kg}, \quad M_{b,2}^{\text{st}} = -3.73\,\text{kg m}, \quad h_2^* = -9.1 \times 10^{-3}\,\text{m} \approx 0\,\text{m},$$

$$M_3^* = 101.28\,\text{kg}, \quad M_{b,3}^{\text{st}} = 88.94\,\text{kg m}, \quad h_3^* = 0.88\,\text{m}.$$

The first mode therefore controls both effective modal mass and effective height, whereas the second mode contributes negligible base overturning moment because $|h_2^*| \ll 1\,\text{m}$.

Mode	$\Gamma_n\,[-]$	$M_n^*\,[\text{kg}]$	$\frac{M_n^*}{\sum_j m_j}\,[\%]$	$h_n^*\,[\text{m}]$
1	52.51	2.76×10^3	84.3	4.71
2	-20.28	4.11×10^2	12.6	≈ 0.00
3	10.06	1.01×10^2	3.1	0.88

Table 1: Effective modal parameters for the 3DOF shear building (participation factors, effective modal masses, and effective modal heights).

3 P3: History Response to 100% Loma Prieta at Palo Alto

Assignment 3

For the 100% Loma Prieta at Palo Alto ground motion provided in item (v), determine and plot:

(a) The displacement response for each mode $q_n(t)$. A Matlab code is provided with the input files that integrates the EOM using Newmark's method for an SDOF system. You can also recycle your code from previous HW.

(b) The displacement response of each floor $u_j(t)$ in inches.

(c) The acceleration response of each floor $\ddot{u}_j(t)$ in in/s^2. Also plot the actual measured acceleration response of each floor provided in item (v) and compare the response.

(d) The base shear (kips) as a function of time.

(e) The base overturning moment (kip-ft) as a function of time.

The response of the 3DOF shear building to the 100% Loma Prieta at Palo Alto ground motion is obtained by solving the uncoupled modal equations derived in Section A.6 for each identified mode of vibration. For mode n, the generalized coordinate $D_n(t)$ [m] satisfies:

$$\ddot{D}_n(t) + 2\,\zeta_n\,\omega_n\,\dot{D}_n(t) + \omega_n^2\,D_n(t) = -\,\ddot{u}_g(t), \qquad n = 1, 2, 3, \tag{3.1}$$

where ω_n [rad/s] and ζ_n [$-$] are the natural circular frequency and damping ratio of mode n, and $\ddot{u}_g(t)$ [m/s^2] is the horizontal ground acceleration. Equation **Eq. (3.1)** is integrated in the time domain using Newmark's method for SDOF systems with the same parameters as those used in previous homework. The modal coordinates in structural space are

$$q_n(t) = \Gamma_n\,D_n(t) \qquad \text{[m]},$$

and are converted to inches for post–processing and plotting. Interactive, Python–generated plots summarizing the modal and floor responses discussed in this section are available (link here).

(a) Modal displacement responses $q_n(t)$

For each mode, the code integrates **Eq. (3.1)** over the duration of the ground motion record and computes

$$q_n(t) = \Gamma_n\,D_n(t)\,\text{[m]}, \qquad q_n(t)\,\text{[in]} = 39.3701\,q_n(t)\,\text{[m]}.$$

The resulting histories $q_n(t)$ for modes $n = 1, 2, 3$ are plotted as [in] versus time. Figure 3.1 shows the modal displacement responses for the three modes, highlighting the dominant contribution of the first mode.

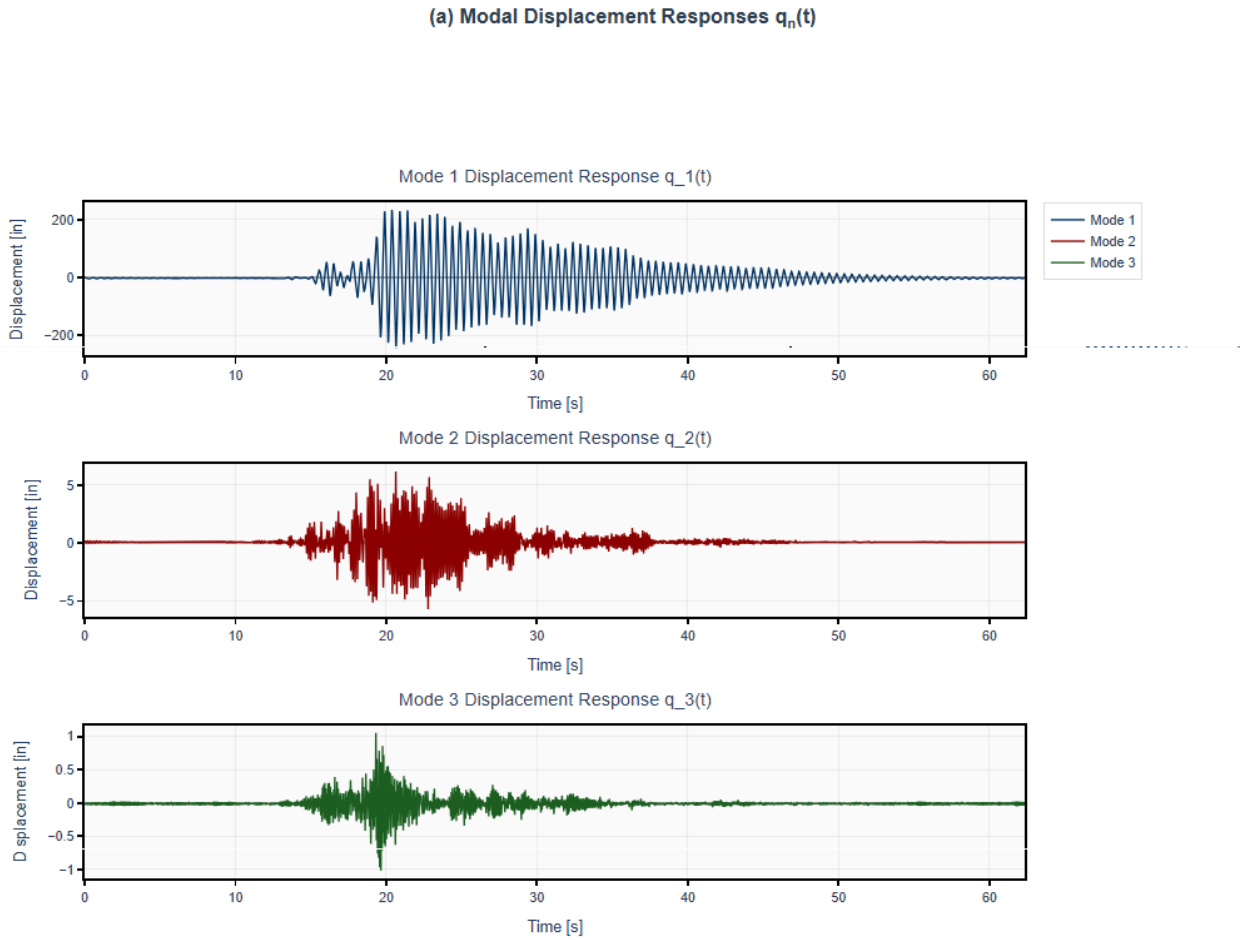


Figure 3.1: Modal displacement responses $q_n(t)$ for the three modes under 100% Loma Prieta at Palo Alto. An interactive version of these curves is available online (link here).

(b) Floor displacement responses $u_j(t)$ in inches

The relative displacement of floor j is obtained by modal superposition using the mass–orthonormal mode shapes:

$$u_j(t) = \sum_{n=1}^{3} \phi_{jn}\, q_n(t) \qquad [\mathrm{m}], \qquad u_j(t)\,[\mathrm{in}] = 39.3701\, u_j(t)\,[\mathrm{m}], \quad j = 1, 2, 3.$$

The script computes $u_j(t)$ at each time step and converts the results to inches. Figure 3.2 presents the displacement histories of Floors 1–3 in [in], illustrating the amplification of motion with height due to modal participation.

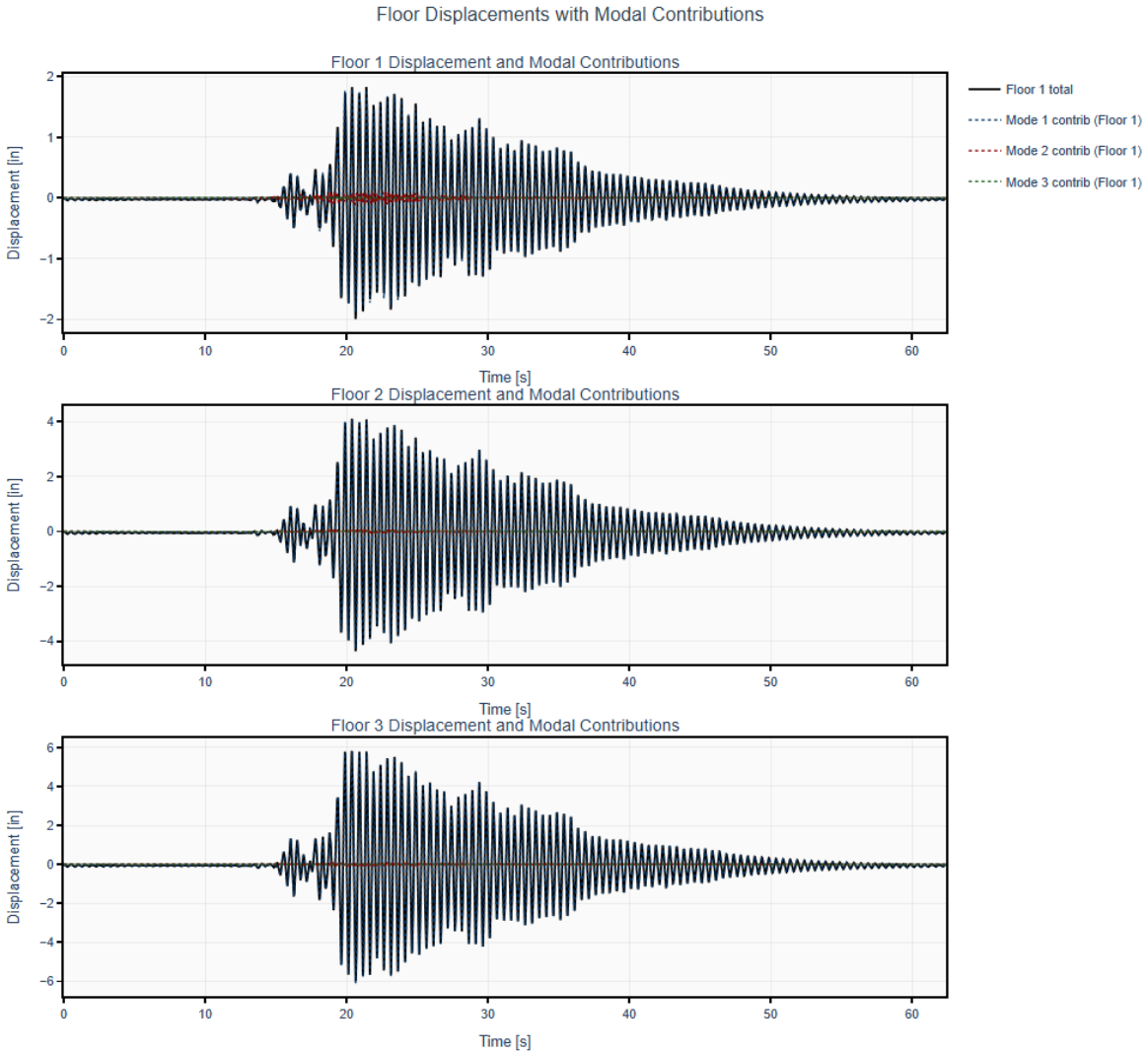


Figure 3.2: Floor displacement responses $u_j(t)$ in inches for all three floors. Interactive plots of these responses are provided in the online material (link here).

Response Statistics

Response	Maximum [in]	Minimum [in]	Time of Max [s]
Floor 1 (u_1)	1.996	-1.996	20.64
Floor 2 (u_2)	4.365	-4.365	20.64
Floor 3 (u_3)	5.974	-5.974	20.65

Note: The left column shows the displacement time history for each floor. The right column shows the scaled mode shape contributions, where each mode's contribution is scaled to match the maximum displacement magnitude for visual comparison.

Figure 3.3: Maximum displacements of the structure.

(c) Floor acceleration responses $\ddot{u}_j(t)$ in in/s^2 and comparison with measurements

The total acceleration at floor j combines the relative modal contribution and the ground acceleration:

$$\ddot{u}_j(t) = \sum_{n=1}^{3} \phi_{jn}\, \Gamma_n\, \ddot{D}_n(t) + \ddot{u}_g(t) \qquad [\text{m/s}^2],$$

$$\ddot{u}_j(t)\,[\text{in/s}^2] = 39.3701\, \ddot{u}_j(t)\,[\text{m/s}^2].$$

The measured floor accelerations from `ground_motion_excitation.csv` (columns `L1AccX_filtered`, `L2AccX_filtered`, `L3AccX_filtered`) are interpreted as being provided in units of g. They are therefore converted to m/s^2 by multiplication by 9.81, and then to in/s^2 for direct comparison with the computed responses.

Figure 3.4 shows the computed total acceleration $\ddot{u}_j(t)$ at each floor in $[\text{in/s}^2]$, overlaid with the corresponding measured acceleration histories. The comparison enables a visual assessment of the agreement in amplitude and phase between the analytical model and the experimental data.

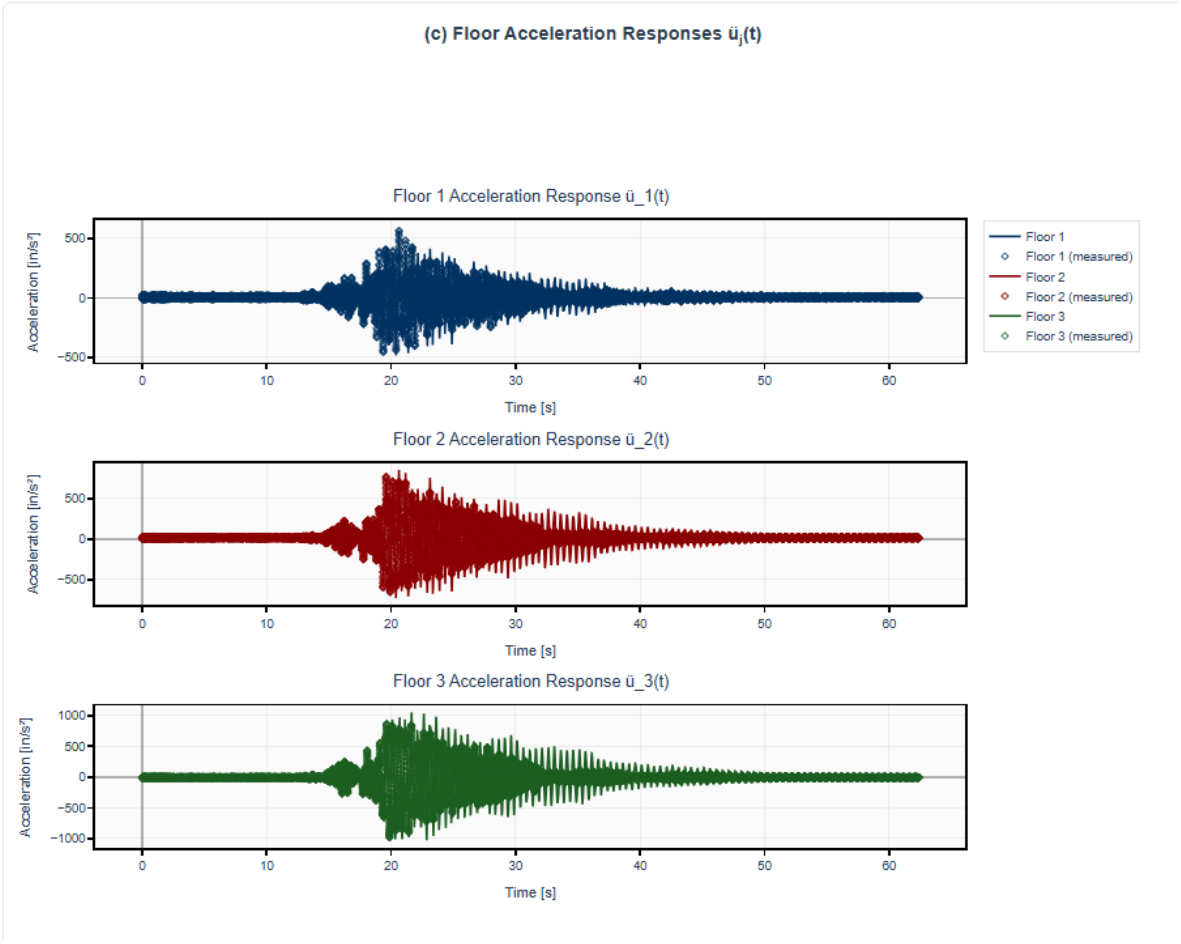


Figure 3.4: Computed and measured floor acceleration responses $\ddot{u}_j(t)$ in in/s^2 for all three floors. Interactive overlays are available in the online material (link here).

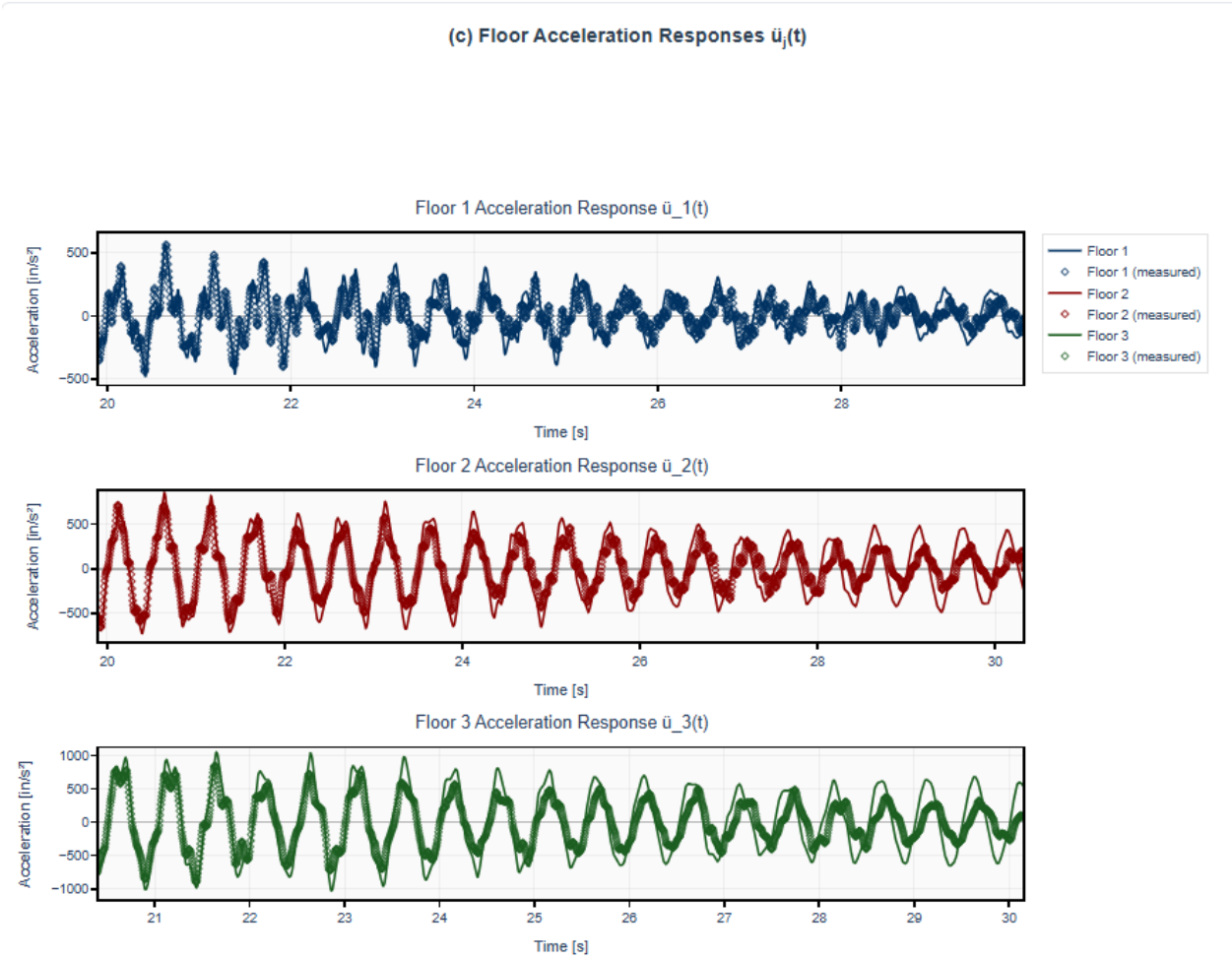


Figure 3.5: Computed and measured floor acceleration responses $\ddot{u}_j(t)$ in in/s^2 for all three floors in the interval $t \in [20, 30]$ s for improved visualization; see **Fig. 3.4** for the full time range. The corresponding interactive zoom controls are included in the online plots (link here).

(d) Base shear $V_b(t)$ in kips

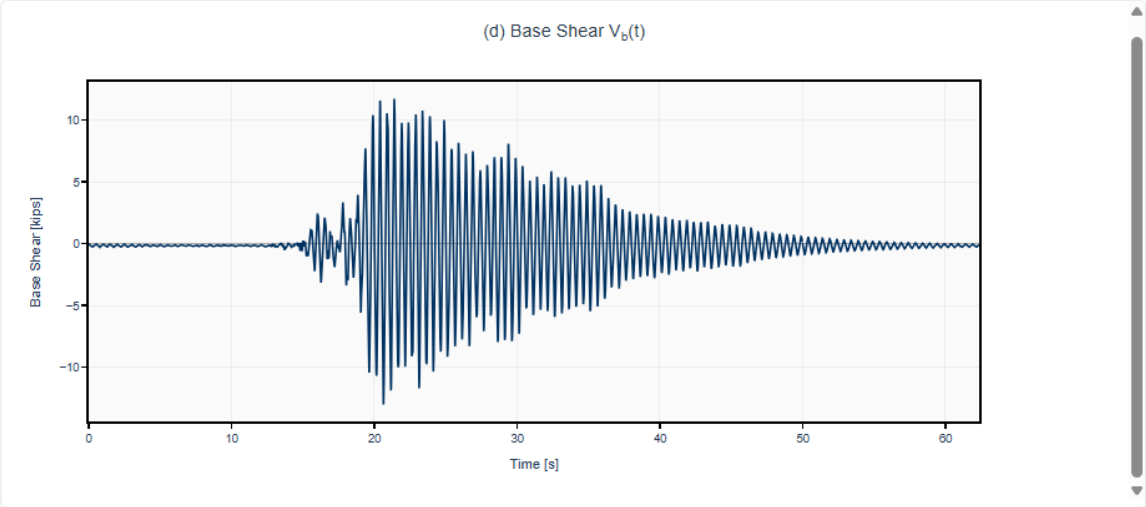
The base shear time history is obtained from the modal responses using the modal shear patterns implemented in the `ModalResponseAnalyzer` class. In terms of the pseudo–accelerations $A_n(t) = \omega_n^2 D_n(t)$ [m/s^2], the base shear can be expressed as:

$$V_b(t) = \sum_{n=1}^{3} V_{b,n}^{\text{st}} A_n(t) \qquad [\text{N}],$$

where $V_{b,n}^{\text{st}}$ [N/(m/s^2)] are the static modal base–shear coefficients. The program computes $V_b(t)$ in Newtons and converts the result to kips via

$$V_b(t)\,[\text{kips}] = \frac{V_b(t)\,[\text{N}]}{4448.22}.$$

The resulting time history of base shear is plotted in Figure 3.6.



Response Statistics

Response	Maximum [kips]	Minimum [kips]	Time of Max [s]
Base Shear (V_b)	13.01	-13.01	20.64

Figure 3.6: Base shear time history $V_b(t)$ in kips. An interactive version of this plot is provided in the online material (link here).

(e) Base overturning moment $M_b(t)$ in kip-ft

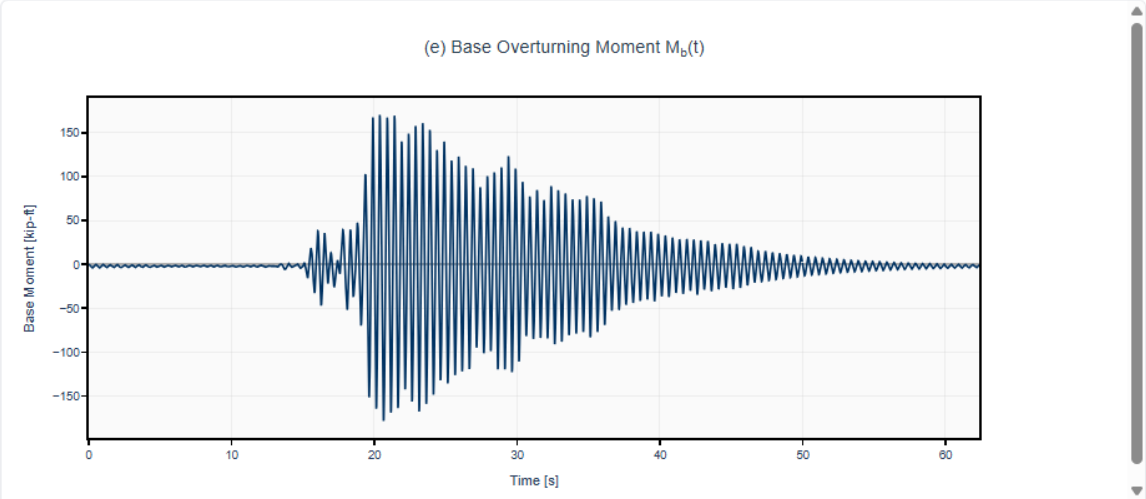
The base overturning moment is computed in an analogous fashion from the modal base–moment coefficients $M_{b,n}^{\text{st}}$ and the pseudo–accelerations $A_n(t)$:

$$M_b(t) = \sum_{n=1}^{3} M_{b,n}^{\text{st}}\, A_n(t) \qquad [\text{N m}],$$

and then converted to kip–ft by

$$M_b(t)\,[\text{kip}-\text{ft}] = 0.000737562\, M_b(t)\,[\text{N m}].$$

Figure 3.7 presents the base overturning moment history in [kip − ft] over the duration of the ground motion record.



Response Statistics

Response	Maximum [kip-ft]	Minimum [kip-ft]	Time of Max [s]
Base Moment (M_b)	178.32	-178.32	20.64

Figure 3.7: Base overturning moment time history $M_b(t)$ in kip-ft. An interactive plot of this response is included in the online material (link here).

4 P4: Peak Structural Response with Response Spectrum

Assignment 4

The peak structural response of the frame is now estimated using response spectrum analysis (RSA) to the 100% Loma Prieta at Palo Alto ground motion.

(a) Determine the spectral ordinates D_n and A_n for the n-th mode SDOF system using the provided response spectrum (item vi).

(b) Using the SRSS method, determine the maximum displacement response of each floor $u_{j,0}$ in inches. Compare this to your solution from Problem 3.

(c) Using the SRSS method, determine the maximum total base shear (kips). Compare this to your solution from Problem 3.

The objective of this section is to characterize the peak structural response of the 3DOF frame using the given linear response spectra. This approach is standard in practice for MDOF systems, and the resulting peak quantities are compared with those obtained from the direct response history analysis in the previous problem.

4.1 Spectral ordinates from the design spectrum

The response spectrum of item (vi) is first plotted in terms of pseudo- acceleration for several damping ratios (0–5%). For each mode n, with natural period T_n and damping ratio ζ_n, the corresponding spectral ordinates are obtained in two interpolation stages:

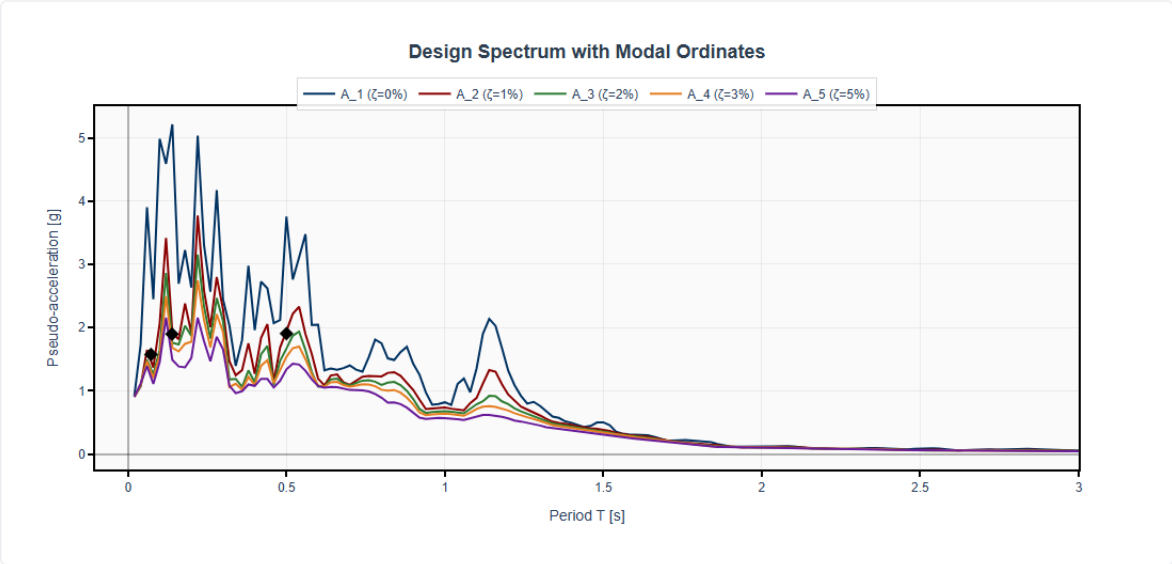
1. For each available damping curve, the value of pseudo-acceleration at period T_n is obtained by linear interpolation in T. This produces $A_{n,0}(T_n, \zeta_i)$ for the tabulated damping ratios ζ_i.
2. The modal damping ratio ζ_n is then located between two successive curves, and a linear interpolation in ζ is performed to obtain the modal pseudo-acceleration:

$$A_{n,0}^{(n)} = (1-w)\, A_{n,0}(T_n, \zeta_{\text{low}}) + w\, A_{n,0}(T_n, \zeta_{\text{high}}), \qquad w = \frac{\zeta_n - \zeta_{\text{low}}}{\zeta_{\text{high}} - \zeta_{\text{low}}}.$$

Since the Peak Ground Acceleration (PGA) is $1\,g$, the quantity $A_{n,0}^{(n)}$ is multiplied by g to obtain pseudo-acceleration in m/s^2. The associated spectral displacement for mode n follows from the usual pseudo-acceleration relation:

$$D_n = \frac{A_{n,0}^{(n)}}{\omega_n^2},$$

which is later used to compute modal peak floor displacements and story forces. The resulting design spectrum with interpolated modal ordinates is shown in **Fig. 4.1**; an interactive version of this spectrum, including hoverable modal points and numerical tooltips, is available online (link here).



Modal Spectral Ordinates

Mode	T [s]	ζ (interp)	$A_{n,0}$ [g]	$A_{n,0}$ [in/s²]	$D_{n,0}$ [in]
Mode 1	0.500	1.13% (bracket 1%→2%, w=0.13)	1.905	735.7	4.659
Mode 2	0.139	1.57% (bracket 1%→2%, w=0.57)	1.898	733.0	0.358
Mode 3	0.073	0.93% (bracket 0%→1%, w=0.93)	1.573	607.7	0.081

Figure 4.1: Design spectrum with modal pseudo-acceleration ordinates $A_{n,0}^{(n)}$ obtained by interpolation in period and damping. Black diamond indicate points of intersection.

4.2 Peak Floor Displacements from Modal Spectral Ordinates

Fig. 4.1 ("Modal Spectral Ordinates") reports the modal spectral displacements $D_{0,n}$ [in] obtained from the design spectrum. For each mode n, the peak contribution to the floor displacements is given by

$$\boldsymbol{u}_{0,n} = \Gamma_n \, \boldsymbol{\phi}_n \, D_{0,n} \qquad [\text{in}],$$

where Γ_n is the participation factor, $\boldsymbol{\phi}_n$ the mass–orthonormal mode shape, and $D_{0,n}$ the spectral displacement of mode n.

Using the identified modal data

$$\Gamma_1 = 52.51, \quad \Gamma_2 = -20.28, \quad \Gamma_3 = 10.06,$$

$$\boldsymbol{\Phi} = \begin{bmatrix} 0.771 & -1.916 & 2.051 \\ 1.755 & -1.331 & -1.903 \\ 2.495 & 1.982 & 0.914 \end{bmatrix} \times 10^{-2}, \quad D_{0,1} = 4.659 \text{ in}, \; D_{0,2} = 0.358 \text{ in}, \; D_{0,3} = 0.081 \text{ in},$$

the modal scaling vectors $\Gamma_n \boldsymbol{\phi}_n$ become

$$\Gamma_1 \boldsymbol{\phi}_1 = 52.51 \begin{bmatrix} 0.00771 \\ 0.01755 \\ 0.02495 \end{bmatrix} \approx \begin{bmatrix} 0.4049 \\ 0.9215 \\ 1.3101 \end{bmatrix},$$

$$\Gamma_2\boldsymbol{\phi}_2 = -20.28\begin{bmatrix}-0.01916\\-0.01331\\0.01982\end{bmatrix} \approx \begin{bmatrix}0.3881\\0.2699\\-0.4012\end{bmatrix},$$

$$\Gamma_3\boldsymbol{\phi}_3 = 10.06\begin{bmatrix}0.02051\\-0.01903\\0.00914\end{bmatrix} \approx \begin{bmatrix}0.2064\\-0.1915\\0.0920\end{bmatrix}.$$

Multiplying by the corresponding spectral displacements gives the peak contribution of each mode (in inches):

$$\boldsymbol{u}_{0,1} = D_{0,1}\,\Gamma_1\boldsymbol{\phi}_1 = 4.659\begin{bmatrix}0.4049\\0.9215\\1.3101\end{bmatrix} \approx \begin{bmatrix}1.89\\4.29\\6.10\end{bmatrix},$$

$$\boldsymbol{u}_{0,2} = D_{0,2}\,\Gamma_2\boldsymbol{\phi}_2 = 0.358\begin{bmatrix}0.3881\\0.2699\\-0.4012\end{bmatrix} \approx \begin{bmatrix}0.14\\0.10\\-0.14\end{bmatrix},$$

$$\boldsymbol{u}_{0,3} = D_{0,3}\,\Gamma_3\boldsymbol{\phi}_3 = 0.081\begin{bmatrix}0.2064\\-0.1915\\0.0920\end{bmatrix} \approx \begin{bmatrix}0.02\\-0.02\\0.01\end{bmatrix}.$$

The peak floor displacement envelope is then obtained by SRSS combination of the three modal contributions at each floor:

$$u_{0,j} = \sqrt{\sum_{n=1}^{3}\left(\Gamma_n\phi_{jn}D_{0,n}\right)^2} \qquad [\text{in}], \quad j = 1,2,3. \tag{4.1}$$

Numerically,

$$\boldsymbol{u}_{0,\text{SRSS}} = \begin{bmatrix}u_{0,1}\\u_{0,2}\\u_{0,3}\end{bmatrix} \approx \begin{bmatrix}1.89\\4.29\\6.11\end{bmatrix} \text{ in},$$

showing the expected amplification of peak displacement with height and the dominance of the first mode in the overall response.

4.3 SRSS combination for base shear

The design spectrum provides a peak value of modal pseudo-acceleration, which is treated as the maximum generalized acceleration for mode n. The associated peak contribution of mode n to the base shear is approximated by scaling the static pattern by this spectral ordinate:

$$V_{b,n}^{\text{RSA}} \approx V_{b,n}^{\text{st}}\,A_{n,0}^{(n)}.$$

Under the assumption that modal peak responses occur at statistically independent times[7], the total peak base shear is obtained by SRSS combination:

$$V_{b,\text{SRSS}} = \sqrt{\sum_{n=1}^{3}\left(V_{b,n}^{\text{st}}\,A_{n,0}^{(n)}\right)^2} = 11.72\,\text{kips}.$$

In the numerical implementation, the spectrum of item (vi) is read, the modal ordinates $A_{n,0}^{(n)}$ and D_n are extracted by interpolation as described above, the modal static base shears $V_{b,n}^{\mathrm{st}}$ are formed from $\boldsymbol{s}_n$ (see first row of **Eq. (2.2)**), and the SRSS expression is evaluated to obtain the peak base shear. Multiplying the modal static base shear (in kg, interpreted as N/(m/s^2)) by $A_{n,0}^{(n)}$ (in m/s^2) gives Newtons; division by 4448.22 converts to kips:

$$\begin{aligned} V_{b,1}^{\mathrm{RSA}} &= 2757.53 \times 18.698 = 51{,}556 \text{ N} = 11.59 \text{ kips}, \\ V_{b,2}^{\mathrm{RSA}} &= 411.21 \times 18.611 = 7{,}653 \text{ N} = 1.72 \text{ kips}, \\ V_{b,3}^{\mathrm{RSA}} &= 101.28 \times 15.430 = 1{,}563 \text{ N} = 0.35 \text{ kips}. \end{aligned}$$

4.4 SRSS total base shear

$$\begin{aligned} V_{b,\mathrm{SRSS}} &= \sqrt{(11.59)^2 + (1.72)^2 + (0.35)^2} \text{ kips} \\ &= \sqrt{134.3 + 2.96 + 0.12} \text{ kips} \\ &= \sqrt{137.4} \text{ kips} \\ &= 11.72 \text{ kips}. \end{aligned}$$

4.5 Comparison with Time–History Peak Response

The peak floor displacements and base shear obtained from the response–spectrum procedure can be compared directly with the maxima from the time–history analysis of Section 3. For the time–history analysis, the peak responses are

$$u_1^{\max} = 1.996 \text{ in}, \quad u_2^{\max} = 4.365 \text{ in}, \quad u_3^{\max} = 5.974 \text{ in}, \quad V_b^{\max} = 13.01 \text{ kips},$$

whereas the RSA–SRSS procedure of this section gives

$$u_{0,1} = 1.89 \text{ in}, \quad u_{0,2} = 4.29 \text{ in}, \quad u_{0,3} = 6.11 \text{ in}, \quad V_{b,\mathrm{SRSS}} = 11.72 \text{ kips}.$$

Response	Time–history peak [in]	RSA peak [in]	Difference [%]
Floor 1 (u_1)	1.996	1.89	−5.3
Floor 2 (u_2)	4.365	4.29	−1.7
Floor 3 (u_3)	5.974	6.11	+2.3

Table 2: Comparison of peak floor displacements from response spectrum analysis (RSA) and direct time–history analysis.

Response	Time–history peak [kips]	RSA peak [kips]	Difference [%]
Base shear (V_b)	13.01	11.72	−9.9

Table 3: Comparison of peak base shear from response spectrum analysis (RSA) and direct time–history analysis.

The displacement comparison in Table 2 shows that the response–spectrum estimate reproduces the peak floor drifts to within about 5% at all stories, while correctly capturing the increase in peak displacement with height and the dominant role of the first mode. The base–shear comparison in Table 3 indicates a larger discrepancy: the RSA value underestimates the time–history peak by approximately 10%. This difference is consistent with the approximate nature of the SRSS combination, which neglects possible correlation between modal peaks and relies on the design spectrum as an envelope of many compatible ground motions rather than the specific recorded accelerogram used in the time–history analysis. Overall, the RSA procedure provides a close estimate of peak floor displacements and a non-conservative, slightly lower estimate of peak base shear for this well–separated three–mode system.

A Multi-Degree-Of-Freedom Systems – Theoretical Background

A.1 Free Vibration, Eigenvalue Problem, and Modal Decomposition of Undamped MDOF Systems

An N-degree-of-freedom linear structural system with time-invariant properties is characterized by the mass matrix $\boldsymbol{m}$ and stiffness matrix $\boldsymbol{k}$. The equation of motion under externally applied loads $\boldsymbol{p}(t)$ is:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{k}\,\boldsymbol{u}(t) = \boldsymbol{p}(t), \tag{A.1}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^N$ collects the generalized displacements. The matrix $\boldsymbol{m}$ represents the inertial properties and is symmetric positive definite, whereas $\boldsymbol{k}$ represents the elastic resistance and is symmetric positive semidefinite [6, 4, 5].

A.1.1 Free vibration

For free vibration, $\boldsymbol{p}(t) = \boldsymbol{0}$ and **Eq. (A.1)** reduces to:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{k}\,\boldsymbol{u}(t) = \boldsymbol{0}. \tag{A.2}$$

The motion is governed by the balance between inertia $\boldsymbol{m}\ddot{\boldsymbol{u}}(t)$ and elastic restoring forces $\boldsymbol{k}\boldsymbol{u}(t)$. The goal is to determine the natural frequencies at which the system oscillates and the associated mode shapes.

A.1.2 Harmonic motion assumption and the generalized eigenvalue problem

For undamped linear systems, free vibration in a single natural mode is harmonic:

$$\boldsymbol{u}(t) = \boldsymbol{\phi}\left(A\sin(\omega t) + B\cos(\omega t)\right),$$

where $\boldsymbol{\phi}$ is a vector of relative amplitudes and A, B are constants. Differentiation gives:

$$\ddot{\boldsymbol{u}}(t) = -\omega^2\boldsymbol{u}(t).$$

Substitution into **Eq. (A.2)** yields:

$$(\boldsymbol{k} - \omega^2\boldsymbol{m})\,\boldsymbol{\phi} = \boldsymbol{0}.$$

This relation is the *generalized eigenvalue problem*. Nontrivial solutions require:

$$\det\left(\boldsymbol{k} - \omega^2\boldsymbol{m}\right) = 0,$$

whose real, nonnegative roots ω_n^2 define the natural frequencies. Each ω_n^2 is associated with a mode shape $\boldsymbol{\phi}^{(n)}$ satisfying:

$$(\boldsymbol{k} - \omega_n^2\boldsymbol{m})\,\boldsymbol{\phi}^{(n)} = \boldsymbol{0}.$$

The pairs $(\omega_n, \boldsymbol{\phi}^{(n)})$ characterize the free-vibration patterns of the structure. However, the generalized eigenproblem couples $\boldsymbol{k}$ and $\boldsymbol{m}$ and does not correspond to a single operator acting on $\boldsymbol{u}(t)$. The subsequent transformation to a symmetric standard eigenproblem allows systematic use of orthogonality properties and leads directly to modal decoupling.

A.1.3 Mass-normalized coordinates and the symmetric eigenproblem

The introduction of mass-normalized coordinates:

$$\boldsymbol{y}(t) = \boldsymbol{m}^{1/2}\boldsymbol{u}(t),$$

with $\boldsymbol{m}^{1/2}$ the unique symmetric positive-definite square root of $\boldsymbol{m}$, has a specific purpose: in these coordinates the mass matrix becomes the identity and the dynamics are governed by a *single* symmetric matrix.

Substituting $\boldsymbol{u}(t) = \boldsymbol{m}^{-1/2}\boldsymbol{y}(t)$ into **Eq. (A.2)** gives:

$$\ddot{\boldsymbol{y}}(t) + \boldsymbol{A}\,\boldsymbol{y}(t) = \boldsymbol{0}, \qquad \boldsymbol{A} := \boldsymbol{m}^{-1/2}\boldsymbol{k}\boldsymbol{m}^{-1/2}.$$

The matrices $\boldsymbol{k}$ and $\boldsymbol{m}^{1/2}$ are symmetric, so the triple product $\boldsymbol{A}$ is symmetric as well. Once a single real symmetric matrix governs the motion, the spectral theorem applies directly [8].

By the spectral theorem, $\boldsymbol{A}$ possesses real eigenvalues λ_n and an orthonormal basis of eigenvectors $\boldsymbol{\psi}^{(n)}$:

$$\boldsymbol{A}\boldsymbol{\psi}^{(n)} = \lambda_n\boldsymbol{\psi}^{(n)}, \qquad (\boldsymbol{\psi}^{(n)})^{\mathsf{T}}\boldsymbol{\psi}^{(m)} = \delta_{nm}.$$

The symmetry of $\boldsymbol{A}$ guarantees:

- real eigenvalues, with $\lambda_n = \omega_n^2$ linked to the natural frequencies;
- an orthonormal eigenbasis $\{\boldsymbol{\psi}^{(n)}\}$ in the mass-normalized space, which later leads to diagonal modal mass and stiffness matrices.

Thus the symmetric eigenproblem replaces the generalized eigenproblem $(\boldsymbol{k}, \boldsymbol{m})$ by a standard symmetric eigenproblem for $\boldsymbol{A}$, making orthogonality and decoupling transparent.

A.1.4 Construction of the orthogonal matrix Q and decoupling in mass-normalized coordinates

Collecting the mass-normalized eigenvectors as columns gives:

$$\boldsymbol{Q} = [\boldsymbol{\psi}^{(1)}\ \boldsymbol{\psi}^{(2)}\ \cdots\ \boldsymbol{\psi}^{(N)}], \qquad \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{Q} = I.$$

The relation $\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{Q} = I$ expresses compactly the orthonormality of the eigenbasis.

Defining the modal coordinates in mass-normalized space as:

$$\boldsymbol{z}(t) = \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{y}(t),$$

and using the eigendecomposition $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\mathsf{T}}$ with $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$, leads to:

$$\ddot{\boldsymbol{z}}(t) + \boldsymbol{\Lambda}\,\boldsymbol{z}(t) = \boldsymbol{0}.$$

This system is diagonal. The nth component satisfies:

$$\ddot{z}_n(t) + \lambda_n z_n(t) = 0.$$

The passage through the symmetric problem therefore produces a set of uncoupled scalar equations in the mass-normalized coordinates. The orthogonality of $\boldsymbol{Q}$ is the mechanism that achieves this decoupling.

A.1.5 Mapping eigenvectors back to physical displacement coordinates

The physical displacement coordinates are related to the mass-normalized ones through $\boldsymbol{y}(t) = \boldsymbol{m}^{1/2}\boldsymbol{u}(t)$, so the inverse mapping is:

$$\boldsymbol{u}(t) = \boldsymbol{m}^{-1/2}\boldsymbol{y}(t).$$

Applying this mapping to the eigenvectors defines the *physical mode shapes*:

$$\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}.$$

This mapping is invertible because $\boldsymbol{m}^{1/2}$ is invertible. Thus the set $\{\boldsymbol{\phi}^{(n)}\}$ is in one-to-one correspondence with $\{\boldsymbol{\psi}^{(n)}\}$; only the coordinate representation is changed.

Substituting $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$ into the eigenvalue relation shows that $\boldsymbol{\phi}^{(n)}$ satisfies the original generalized eigenproblem:

$$(\boldsymbol{k} - \omega_n^2\boldsymbol{m})\,\boldsymbol{\phi}^{(n)} = \boldsymbol{0}.$$

Solving the symmetric problem for $\boldsymbol{A}$ or solving the generalized problem for $(\boldsymbol{k}, \boldsymbol{m})$ thus produces the same physical mode shapes. The symmetric formulation has the advantage that orthonormality of the $\boldsymbol{\psi}^{(n)}$ vectors is simple to state and use and translates into convenient mass-orthogonality properties for the vectors $\boldsymbol{\phi}^{(n)}$.

A.2 Modal Mass and Stiffness Matrices

The physical mode shapes are assembled into the modal matrix:

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}^{(1)}\ \boldsymbol{\phi}^{(2)}\ \cdots\ \boldsymbol{\phi}^{(N)}].$$

The physical displacements admit a modal superposition:

$$\boldsymbol{u}(t) = \boldsymbol{\Phi}\,\boldsymbol{q}(t),$$

where $\boldsymbol{q}(t)$ collects the generalized (modal) coordinates. This relation is the algebraic statement that any free-vibration response is a linear combination of the mode shapes.

Substituting $\boldsymbol{u}(t) = \boldsymbol{\Phi}\boldsymbol{q}(t)$ into **Eq. (A.2)** and premultiplying by $\boldsymbol{\Phi}^{\mathsf{T}}$ gives:

$$\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi}\,\ddot{\boldsymbol{q}}(t) + \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi}\,\boldsymbol{q}(t) = \boldsymbol{0}.$$

A.2.1 Modal mass and stiffness

The matrices:

$$\boldsymbol{M} := \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi}, \qquad \boldsymbol{K} := \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi} \tag{A.3}$$

are the *modal mass* and *modal stiffness* matrices. Using $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$ and the orthonormality of $\boldsymbol{\psi}^{(n)}$ leads to[3]:

$$\boldsymbol{M} = \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi} = I, \qquad \boldsymbol{K} = \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi} = \boldsymbol{\Lambda}.$$

[3]The normalization $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi} = I$ is not mathematically necessary. Any set of eigenvectors of the generalized problem $(\boldsymbol{k}-\omega^2\boldsymbol{m})\boldsymbol{\phi} = \boldsymbol{0}$ can be used to form $\boldsymbol{\Phi}$, and the resulting modal mass matrix $\boldsymbol{M} = \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi}$ is always diagonal but not necessarily the identity. Choosing *mass-normalized* modes so that $\boldsymbol{M} = I$ is a convenient convention: it places the modal equations in the canonical form $\ddot{q}_n + \omega_n^2 q_n = 0$, makes the eigenvalues appear directly as the squared natural frequencies, and simplifies the transformation of initial conditions. Other normalizations (for example, imposing $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi} = I$ or allowing a general diagonal $\boldsymbol{M}$) are possible but lead to less compact expressions.

Hence, in the modal coordinates $\boldsymbol{q}(t)$, the equations of motion become:

$$\ddot{q}_n(t) + \omega_n^2 q_n(t) = 0, \tag{A.4}$$

with no coupling[4] between different $q_n(t)$. The decoupling is a direct consequence of:

- orthonormality of the eigenvectors $\boldsymbol{\psi}^{(n)}$ of the symmetric matrix $\boldsymbol{A}$;
- the mapping $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$ connecting these vectors to the physical mode shapes;
- the resulting diagonal form of $\boldsymbol{M}$ and $\boldsymbol{K}$.

If the mode shapes $\boldsymbol{\phi}^{(n)}$ are obtained directly from the generalized eigenproblem $(\boldsymbol{k} - \omega^2\boldsymbol{m})\boldsymbol{\phi} = \boldsymbol{0}$, the same modal mass and stiffness matrices arise after enforcing the mass-orthogonality normalization $\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\boldsymbol{\Phi} = I$. The mass-normalized formulation therefore does not introduce new physics; it provides a systematic route to orthogonality and decoupling via the spectral theorem.

A.3 Transformation of Initial Conditions in Modal Analysis

The modal coordinates $\boldsymbol{q}(t)$ follow from the transformation:

$$\boldsymbol{u}(t) = \boldsymbol{\Phi}\,\boldsymbol{q}(t), \tag{A.5}$$

where the columns of $\boldsymbol{\Phi}$ are the physical mode shapes $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$. These mode shapes satisfy the mass-orthogonality relation:

$$\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\,\boldsymbol{\Phi} = I, \tag{A.6}$$

because the vectors $\boldsymbol{\psi}^{(n)}$ form an orthonormal basis in the mass-normalized space. Identity **Eq. (A.6)** is a consequence of the symmetry of $\boldsymbol{A}$ and the definition $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$ and is the key to diagonal modal mass and stiffness matrices.

Premultiplying **Eq. (A.5)** by $\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}$ gives:

$$\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\,\boldsymbol{u}(t) = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\,\boldsymbol{\Phi}\,\boldsymbol{q}(t) = I\,\boldsymbol{q}(t) = \boldsymbol{q}(t). \tag{A.7}$$

Because $\boldsymbol{\Phi}$ has full rank, this relation defines an invertible linear map between physical and modal coordinates. The modal coordinates can therefore be obtained explicitly as:

$$\boldsymbol{q}(t) = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\,\boldsymbol{u}(t). \tag{A.8}$$

A.3.1 Initial displacements and their interpretation

Given an initial displacement $\boldsymbol{u}(0)$, the corresponding modal components are:

$$\boldsymbol{q}(0) = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{m}\,\boldsymbol{u}(0). \tag{A.9}$$

[4]The relations $\boldsymbol{M} = I$ and $\boldsymbol{K} = \boldsymbol{\Lambda}$ follow directly from $\boldsymbol{\phi}^{(n)} = \boldsymbol{m}^{-1/2}\boldsymbol{\psi}^{(n)}$ and $(\boldsymbol{\psi}^{(n)})^\mathsf{T}\boldsymbol{\psi}^{(m)} = \delta_{nm}$. Mass orthogonality gives $(\boldsymbol{\phi}^{(n)})^\mathsf{T}\boldsymbol{m}\boldsymbol{\phi}^{(m)} = \delta_{nm}$, while stiffness orthogonality follows from the eigenvalue relation $\boldsymbol{A}\boldsymbol{\psi}^{(m)} = \lambda_m\boldsymbol{\psi}^{(m)}$, yielding $(\boldsymbol{\phi}^{(n)})^\mathsf{T}\boldsymbol{k}\boldsymbol{\phi}^{(m)} = \lambda_m\delta_{nm}$. Thus the modal transformation simultaneously diagonalizes $\boldsymbol{m}$ and $\boldsymbol{k}$ and leads to the uncoupled modal equations $\ddot{q}_n(t) + \omega_n^2 q_n(t) = 0$.

The scalar $q_n(0)$ is the projection of the initial displacement onto the nth mode shape using the *mass inner product*:

$$(\boldsymbol{x}, \boldsymbol{y})_m = \boldsymbol{x}^\mathsf{T} \boldsymbol{m} \boldsymbol{y}.$$

This inner product is natural in structural dynamics because kinetic energy is:

$$T = \tfrac{1}{2} \dot{\boldsymbol{u}}^\mathsf{T} \boldsymbol{m} \dot{\boldsymbol{u}}.$$

The projection in **Eq. (A.9)** therefore measures how much kinetic energy the initial displacement is poised to excite in each mode.

A.3.2 Initial velocities

Differentiating **Eq. (A.5)** gives:

$$\dot{\boldsymbol{u}}(t) = \boldsymbol{\Phi}\, \dot{\boldsymbol{q}}(t).$$

Applying the same transformation as before yields:

$$\dot{\boldsymbol{q}}(0) = \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{m}\, \dot{\boldsymbol{u}}(0). \tag{A.10}$$

The initial modal velocities are therefore the mass-weighted projections of the initial velocity onto the modal directions.

A.3.3 Comment on mass matrix normalization

Relations **Eq. (A.9)**–**Eq. (A.10)** rely on the normalization **Eq. (A.6)**, that is:

$$\boldsymbol{M} := \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{m} \boldsymbol{\Phi} = I.$$

This choice is called *mass normalization* of the mode shapes. With this normalization the modal equations assume the simplest possible form:

$$\ddot{q}_n(t) + \omega_n^2 q_n(t) = 0.$$

Some references prefer to normalize mode shapes differently, for example:

$$(\boldsymbol{\phi}^{(n)})^\mathsf{T} \boldsymbol{k}\, \boldsymbol{\phi}^{(n)} = 1, \qquad \text{or} \qquad \|\boldsymbol{\phi}^{(n)}\| = 1.$$

In that case:

$$\boldsymbol{M} = \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{m} \boldsymbol{\Phi} \quad \text{is diagonal but not the identity,} \tag{A.11}$$

and the modal equations take the form:

$$M_n\, \ddot{q}_n(t) + K_n\, q_n(t) = 0.$$

The transformation of initial conditions then employs the inverse modal mass matrix:

$$\boldsymbol{q}(0) = \boldsymbol{M}^{-1} \boldsymbol{\Phi}^\mathsf{T} \boldsymbol{m}\, \boldsymbol{u}(0). \tag{A.12}$$

This transformation remains diagonal (because $\boldsymbol{M}$ is diagonal) but is less convenient than in the mass-normalized case.

The choice $\boldsymbol{\Phi}^\mathsf{T} \boldsymbol{m} \boldsymbol{\Phi} = I$ is therefore not required by physics but is a mathematical convenience that simplifies the transformation of initial conditions and the structure of the modal equations.

A.3.4 Role and significance of modal initial conditions

Each modal coordinate satisfies the uncoupled SDOF equation:

$$\ddot{q}_n(t) + \omega_n^2 q_n(t) = 0, \tag{A.13}$$

whose solution depends solely on $q_n(0)$ and $\dot{q}_n(0)$. The transformation:

$$\big(\boldsymbol{u}(0), \dot{\boldsymbol{u}}(0)\big) \longmapsto \big(\boldsymbol{q}(0), \dot{\boldsymbol{q}}(0)\big)$$

converts the full N-DOF initial-value problem into N uncoupled scalar problems. Once each modal oscillator in **Eq. (A.13)** is solved, the physical displacement response is obtained by modal superposition:

$$\boldsymbol{u}(t) = \boldsymbol{\Phi}\, \boldsymbol{q}(t). \tag{A.14}$$

This expression reflects the principle that free vibration of a linear MDOF system is a linear combination of its natural modes.

A.4 Rayleigh Damping

In many structural applications the full damping matrix $\boldsymbol{c}$ is not known from first principles. A common and effective modeling assumption is to express $\boldsymbol{c}$ as a linear combination of the mass and stiffness matrices:

$$\boldsymbol{c} = a_0\, \boldsymbol{m} + a_1\, \boldsymbol{k}, \tag{A.15}$$

where a_0 and a_1 are the *Rayleigh damping coefficients*. This representation is known as *Rayleigh damping* or *proportional damping*.

A.4.1 Justification and purpose

The main motivation for the Rayleigh form **Eq. (A.15)** is that it guarantees *classical damping*. Using the modal matrix $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{c}\, \boldsymbol{\Phi} = a_0\, \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{m} \boldsymbol{\Phi} + a_1\, \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{k} \boldsymbol{\Phi}.$$

Because the modal transformation simultaneously diagonalizes $\boldsymbol{m}$ and $\boldsymbol{k}$, namely:

$$\boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{m} \boldsymbol{\Phi} = I, \qquad \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{k} \boldsymbol{\Phi} = \boldsymbol{\Lambda} = \operatorname{diag}(\omega_1^2, \ldots, \omega_N^2),$$

the modal damping matrix is automatically diagonal:

$$\boldsymbol{C}_n := \boldsymbol{\Phi}^{\mathsf{T}} \boldsymbol{c}\, \boldsymbol{\Phi} = a_0\, I + a_1\, \boldsymbol{\Lambda} = \operatorname{diag}\big(2\zeta_1\omega_1,\, 2\zeta_2\omega_2,\, \ldots,\, 2\zeta_N\omega_N\big). \tag{A.16}$$

Rayleigh damping therefore satisfies the requirement for classical damping without additional assumptions.

A.4.2 Modal damping ratios

From **Eq. (A.16)**, the damping ratio associated with the nth mode is:

$$\zeta_n = \frac{1}{2}\left(a_0 \frac{1}{\omega_n} + a_1\, \omega_n\right). \tag{A.17}$$

Expression **Eq. (A.17)** highlights two contributions:

- the term a_0/ω_n dominates at *low* frequencies;
- the term $a_1\omega_n$ dominates at *high* frequencies.

Rayleigh damping therefore tends to increase with mode number unless a_1 is selected small.

A.4.3 Determination of the coefficients

Given two target damping ratios ζ_j and ζ_k for two modes with circular frequencies ω_j and ω_k, the Rayleigh coefficients follow from:

$$\begin{bmatrix} \frac{1}{2\omega_j} & \frac{\omega_j}{2} \\ \frac{1}{2\omega_k} & \frac{\omega_k}{2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \zeta_j \\ \zeta_k \end{bmatrix}.$$

This system is nonsingular as long as $\omega_j \neq \omega_k$.

A.4.4 Interpretation

Rayleigh damping is not a constitutive model of material dissipation. Instead, it is a mathematically convenient representation that:

- Preserves classical damping automatically, ensuring full modal. Decoupling even in the presence of damping.
- Allows prescribed damping levels at selected frequencies;
- Introduces velocity-dependent forces that follow the same spatial patterns as mass and stiffness.

A.5 Harmonic Forcing in MDOF Systems

Consider the forced-vibration equation of the linear system:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{c}\,\dot{\boldsymbol{u}}(t) + \boldsymbol{k}\,\boldsymbol{u}(t) = \boldsymbol{p}(t), \tag{A.18}$$

subjected to a harmonic excitation:

$$\boldsymbol{p}(t) = \boldsymbol{p}_0\,\sin(\omega t), \tag{A.19}$$

where ω is the forcing circular frequency and $\boldsymbol{p}_0$ is the vector of force amplitudes.

A.5.1 Transformation to modal coordinates

Introduce the modal expansion:

$$\boldsymbol{u}(t) = \boldsymbol{\Phi}\,\boldsymbol{q}(t), \tag{A.20}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}^{(1)}\ \ldots\ \boldsymbol{\phi}^{(N)}]$ contains the mass-normalized mode shapes satisfying $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{m}\boldsymbol{\Phi} = I$ and $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{k}\boldsymbol{\Phi} = \boldsymbol{\Lambda} = \mathrm{diag}(\omega_n^2)$.

Premultiplying **Eq. (A.18)** by $\boldsymbol{\Phi}^{\mathsf{T}}$ yields the uncoupled modal equations:

$$\ddot{q}_n(t) + 2\zeta_n\omega_n\,\dot{q}_n(t) + \omega_n^2\,q_n(t) = P_n(t), \qquad n = 1, \ldots, N, \tag{A.21}$$

because classical (Rayleigh) damping makes the modal damping matrix diagonal: $\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{c}\boldsymbol{\Phi} = \text{diag}(2\zeta_n\omega_n)$. The modal forcing is:

$$P_n(t) = \boldsymbol{\phi}^{(n)\mathsf{T}}\,\boldsymbol{p}_0\,\sin(\omega t) = P_{n0}\,\sin(\omega t), \qquad P_{n0} := \boldsymbol{\phi}^{(n)\mathsf{T}}\boldsymbol{p}_0. \tag{A.22}$$

A.5.2 Steady-state response of each modal SDOF equation

For the scalar forced-damped equation:

$$\ddot{q}_n + 2\zeta_n\omega_n\,\dot{q}_n + \omega_n^2 q_n = P_{n0}\,\sin(\omega t),$$

the steady-state solution has the form:

$$q_n(t) = Q_n\,\sin(\omega t - \varphi_n), \tag{A.23}$$

with modal amplitude:

$$Q_n = \frac{P_{n0}}{\sqrt{(\omega_n^2 - \omega^2)^2 + (2\zeta_n\omega_n\omega)^2}}, \tag{A.24}$$

and phase lag:

$$\varphi_n = \arctan\left(\frac{2\zeta_n\omega_n\omega}{\omega_n^2 - \omega^2}\right). \tag{A.25}$$

The damping therefore affects the response through the factor $2\zeta_n\omega_n\omega$, which broadens the resonance peak and introduces a phase shift.

A.5.3 Reconstruction of the physical steady-state motion

Using **Eq. (A.20)** and **Eq. (A.23)**, the displacement vector is:

$$\boldsymbol{u}(t) = \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\,Q_n\,\sin(\omega t - \varphi_n). \tag{A.26}$$

If the response at a specific degree of freedom j is required, the jth component is:

$$u_j(t) = \sum_{n=1}^{N} \phi_{jn}\,Q_n\,\sin(\omega t - \varphi_n), \tag{A.27}$$

where ϕ_{jn} denotes the jth component of $\boldsymbol{\phi}^{(n)}$. The acceleration amplitude at degree of freedom j follows from differentiation:

$$a_j^{\max}(\omega) = \omega^2 \left|\sum_{n=1}^{N} \phi_{jn}\,Q_n\right|.$$

This sequence,

$$\text{physical forces} \;\rightarrow\; \text{modal forces} \;\rightarrow\; \text{SDOF steady-state} \;\rightarrow\; \text{physical response},$$

forms the basis for the harmonic-response calculations used in the main text.

A.6 Response of MDOF Systems to Ground Motion

For an N-DOF structure subjected to spatially uniform earthquake ground acceleration $\ddot{u}_g(t)$, the dynamic behavior is governed by the equations of motion with effective inertia forces due to the ground motion:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{c}\,\dot{\boldsymbol{u}}(t) + \boldsymbol{k}\,\boldsymbol{u}(t) = \boldsymbol{p}_{\text{eff}}(t) = -\,\boldsymbol{m}\,\boldsymbol{\iota}\,\ddot{u}_g(t), \tag{A.28}$$

where $\boldsymbol{m}$, $\boldsymbol{c}$, and $\boldsymbol{k}$ are the mass, damping, and stiffness matrices, $\boldsymbol{u}(t)$ is the vector of floor displacements relative to the ground, and $\boldsymbol{\iota}$ is the influence vector.

A.6.1 Modal expansion of displacements and effective inertia forces

In modal analysis the displacement vector is expanded in terms of the normal modes $\boldsymbol{\phi}_n$ and the modal coordinates $q_n(t)$ [6, 4]:

$$\boldsymbol{u}(t) = \sum_{n=1}^{N} \boldsymbol{\phi}_n\, q_n(t), \qquad \boldsymbol{m}\,\boldsymbol{\iota} = \sum_{n=1}^{N} \boldsymbol{s}_n, \qquad \boldsymbol{s}_n = \Gamma_n \boldsymbol{m}\boldsymbol{\phi}_n, \tag{A.29}$$

where $\boldsymbol{s}_n$ is the nth modal inertia-force distribution and Γ_n is the modal participation factor:

$$\Gamma_n = \frac{L_n}{M_n}, \qquad L_n = \boldsymbol{\phi}_n^{\mathsf{T}} \boldsymbol{m}\boldsymbol{\iota}, \qquad M_n = \boldsymbol{\phi}_n^{\mathsf{T}} \boldsymbol{m}\boldsymbol{\phi}_n. \tag{A.30}$$

Substitution of the modal expansion into **Eq. (A.28)** and use of modal orthogonality lead to a set of uncoupled SDF equations in terms of the generalized coordinates $D_n(t)$:

$$\ddot{D}_n(t) + 2\,\zeta_n\omega_n \dot{D}_n(t) + \omega_n^2 D_n(t) = -\,\ddot{u}_g(t), \qquad q_n(t) = \Gamma_n D_n(t), \tag{A.31}$$

where ω_n and ζ_n are the natural frequency and damping ratio of the nth mode. Each mode therefore behaves as an equivalent SDF system subjected directly to the ground acceleration record $\ddot{u}_g(t)$.

A.6.2 Combination of modal contributions

The seismic response is obtained by solving the SDF problem in **Eq. (A.31)** for each mode and then superposing the modal contributions. The pseudo-acceleration response of the nth SDF system is:

$$A_n(t) = \omega_n^2 D_n(t).$$

The total structural response is recovered from the modal sums, for example in terms of displacements and any linear response quantity $r(t)$:

$$\boldsymbol{u}(t) = \sum_{n=1}^{N} \Gamma_n \boldsymbol{\phi}_n D_n(t), \qquad r(t) = \sum_{n=1}^{N} r_n^{\text{st}} A_n(t), \tag{A.32}$$

where r_n^{st} denotes the nth modal static response due to the force pattern $\boldsymbol{s}_n$.

This framework reduces the MDOF response under ground motion to a family of SDF analyses driven by the same acceleration record and then recombines their modal contributions in physical space.

B Experimental Identification of Mode Shapes from Floor Accelerations

The theoretical development in Sections A.1 and A.4 shows that, for a linear MDOF system with classical damping undergoing free vibration, the response admits the modal expansion (cf. **Eq. (A.14)**):

$$\boldsymbol{u}(t) = \sum_{n=1}^{N} \boldsymbol{\phi}^{(n)}\, q_n(t),$$

where $\boldsymbol{\phi}^{(n)}$ denotes the nth mode shape and $q_n(t)$ is the corresponding modal coordinate. Each $q_n(t)$ satisfies the underdamped SDOF equation **Eq. (A.13)** and therefore behaves as a decaying sinusoid with modal circular frequency ω_n and damping ratio ζ_n.

In time intervals where the motion is dominated by a single mode n (for example, after band-pass filtering around $f_n = \omega_n/(2\pi)$), the free-vibration response is well approximated by:

$$\boldsymbol{u}^{(n)}(t) \approx \boldsymbol{\phi}^{(n)}\, q_n(t), \qquad \boldsymbol{a}^{(n)}(t) = \ddot{\boldsymbol{u}}^{(n)}(t) \approx \boldsymbol{\phi}^{(n)}\, \ddot{q}_n(t).$$

Thus, apart from the common scalar factor $a_n(t) := \ddot{q}_n(t)$, the floor accelerations differ only by the components $\phi_j^{(n)}$.

The algorithm implemented in the Python can be found at this url[5].

The class `ModeShapeAnalyzer` uses this property to extract mode shapes from the measured floor accelerations and then enforces mass orthonormality using the known floor masses.

Python 3.13 Code

```
class ModeShapeAnalyzer:
    """Computes mode shapes using RMS ratios and correlation."""

    def __init__(self, num_floors, reference_floor=None,
                 filter_thresholds=None, mass_matrix=None):
        """
        num_floors : int
            Number of floors
        reference_floor : int, optional
            Reference floor (1-indexed). If None, uses top floor.
        filter_thresholds : dict, optional
            'ref_amp_ratio', 'glob_amp_ratio',
            'cos_theta_min', 'corr_threshold'
        mass_matrix : array-like, optional
            Mass matrix M (num_floors x num_floors). If provided, enables
            mass-normalization and mass-orthogonalization of mode shapes.
        """
```

The procedure has three main stages:

(1) constructing a reference mode shape from root-mean-square (RMS) floor accelerations and inter-floor correlations;

[5] https://github.com/Facundo-Pfeffer/UCBerkeley-SEMM-MS-Codebook/tree/master/CEE225_Dynamics/ce225_specific_homework_runs/HW11

(2) forming instantaneous, normalized shape snapshots from the acceleration time histories and optionally filtering them by amplitude and shape consistency;

(3) post-processing the identified shapes using the given mass matrix so that the final experimental modal matrix satisfies $\Phi^{\mathsf{T}} \boldsymbol{m} \Phi \approx I$.

B.1 RMS-based reference mode shape

For a time window where mode n is dominant, the input to the analyzer consists of the common time array and one acceleration array per floor:

Python 3.13 Code

```
mode_shape_ref, stats = analyzer.compute_mode_shape_statistics(
    time, acc_floor1, acc_floor2, acc_floor3, use_filter=True
)
```

Inside this routine, the first step is to compute the RMS acceleration at each floor:

Python 3.13 Code

```
time = np.asarray(time, dtype=float)
acc_data = [np.asarray(acc, dtype=float) for acc in acc_data]
rms_values = np.array([np.sqrt(np.mean(acc ** 2)) for acc in acc_data])
```

For floor j, the RMS acceleration is:

$$a_{j,\mathrm{RMS}}^{(n)} = \sqrt{\frac{1}{N_s} \sum_{k=1}^{N_s} \left(a_j^{(n)}(t_k)\right)^2}. \tag{B.1}$$

RMS stands for *root mean square*: the signal is squared, averaged in time, and then square-rooted. It provides an energy-consistent measure of the typical amplitude at each floor over the selected window.

A reference floor r is selected (by default the top floor), and RMS ratios with respect to that floor are formed as:

$$\rho_j^{(n)} = \frac{a_{j,\mathrm{RMS}}^{(n)}}{a_{r,\mathrm{RMS}}^{(n)}}, \qquad j = 1, \ldots, N_f.$$

These ratios determine the relative magnitudes but not the signs. To infer the signs, each floor signal is correlated with the reference floor:

Python 3.13 Code

```
def _compute_signs(self, acc_data, ref_idx):
    signs = np.ones(self.num_floors)
    ref_acc = acc_data[ref_idx]
    for i in range(self.num_floors):
        if i == ref_idx:
            continue
```

```
        num = np.mean(acc_data[i] * ref_acc)
        den = np.sqrt(np.mean(acc_data[i]**2) * np.mean(ref_acc**2)) + 1e-12
        c = num / den
        if abs(c) > self.filter_thresholds['corr_threshold']:
            signs[i] = np.sign(c)
        else:
            signs[i] = 1.0
    return signs
```

Here c is the sample correlation coefficient between floor i and the reference floor. The small constant 10^{-12} in the denominator is a regularization term that prevents division by (almost) zero when one of the two signals has very small energy in the window; without it, the ratio could become numerically unstable or spuriously large. If $|c|$ exceeds a minimum correlation level `corr_threshold` (default 0.05), the sign of floor i is set equal to the sign of c; otherwise the sign is left as $+1$.

The preliminary mode-shape components based on RMS ratios and signs are:

$$\phi_{j,\text{raw}}^{(n)} = s_j^{(n)}\,\rho_j^{(n)}, \qquad j = 1, \ldots, N_f.$$

This vector is then normalized by its largest absolute component and, if necessary, flipped so that that component is nonnegative:

Python 3.13 Code

```
rms_ratios = rms_values / rms_values[self.ref_idx]
signs = self._compute_signs(acc_data, self.ref_idx)
phi_raw = signs * rms_ratios

max_abs = np.max(np.abs(phi_raw))
if max_abs < 1e-12:
    mode_shape_normalized = phi_raw.copy()
else:
    mode_shape_normalized = phi_raw / max_abs

ref_idx_max = int(np.argmax(np.abs(mode_shape_normalized)))
if mode_shape_normalized[ref_idx_max] < 0.0:
    mode_shape_normalized *= -1.0
```

The resulting vector `mode_shape_normalized` is the reference mode shape $\widehat{\phi}_{\text{ref}}^{(n)}$, normalized so that its largest component is $+1$.

B.2 Instantaneous normalized shape snapshots and filtering

In addition to the RMS-based reference, the algorithm constructs *instantaneous* shape snapshots from each time step. For each sample t_k, the floor accelerations are collected and normalized by the instantaneous maximum absolute value across floors:

Python 3.13 Code

```
def _compute_instantaneous_shapes(self, time, acc_data, mode_shape_ref):
    n_samples = len(time)
```

```
    raw_shapes, raw_times = [], []
    raw_ref_amp, raw_max_inst = [], []
    for i in range(n_samples):
        inst_acc = np.array([acc[i] for acc in acc_data], dtype=float)
        max_inst = np.max(np.abs(inst_acc))
        if max_inst < 1e-12:
            continue
        inst_norm = inst_acc / max_inst
        if np.dot(inst_norm, mode_shape_ref) < 0.0:
            inst_norm = -inst_norm
        raw_shapes.append(inst_norm)
        raw_times.append(time[i])
        raw_ref_amp.append(abs(acc_data[self.ref_idx][i]))
        raw_max_inst.append(max_inst)
    ...
```

The vector `inst_norm` corresponds to a normalized shape $\boldsymbol{v}^{(n)}(t_k)$ with $\max_j |v_j^{(n)}(t_k)| = 1$. Its sign is chosen so that the dot product with the reference mode shape is nonnegative, which removes the global sign ambiguity. The arrays `raw_ref_amp` and `raw_max_inst` store, respectively, the instantaneous reference-floor amplitude and the global maximum amplitude and are used in the subsequent amplitude filters.

Samples with very low amplitudes are often dominated by measurement noise, and samples whose shape deviates strongly from the reference are likely to be contaminated by other modes or transients. The method therefore applies a simple amplitude filter and an angular (consistency) filter:

Python 3.13 Code

```
def _apply_filters(self, raw_shapes, raw_times,
                   raw_ref_amp, raw_max_inst, mode_shape_ref):
    ref_max = np.max(raw_ref_amp)
    glob_max = np.max(raw_max_inst)
    ref_amp_thresh  = self.filter_thresholds['ref_amp_ratio'] * ref_max
    glob_amp_thresh = self.filter_thresholds['glob_amp_ratio'] * glob_max
    cos_theta_min   = self.filter_thresholds['cos_theta_min']

    shapes_new, times_new = [], []
    for j in range(len(raw_shapes)):
        if raw_ref_amp[j]  < ref_amp_thresh:
            continue
        if raw_max_inst[j] < glob_amp_thresh:
            continue
        v = raw_shapes[j]
        num = float(np.dot(v, mode_shape_ref))
        den = (np.linalg.norm(v) * np.linalg.norm(mode_shape_ref) + 1e-12)
        cos_theta = num / den
        if cos_theta < cos_theta_min:
            continue
        shapes_new.append(v)
        times_new.append(raw_times[j])
    ...
```

The thresholds `ref_amp_ratio` and `glob_amp_ratio` (default values 0.05 and 0.02) retain only samples where both the reference floor and at least one floor exhibit a significant acceleration level.

The cosine of the angle between the instantaneous shape and the reference,

$$\cos\theta^{(n)}(t_k) = \frac{\boldsymbol{v}^{(n)}(t_k)^{\mathsf{T}}\widehat{\boldsymbol{\phi}}_{\text{ref}}^{(n)}}{\|\boldsymbol{v}^{(n)}(t_k)\|\,\|\widehat{\boldsymbol{\phi}}_{\text{ref}}^{(n)}\| + 10^{-12}},$$

is then required to be larger than `cos_theta_min` (typically 0.95). The regularization term 10^{-12} in the denominator avoids division by zero if an almost-zero snapshot slips through. Only snapshots that are sufficiently strong and closely aligned with the reference mode shape are kept; if all snapshots are rejected, the algorithm falls back to using the unfiltered set.

B.3 Statistical summary of instantaneous shapes

The function `_compute_statistics` condenses the cloud of instantaneous shapes into simple statistics:

Python 3.13 Code

```python
def _compute_statistics(self, shapes):
    if len(shapes) == 0:
        ...
    mean = np.mean(shapes, axis=0)
    std  = np.std(shapes, axis=0)
    cv   = std / (np.abs(mean) + 1e-10)
    ci_lower = mean - 1.96 * std
    ci_upper = mean + 1.96 * std
    return {
        'mean': mean, 'std': std, 'cv': cv,
        'ci_lower': ci_lower, 'ci_upper': ci_upper
    }
```

For each floor j and mode n, the component of `mean` approximates the average instantaneous shape, `std` measures the dispersion, and `cv` (coefficient of variation) is a non-dimensional measure of relative scatter. The factor 1.96 corresponds to an approximate 95% confidence interval under a normality assumption.

The dictionary returned by `compute_mode_shape_statistics` includes the reference mode shape, the raw and filtered instantaneous shapes, and the associated statistics. In the subsequent comparison with analytical eigenvectors from **Eq. (A.2)**–**Eq. (A.4)**, the reference vector `mode_shape_reference` is taken as the experimental mode shape for that mode, before mass-based post-processing.

B.4 Mass-based post-processing and orthonormalization

The experimental mode shapes identified from accelerations determine only *relative* amplitudes across floors. Their overall scaling is still arbitrary, and small inconsistencies may lead to a modal matrix that does not exactly satisfy the mass-orthogonality condition:

$$\Phi^{\mathsf{T}}\boldsymbol{m}\Phi = \boldsymbol{M} = \text{diag}(M_1, M_2, \ldots, M_{N_m}),$$

where $\boldsymbol{M}$ is the diagonal matrix of modal masses M_n. A particularly convenient normalization is the one that enforces unit modal mass for each mode:

$$(\boldsymbol{\phi}^{(n)})^{\mathsf{T}}\boldsymbol{m}\,\boldsymbol{\phi}^{(n)} = 1, \qquad n = 1, \ldots, N_m,$$

which is equivalent to the compact matrix identity used in Section A.2:

$$\Phi^{\mathsf{T}} \boldsymbol{m} \Phi = I.$$

Because the floor masses in the experiment are known and assumed to be less uncertain than the measured accelerations, a final post-processing step is performed to enforce this condition.

The class therefore accepts the physical mass matrix $\boldsymbol{m}$ (argument `mass_matrix`) and provides utilities for:

(i) mass-normalizing each experimental mode shape so that $(\boldsymbol{\phi}^{(n)})^{\mathsf{T}} \boldsymbol{m}\, \boldsymbol{\phi}^{(n)} = 1$ (method `normalize_mode_shape_`

(ii) applying a mass-weighted Gram–Schmidt procedure to a collection of experimental modes so that the resulting matrix Φ_{exp} satisfies $\Phi_{\text{exp}}^{\mathsf{T}} \boldsymbol{m} \Phi_{\text{exp}} \approx I$ (method `orthogonalize_mode_shapes_mass`).

After this post-processing, the experimental mode shapes are mass-orthonormal in the sense of Section A.2, and can be directly compared with the analytical eigenvectors obtained from the generalized eigenvalue problem **Eq. (A.2)**–**Eq. (A.4)**.

B.4.1 Mass normalization of a single mode

For a single mode shape vector $\boldsymbol{\phi}$ of length N_f, the method `normalize_mode_shape_mass` rescales it so that $\boldsymbol{\phi}^{\mathsf{T}} \boldsymbol{m} \boldsymbol{\phi} = 1$:

Python 3.13 Code

```
def normalize_mode_shape_mass(self, mode_shape):
    if self.mass_matrix is None:
        raise ValueError("mass_matrix must be provided")
    mode_shape = np.asarray(mode_shape, dtype=float)
    modal_mass = mode_shape.T @ self.mass_matrix @ mode_shape
    if modal_mass < 1e-12:
        raise ValueError("Modal mass is essentially zero")
    normalization_factor = 1.0 / np.sqrt(modal_mass)
    normalized_mode = mode_shape * normalization_factor
    return normalized_mode
```

The quantity $m_r = \boldsymbol{\phi}^{\mathsf{T}} \boldsymbol{m} \boldsymbol{\phi}$ is the *modal mass* of the experimental mode. Dividing $\boldsymbol{\phi}$ by $\sqrt{m_r}$ yields a mass-normalized vector consistent with the convention adopted in Section A.2.

B.4.2 Mass-orthogonalization of several experimental modes

When several experimental mode shapes are available (for example, the first three modes of the three-story building), they are stacked into a matrix $\Phi_{\text{exp}} \in \mathbb{R}^{N_f \times N_m}$ and processed by a mass-weighted Gram–Schmidt algorithm [9, 10]:

Python 3.13 Code

```
def orthogonalize_mode_shapes_mass(self, mode_shapes):
    if self.mass_matrix is None:
        raise ValueError("mass_matrix must be provided")
```

```
    mode_shapes = np.asarray(mode_shapes, dtype=float)
    if mode_shapes.ndim == 1:
        mode_shapes = mode_shapes.reshape(-1, 1)
    num_modes = mode_shapes.shape[1]
    orthogonalized = np.zeros_like(mode_shapes)
    for i in range(num_modes):
        v = mode_shapes[:, i].copy()
        for j in range(i):
            proj = v.T @ self.mass_matrix @ orthogonalized[:, j]
            v = v - proj * orthogonalized[:, j]
        modal_mass = v.T @ self.mass_matrix @ v
        if modal_mass < 1e-12:
            raise ValueError(
                "Mode became essentially zero after orthogonalization."
            )
        normalization_factor = 1.0 / np.sqrt(modal_mass)
        orthogonalized[:, i] = v * normalization_factor
    return orthogonalized
```

This implementation uses a modified Gram–Schmidt procedure in the mass-weighted inner product:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_m = \boldsymbol{x}^\mathsf{T} \boldsymbol{m} \boldsymbol{y},$$

which is the inner product associated with kinetic energy. Processing mode i consists of three steps: starting from the original experimental shape $\boldsymbol{v}_i$, subtracting its mass-weighted projections onto all previously processed modes, and finally rescaling the result so that $\boldsymbol{v}_i^\mathsf{T} \boldsymbol{m} \boldsymbol{v}_i = 1$. The subtraction of projections makes each new mode shape $\boldsymbol{\phi}^{(i)}$ mass-orthogonal to the previously obtained ones, $\boldsymbol{\phi}^{(i)\mathsf{T}} \boldsymbol{m} \boldsymbol{\phi}^{(j)} = 0$ for $j < i$, and the final rescaling enforces unit modal mass for each column.

Geometrically, the algorithm constructs a set of vectors that form an orthonormal basis with respect to the kinetic-energy inner product rather than the usual Euclidean one. This is the natural notion of orthonormality for structural dynamics, since the transformation $\boldsymbol{u}(t) = \Phi\, \boldsymbol{q}(t)$ decouples the equations of motion only when the columns of Φ satisfy the mass-orthogonality conditions in Section A.2. After processing all experimental modes, the resulting modal matrix Φ_{exp} satisfies:

$$\Phi_{\text{exp}}^\mathsf{T} \boldsymbol{m} \Phi_{\text{exp}} \approx I,$$

so that the experimental modes are directly comparable to the analytical eigenvectors obtained from the generalized eigenvalue problem **Eq. (A.2)**–**Eq. (A.4)**. Mass-weighted modified Gram–Schmidt is a standard construction in numerical linear algebra; see, for example, [**?**] for a detailed discussion of Gram–Schmidt orthogonalization and weighted inner products.

B.4.3 Verification of mass orthonormality

Finally, the method `verify_mass_orthonormality` evaluates how closely the processed modes satisfy the target condition:

Python 3.13 Code

```
def verify_mass_orthonormality(self, mode_shapes, tolerance=1e-3):
    modal_mass_matrix = self.compute_modal_mass_matrix(mode_shapes)
    diagonal = np.diag(modal_mass_matrix)
```

```
    off_diagonal = modal_mass_matrix - np.diag(diagonal)
    max_off_diagonal = np.max(np.abs(off_diagonal))
    max_diag_dev = np.max(np.abs(diagonal - 1.0))
    is_orthonormal = (max_off_diagonal < tolerance and
                      max_diag_dev < tolerance)
    return is_orthonormal, modal_mass_matrix, \
           max_off_diagonal, max_diag_dev
```

The matrix $\Phi^{\mathsf{T}}\boldsymbol{m}\Phi$ is computed, and the maximum off-diagonal entry and the maximum deviation of the diagonal from unity are reported. For the present three-story building, these quantities are small (well below the prescribed tolerance), confirming that the final experimental modal matrix is effectively mass-orthonormal and is therefore directly comparable to the analytical eigenvectors obtained from the generalized eigenvalue problem **Eq. (A.2)**–**Eq. (A.4)**.

C Damping Analysis Using Logarithmic Decrement

The damping ratios for each mode are estimated from free-vibration decay using the logarithmic decrement method, which is particularly suitable for lightly damped systems ($\zeta \lesssim 0.1$). In the single-mode regime described in Section A.1, the floor accelerations satisfy:

$$a_j^{(n)}(t) \approx \phi_j^{(n)}\, \ddot{q}_n(t), \qquad j = 1, 2, 3,$$

so each floor provides a scalar record that decays according to the nth modal SDOF equation **Eq. (A.13)**. Damping is therefore identified by analyzing the exponential decay of these free-vibration histories. The corresponding implemented code can be found at the publicly available repository.

C.1 Modal Acceleration and Combined Signal

To enforce a common decay window for all floors, a single scalar *combined modal signal* is constructed from the floor accelerations and the identified mode shape components $\phi_j^{(n)}$ as:

$$a_{\text{comb}}^{(n)}(t) = \sqrt{\frac{1}{3}\sum_{j=1}^{3}\left(\phi_j^{(n)}\, a_j^{(n)}(t)\right)^2}. \tag{C.1}$$

This RMS-type combination ensures that the decay start and end times are consistent across all floors of the structure vibrating in mode n.

C.2 Robust Detection of the Decay Window

The decay window is determined from $a_{\text{comb}}^{(n)}(t)$ using three steps:

- **Envelope computation:** The amplitude envelope is obtained from the Hilbert transform [9, 11]:
  $$A(t) = \left|\mathcal{H}\{a_{\text{comb}}^{(n)}(t)\}\right|,$$
  and smoothed by a moving average with window length $w \approx 0.5T_n$, where T_n is the modal period.
- **Adaptive search region:** The search for the decay start begins after the main forced response, using a fraction (about 30–40%) of the total record length and at least a few periods after the global maximum of $A(t)$.
- **Multi-criteria scoring:** Candidate start times are evaluated with a scalar score that rewards windows where the smoothed envelope decreases monotonically, the local amplitude is neither too small nor too close to the peak, and the trend is consistent with exponential decay. The candidate with the highest score above a minimum threshold is adopted as the decay start; otherwise, a fallback at a few periods after the peak amplitude is used.

After the decay start is fixed, the envelope in log-space is fitted over an early, high-amplitude segment to obtain a reference exponential decay rate. Using $\ln A(t)$ versus t, a linear regression yields:

$$\ln A(t) \approx \ln A_0 + \alpha\,(t - t_0), \qquad \alpha < 0, \tag{C.2}$$

where t_0 and A_0 denote the decay start time and initial envelope value. A forward scan in log-space then identifies the decay end as the time when the local slope deviates significantly from α or the envelope reaches the noise floor (about 1% of A_0). This trimming ensures that only the portion of the record that behaves as a clean exponential decay is used for logarithmic decrement.

C.3 Peak Extraction and Logarithmic Decrement

Within the identified decay window, peaks are extracted from each floor's acceleration using scipy.signal.find_peaks [9] with:

- an adaptive prominence threshold proportional to the maximum decay amplitude, increasing with the number of available cycles;
- a minimum spacing of about $0.7T_n$ between peaks to avoid multiple detections within a single cycle.

Only positive peaks are retained. For peaks x_i and x_{i+n} separated by n cycles, the ith logarithmic decrement is defined as:

$$\delta_i = \frac{1}{n} \ln\left(\frac{x_i}{x_{i+n}}\right),$$

and the average decrement is:

$$\bar{\delta} = \frac{1}{M} \sum_{i=1}^{M} \delta_i, \tag{C.3}$$

where M is the number of valid peak pairs. The damping ratio follows from the exact relation:

$$\zeta = \frac{\bar{\delta}}{\sqrt{4\pi^2 + \bar{\delta}^2}}, \tag{C.4}$$

which reduces to $\zeta \approx \bar{\delta}/(2\pi)$ for small damping.

C.4 Per-Floor Estimates and Consistency Checks

The decay window determined from $a_{\text{comb}}^{(n)}(t)$ is applied uniformly to the three floor records, and ζ is computed independently for each floor. For mode n, the mean damping ratio and its dispersion are:

$$\bar{\zeta}_n = \frac{1}{3} \sum_{j=1}^{3} \zeta_{n,j}, \qquad \sigma_{\zeta,n} = \sqrt{\frac{1}{2} \sum_{j=1}^{3} \left(\zeta_{n,j} - \bar{\zeta}_n\right)^2},$$

and the coefficient of variation is $\text{CoV}_n = \sigma_{\zeta,n}/\bar{\zeta}_n$. A low CoV_n (typically below 0.3) indicates that the three independent estimates are consistent and supports the interpretation of damping as a global structural property.

C.5 Exponential Fit Interpretation

For each mode, the identified damping ratio is also compared with the theoretical exponential decay in the time domain:

$$A(t) = A_0 \, \exp\big(-\zeta \, \omega_n (t - t_0)\big),$$

where $\omega_n = 2\pi f_n$ is the modal circular frequency. On a semi-logarithmic plot, this relation appears as a straight line of slope $-\zeta\omega_n$, and the close agreement between the fitted line and the measured envelope provides a visual validation of the logarithmic decrement results.

The summary damping ratios for the three identified modes of the three-story building are reported in Section 1.4, together with the per-floor estimates and associated statistics, and fall within typical ranges for lightly damped civil structures.

D Appendix: Python of History Response Analysis

This appendix summarizes the Python implementation used to solve Problem 3. The goal is computing the modal response of the 3-story MDOF building subjected to the 100% Loma Prieta (Palo Alto) ground motion, and producing the plots requested in items (a)–(e) of the problem statement. The corresponding implemented code can be found at the publicly available repository.

The implementation is organized in three main components:

1. A driver script `problem3_modal_response.py` that loads input data, calls the modal solver, and generates the plots and HTML summary.
2. A helper class `ModalResponseAnalyzer` that stores modal properties and carries out modal superposition, base-shear and base-moment calculations.
3. A generic Newmark routine `newmark_sdof` that integrates each uncoupled modal SDOF equation in time [4, 5].

All core calculations are carried out in SI units. Displacements are subsequently converted to inches and forces to kips/kip-ft for plotting and comparison with the problem statement.

D.1 Driver Script `problem3_modal_response.py`

The driver script handles data loading, calls the modal solver, and post-processes the results into the required response quantities. The entry point is:

Python 3.13 Code

```
def main():
    print("=" * 70)
    print("Problem #3: Modal Response Analysis")
    print("=" * 70)

    # System properties from the LaTeX section
    m = np.array([
        [1180, 0,   0],
        [0,   1180, 0],
        [0,   0,   910]
    ])
    Phi = np.array([
        [0.771, -1.916,  2.051],
        [1.755, -1.331, -1.903],
        [2.495,  1.982,  0.914]
    ]) * 1e-2         # mass-orthonormal modes ( 10 ^-2 in hand calc)
    f_n    = np.array([2.00, 7.20, 13.75])  # Hz
    omega_n = 2 * np.pi * f_n               # rad/s
    zeta    = np.array([0.0113, 0.0157, 0.0093])
    floor_heights = np.array([3.5, 7.0, 10.5])

    script_dir = Path(__file__).parent
    output_dir = script_dir.parent.parent / "highlighted_htmls"
    output_dir.mkdir(exist_ok=True)

    run_problem3_analysis(
```

```
        mass_matrix=m,
        mode_shapes=Phi,
        natural_freqs=omega_n,
        damping_ratios=zeta,
        floor_heights=floor_heights,
        output_dir=output_dir,
    )
```

Function `run_problem3_analysis` performs the following steps:

a) Reads the ground-motion file `ground_motion_excitation.csv`, identifies the time and table acceleration columns (e.g. `tableAccX_filtered`), and converts the ground acceleration to $\mathrm{m/s^2}$ based on its magnitude.

b) Instantiates `ModalResponseAnalyzer` from the mass matrix, mass-orthonormal mode shapes, modal frequencies, damping ratios, and floor heights; participation factors, effective modal masses, and effective heights are printed for verification.

c) Solves the uncoupled modal equations for $D_n(t)$, $\dot{D}_n(t)$, and $\ddot{D}_n(t)$ using Newmark integration, then forms the modal coordinates $q_n(t) = \Gamma_n D_n(t)$ and converts them to inches for item (a).

d) Reconstructs floor displacements and accelerations using modal superposition, and adds $\ddot{u}_g(t)$ to obtain total floor accelerations, which are then converted to inches and inches per second squared for items (b) and (c).

e) Reads the measured floor and table accelerations (item (v)), converts them from g to $\mathrm{m/s^2}$, and performs a simple double integration (trapezoidal rule) to obtain measured displacements for overlay in the floor plots.

f) Computes base shear and base overturning moment from modal pseudo-accelerations and the static modal patterns, then converts to kips and kip-ft for items (d) and (e).

g) Generates Plotly HTML files for each requested plot and a summary HTML page *problem3_summary.html* that aggregates modal properties, peak responses, and all figures.

A representative example of the plotting functions is the floor displacement plot for item (b):

Python 3.13 Code

```
def plot_floor_displacements(time, u, output_dir,
                             measured_time=None, measured_disp=None):
    fig = make_subplots(
        rows=3, cols=1,
        subplot_titles=[
            f"Floor {j+1} Displacement Response u_{j+1}(t)"
            for j in range(3)
        ],
        vertical_spacing=0.08,
    )

    colors      = [Colors.BERKELEY_BLUE,
                   Colors.CALIFORNIA_GOLD,
                   Colors.FOUNDERS_ROCK]
```

```
    floor_names = ["Floor 1", "Floor 2", "Floor 3"]

    for j in range(3):
        # analytical (relative) displacement in inches
        fig.add_trace(
            go.Scatter(
                x=time,
                y=u[j, :],
                mode="lines",
                name=floor_names[j],
                line=dict(color=colors[j], width=2),
                hovertemplate=(
                    f"<b>{floor_names[j]}</b><br>"
                    "Time: %{x:.2f} s<br>"
                    "Floor displacement u%{customdata}: %{y:.3f} in<br>"
                    "Meaning: relative to moving ground, converted to inches."
                    "<extra></extra>"
                ),
                customdata=np.full_like(time, j+1, dtype=float),
            ),
            row=j+1, col=1,
        )

        # optional overlay: measured displacements from double integration
        if measured_time is not None and measured_disp is not None:
            fig.add_trace(
                go.Scatter(
                    x=measured_time,
                    y=measured_disp[j],
                    mode="markers",
                    name=f"{floor_names[j]} (measured)",
                    marker=dict(color=colors[j], size=5, symbol="circle-open"),
                    hovertemplate=(
                        f"<b>{floor_names[j]} measured</b><br>"
                        "Time: %{x:.2f} s<br>"
                        "Disp (via    acc   ): %{y:.3f} in<br>"
                        "Computed by double-integrating measured accel."
                        "<extra></extra>"
                    ),
                ),
                row=j+1, col=1,
            )
```

Similar functions generate modal displacement plots, floor acceleration plots with measured overlays, base shear and base moment time histories, and a diagnostic figure comparing absolute and relative accelerations of the third floor (sign consistency check between computed and measured signals).

D.2 `ModalResponseAnalyzer`: Modal Backbone

The class `ModalResponseAnalyzer` provides a reusable modal backbone for both the direct time-history analysis (Problem 3) and the response spectrum analysis (Problem 4). Its constructor stores the mass matrix m, mode shapes Φ, modal frequencies ω_n, damping ratios ζ_n, and floor heights, and precomputes all modal quantities required later:

Python 3.13 Code

```python
class ModalResponseAnalyzer:
    """Analyzes modal response of MDOF structure to ground motion."""

    def __init__(self, mass_matrix, mode_shapes,
                 natural_freqs, damping_ratios,
                 floor_heights=None):
        self.m       = np.asarray(mass_matrix, dtype=float)
        self.Phi     = np.asarray(mode_shapes, dtype=float)
        self.omega_n = np.asarray(natural_freqs, dtype=float)
        self.zeta    = np.asarray(damping_ratios, dtype=float)

        if self.Phi.ndim == 1:
            self.Phi = self.Phi.reshape(-1, 1)

        self.n_floors, self.n_modes = self.Phi.shape
        # basic checks on input lengths omitted here for brevity

        self.floor_heights = (
            np.array([i + 1 for i in range(self.n_floors)], dtype=float)
            if floor_heights is None
            else np.asarray(floor_heights, dtype=float)
        )

        # influence vector for uniform base excitation
        self.iota = np.ones(self.n_floors)

        # precompute participation factors, effective masses, etc.
        self._compute_modal_properties()
```

The private method `_compute_modal_properties` implements the standard modal definitions. With $\boldsymbol{\phi}_n$ denoting the n-th mode shape, $\boldsymbol{\iota}$ the influence vector, and h_j the floor heights, the participation numerator, modal mass, participation factor, effective modal mass, and effective height are:

$$L_n = \boldsymbol{\phi}_n^{\mathsf{T}} m \boldsymbol{\iota}, \qquad M_n = \boldsymbol{\phi}_n^{\mathsf{T}} m \boldsymbol{\phi}_n, \qquad \Gamma_n = \frac{L_n}{M_n}, \qquad M_n^* = \Gamma_n^2 M_n, \qquad h_n^* = \frac{\sum_j m_{jj} \phi_{jn} h_j}{\sum_j m_{jj} \phi_{jn}}. \tag{D.1}$$

In code, these quantities are computed as:

Python 3.13 Code

```python
def _compute_modal_properties(self):
    # participation numerator L_n and modal mass M_n
    self.L = np.array([
        self.Phi[:, n].T @ self.m @ self.iota
        for n in range(self.n_modes)
    ])
    self.M_modal = np.array([
        self.Phi[:, n].T @ self.m @ self.Phi[:, n]
        for n in range(self.n_modes)
    ])

    # participation factor  _n  and effective modal mass M_n*
    self.Gamma = self.L / self.M_modal
```

```
        self.M_eff = self.Gamma**2 * self.M_modal

        # effective modal heights h*
        self.h_star = np.zeros(self.n_modes)
        for n in range(self.n_modes):
            num = sum(
                self.m[j, j] * self.Phi[j, n] * self.floor_heights[j]
                for j in range(self.n_floors)
            )
            den = sum(
                self.m[j, j] * self.Phi[j, n]
                for j in range(self.n_floors)
            )
            self.h_star[n] = num / den if abs(den) > 1e-12 else 0.0
```

For base-shear and response-spectrum calculations, modal static lateral force patterns and base moments are also precomputed. The modal lateral force vector for mode n is:

$$\boldsymbol{s}_n = \Gamma_n \, m \, \boldsymbol{\phi}_n,$$

from which the story shears and base moment follow by vertical summation:

Python 3.13 Code

```
        # modal static story shears V_static[r, n]
        self.V_static = np.zeros((self.n_floors, self.n_modes))
        for n in range(self.n_modes):
            s_n = self.Gamma[n] * self.m @ self.Phi[:, n]
            for r in range(self.n_floors):
                self.V_static[r, n] = np.sum(s_n[r:])

        # modal static base moment
        self.Mb_static = np.zeros(self.n_modes)
        for n in range(self.n_modes):
            s_n = self.Gamma[n] * self.m @ self.Phi[:, n]
            self.Mb_static[n] = sum(
                s_n[j] * self.floor_heights[j]
                for j in range(self.n_floors)
            )
```

D.2.1 Uncoupled Modal Equations and Newmark Integration

For each mode, the uncoupled SDOF equation under base acceleration $\ddot{u}_g(t)$ is:

$$\ddot{D}_n(t) + 2\,\zeta_n \omega_n \dot{D}_n(t) + \omega_n^2 D_n(t) = -\,\ddot{u}_g(t). \tag{D.2}$$

This is solved with zero initial conditions and an equivalent SDOF interpretation with $m_{\text{eff}} = 1$, $c_{\text{eff}} = 2\zeta_n\omega_n$, $k_{\text{eff}} = \omega_n^2$, and $p_{\text{eff}}(t) = -\ddot{u}_g(t)$. The method `solve_modal_equations` encapsulates this step and calls `newmark_sdof`:

Python 3.13 Code

```
def solve_modal_equations(self, ug_ddot, time, dt=None):
    ug_ddot = np.asarray(ug_ddot, dtype=float).flatten()
    time    = np.asarray(time,    dtype=float).flatten()

    if dt is None:
        dt = time[1] - time[0] if len(time) > 1 else 0.01

    n_steps = len(time)
    D      = np.zeros((self.n_modes, n_steps))
    D_dot  = np.zeros((self.n_modes, n_steps))
    D_ddot = np.zeros((self.n_modes, n_steps))

    for n in range(self.n_modes):
        m_eff = 1.0
        c_eff = 2.0 * self.zeta[n] * self.omega_n[n]
        k_eff = self.omega_n[n]**2
        p_eff = -ug_ddot  # effective base-excitation load

        results = newmark_sdof(
            m_eff, k_eff, c_eff, p_eff,
            dt, u0=0.0, v0=0.0, method="constant"
        )
        D[n, :]      = results[:, 0]
        D_dot[n, :]  = results[:, 1]
        D_ddot[n, :] = results[:, 2]

    return D, D_dot, D_ddot
```

Modal coordinates for reporting and later use are then given by $q_n(t) = \Gamma_n D_n(t)$:

Python 3.13 Code

```
def compute_modal_coordinates(self, D):
    q = np.zeros_like(D)
    for n in range(self.n_modes):
        q[n, :] = self.Gamma[n] * D[n, :]
    return q
```

D.2.2 Floor Responses and Base Actions

Floor displacements, velocities, and accelerations follow from modal superposition. With $q_n(t) = \Gamma_n D_n(t)$, the relative floor responses are:

$$u_j(t) = \sum_n \phi_{jn} q_n(t), \qquad \ddot{u}_j^{\text{rel}}(t) = \sum_n \phi_{jn}\, \Gamma_n \ddot{D}_n(t),$$

and total accelerations at each floor include the ground term:

$$\ddot{u}_j(t) = \ddot{u}_j^{\text{rel}}(t) + \ddot{u}_g(t).$$

The method `compute_floor_responses` implements these expressions:

Python 3.13 Code

```
def compute_floor_responses(self, D, D_dot, D_ddot, ug_ddot=None):
    n_steps = D.shape[1]
    u      = np.zeros((self.n_floors, n_steps))
    u_dot  = np.zeros((self.n_floors, n_steps))
    u_ddot = np.zeros((self.n_floors, n_steps))

    for j in range(self.n_floors):
        for n in range(self.n_modes):
            q_n = self.Gamma[n] * D[n, :]
            u[j,     :] += self.Phi[j, n] * q_n
            u_dot[j, :] += self.Phi[j, n] * self.Gamma[n] * D_dot[n, :]
            u_ddot[j,:] += self.Phi[j, n] * self.Gamma[n] * D_ddot[n, :]

    if ug_ddot is not None:
        for j in range(self.n_floors):
            u_ddot[j, :] += ug_ddot

    return u, u_dot, u_ddot
```

Base shear and base moment are obtained by combining modal pseudo-accelerations $A_n(t) = \omega_n^2 D_n(t)$ with the static modal patterns:

$$V_b(t) = \sum_n V_{b,n}^{\text{st}} A_n(t), \qquad M_b(t) = \sum_n M_{b,n}^{\text{st}} A_n(t),$$

where $V_{b,n}^{\text{st}}$ is the modal static base shear (first row of `V_static`) and $M_{b,n}^{\text{st}}$ is the modal static base moment:

Python 3.13 Code

```
def compute_base_shear(self, D):
    n_steps = D.shape[1]
    V_base  = np.zeros(n_steps)
    V_b_static = self.V_static[0, :]  # base story for each mode
    for n in range(self.n_modes):
        A_n    = self.omega_n[n]**2 * D[n, :]
        V_base += V_b_static[n] * A_n
    return V_base

def compute_base_moment(self, D):
    n_steps = D.shape[1]
    M_base  = np.zeros(n_steps)
    for n in range(self.n_modes):
        A_n    = self.omega_n[n]**2 * D[n, :]
        M_base += self.Mb_static[n] * A_n
    return M_base
```

D.3 Newmark Integrator `newmark_sdof`

The function `newmark_sdof` provides a reusable implementation of Newmark's method for a single SDOF equation:

$$m\,\ddot{u}(t) + c\,\dot{u}(t) + k\,u(t) = p(t).$$

The routine supports constant-average acceleration ($\beta = 1/4$, $\gamma = 1/2$) and linear-acceleration ($\beta = 1/6$, $\gamma = 1/2$) variants:

Python 3.13 Code

```
def newmark_sdof(m: float, k: float, c: float, p: np.ndarray, dt: float,
                 u0: float, v0: float,
                 method: str = "constant") -> np.ndarray:
    """
    Newmark's Method for a single-degree-of-freedom system.
    m*u'' + c*u' + k*u = p(t)
    """
    p = np.asarray(p).flatten()
    n = len(p)

    # choose beta, gamma
    method_lower = method.lower()
    if method_lower == "constant":
        beta  = 1/4   # constant-average acceleration
        gamma = 1/2
    elif method_lower == "linear":
        beta  = 1/6   # linear acceleration
        gamma = 1/2
    else:
        raise ValueError("newmark_sdof: method must be 'constant' or 'linear'."
            )

    # allocate response vectors
    u = np.zeros(n)
    v = np.zeros(n)
    a = np.zeros(n)

    # initial conditions
    u[0] = u0
    v[0] = v0
    a[0] = (p[0] - c * v[0] - k * u[0]) / m

    # standard Newmark constants
    a0 = 1.0 / (beta * dt**2)
    a1 = gamma / (beta * dt)
    a2 = 1.0 / (beta * dt)
    a3 = 1.0 / (2.0 * beta) - 1.0
    a4 = gamma / beta - 1.0
    a5 = dt * (gamma / (2.0 * beta) - 1.0)

    # effective stiffness
    k_eff = k + a0 * m + a1 * c

    # time stepping
    for i in range(n - 1):
        p_eff = (p[i+1]
```

```
                + m * (a0 * u[i] + a2 * v[i] + a3 * a[i])
                + c * (a1 * u[i] + a4 * v[i] + a5 * a[i]))
        u[i+1] = p_eff / k_eff
        a[i+1] = a0 * (u[i+1] - u[i]) - a2 * v[i] - a3 * a[i]
        v[i+1] = v[i] + dt * ((1.0 - gamma) * a[i] + gamma * a[i+1])

    return np.column_stack((u, v, a))
```

In Problem 3, this routine is called once per mode by `ModalResponseAnalyzer`, using the effective modal parameters and the base-acceleration forcing defined in **Eq. (D.2)**. The resulting arrays of $D_n(t)$, $\dot{D}_n(t)$, and $\ddot{D}_n(t)$ feed directly into the modal coordinates, floor responses, and base actions described above.

Overall, the combination of `problem3_modal_response.py`, `ModalResponseAnalyzer`, and `newmark_sdof` provides a compact and reusable framework for:

a) exposing the modal displacement histories $q_n(t)$,

b) reconstructing floor displacement and acceleration responses in inches,

c) and computing base shear and overturning moment time histories in kips and kip-ft,

fully addressing items (a)–(e) of Problem 3 and allowing direct comparison with the measured response from the shake-table records. ::contentReference[oaicite:0]index=0

E Modal Spectral Analysis

The response spectrum analysis in Problem 4 is implemented in the script `problem4_rsa.py`. The code uses the design spectrum in `spectrum.csv`, the modal properties from Problem 3, and the modal participation factors computed by `ModalResponseAnalyzer` to obtain:

1. Modal pseudo-accelerations $A_{n,0}$ (in g) by interpolation in period and damping.
2. Modal spectral displacements $D_{n,0} = A_{n,0}/\omega_n^2$ (reported in inches).
3. SRSS peak floor displacements $u_{j,\text{SRSS}}$ from modal contributions $\Gamma_n \, \phi_{jn} \, D_{n,0}$.
4. SRSS base shear $V_{b,\text{SRSS}}$ from modal static base shears and pseudo-accelerations.

E.1 Spectrum Input and Damping Interpolation

The design spectrum is read from `spectrum.csv`, which contains the period column `Tn` and five pseudo-acceleration columns `A_1`–`A_5` corresponding to $0, 1, 2, 3, 5\%$ damping:

Python 3.13 Code

```python
def load_spectrum(csv_path: Path):
    df = pd.read_csv(csv_path)
    return df
```

For each mode, the code interpolates both in period and damping. Given a modal damping ζ_n (in decimal form), the helper `map_damping_brackets` identifies the two bounding damping curves and the interpolation weight w between them:

Python 3.13 Code

```python
def map_damping_brackets(zeta_decimal: float):
    pct     = zeta_decimal * 100.0       # e.g. 1.13, 1.57, ...
    targets = [0, 1, 2, 3, 5]
    # clamp and bracket in {0,1,2,3,5}%
    ...
    return lower_idx, upper_idx, low_pct, up_pct, w_up
```

The function `get_psa_g` first interpolates in period along each bracketing damping curve to obtain $A_{n,0}$ at T_n, then interpolates between the two damping curves:

Python 3.13 Code

```python
def get_psa_g(df_spectrum, T_mode, zeta_decimal):
    lower_idx, upper_idx, low_pct, up_pct, w_up = \
        map_damping_brackets(zeta_decimal)

    Tn   = df_spectrum["Tn"].values
    cols = [c for c in df_spectrum.columns if c != "Tn"]

    def col_name(idx):
```

```
        return cols[idx]  # A_1..A_5

    psa_low = np.interp(T_mode, Tn, df_spectrum[col_name(lower_idx)])
    psa_up  = np.interp(T_mode, Tn, df_spectrum[col_name(upper_idx)])

    psa = (1.0 - w_up) * psa_low + w_up * psa_up   # in g
    return psa, (low_pct, up_pct, w_up)
```

This provides, for each mode n, a pseudo-acceleration $A_{n,0}$ in units of g plus the bracketed damping information $(\zeta_{\text{low}}, \zeta_{\text{up}}, w)$ used in the interpolation.

E.2 Modal Spectral Ordinates

The function `run_problem4` sets default modal properties (when not passed explicitly), constructs a `ModalResponseAnalyzer` to obtain participation factors, and then computes the spectral ordinates:

Python 3.13 Code

```
analyzer = ModalResponseAnalyzer(
    mass_matrix=mass_matrix,
    mode_shapes=mode_shapes,
    natural_freqs=natural_freqs,
    damping_ratios=damping_ratios,
    floor_heights=floor_heights,
)

# Modal periods
T_modes = 2 * np.pi / natural_freqs

# Pseudo-acceleration A_{n,0} in g, with damping interpolation info
psa_results = [
    get_psa_g(df_spec, T_modes[i], damping_ratios[i])
    for i in range(len(natural_freqs))
]
psa_g       = np.array([r[0] for r in psa_results])
damping_info = [r[1] for r in psa_results]

# Convert to SI and compute spectral displacements
psa_mps2   = psa_g * 9.81
D_modes_m  = psa_mps2 / (natural_freqs ** 2)   # [m]
D_modes_in = D_modes_m * 39.3701               # [in]
A_modes_inps2 = psa_mps2 * 39.3701             # [in/s^2]
```

The quantities $D_{n,0}$ in inches and $A_{n,0}$ in in/s^2 constitute the modal spectral displacements and pseudo-accelerations required for the SRSS combination and for comparison with the time-history results from Problem 3.

E.3 SRSS Floor Displacements and Base Shear

The peak floor displacements are obtained by superposing the modal contributions $\Gamma_n \, \phi_{jn} \, D_{n,0}$ by the square-root of sum of squares (SRSS), under the assumption of statistical independence:

$$u_{j,\text{SRSS}} = \sqrt{\sum_{n=1}^{3} \left(\Gamma_n \, \phi_{jn} \, D_{n,0}\right)^2}.$$

In the code, `mode_shapes` stores ϕ_{jn} and `analyzer.Gamma` stores Γ_n:

Python 3.13 Code

```
n_floors = mode_shapes.shape[0]
u_srss_in = np.zeros(n_floors)

for j in range(n_floors):
    terms = []
    for n in range(len(natural_freqs)):
        contrib = analyzer.Gamma[n] * mode_shapes[j, n] * D_modes_in[n]
        terms.append(contrib ** 2)
    u_srss_in[j] = np.sqrt(np.sum(terms))
```

The SRSS base shear is computed by combining the modal static base shears $V_{b,n}^{\text{st}}$ (stored in `analyzer.V_static`) with the modal pseudo-accelerations:

$$V_{b,\text{SRSS}} = \sqrt{\sum_{n=1}^{3} \left(V_{b,n}^{\text{st}} \, A_{n,0}\right)^2}.$$

The code evaluates this in Newtons and converts the result to kips:

Python 3.13 Code

```
Vb_static   = analyzer.V_static[0, :]      # modal static base shear [N/(m/s  )
    ]
Vb_modal_N  = Vb_static * psa_mps2
Vb_srss_N   = np.sqrt(np.sum(Vb_modal_N ** 2))
Vb_srss_kips = Vb_srss_N / 4448.22
```

The SRSS peak floor displacements $u_{j,\text{SRSS}}$ and base shear $V_{b,\text{SRSS}}$ provide the RSA estimates required by Problem 4 and are intended to be compared directly against the peak responses obtained from the time-history solution in Problem 3.

E.4 Spectrum Plot and HTML Summary

The function `plot_spectrum` creates an interactive HTML plot of the design spectrum, drawing all damping curves and marking the modal points at $(T_n, A_{n,0})$. Finally, `create_summary_html` gathers the modal table, SRSS floor displacements, and SRSS base shear into `problem4_summary.html`, which embeds the spectrum figure and reports the numerical values used in the comparison with the time-history analysis.

F References

[1] University of California, Berkeley, "Richmond field station." https://rfs.berkeley.edu/home, 2025. Accessed: December 1, 2025.

[2] Department of Civil and Environmental Engineering, University of California, Berkeley, "Richmond field station laboratories and earthquake simulator facility." https://ce.berkeley.edu/programs/semm/facilities, 2025. Accessed: December 1, 2025.

[3] Pacific Earthquake Engineering Research Center, "Earthquake simulator laboratory, richmond field station." https://peer.berkeley.edu/earthquake-simulator-laboratory, 2025. Accessed: December 1, 2025.

[4] M. DeJong, "Lectures for cee225: Structural dynamics." Course lectures, University of California, Berkeley, 2025. Fall 2025 semester SEMM MS Program.

[5] M. A. Gomez, "Discussion sections for cee225: Structural dynamics." GSI-led discussions, University of California, Berkeley, 2025. Fall 2025 semester SEMM MS Program.

[6] A. K. Chopra, *Dynamics of Structures: Theory and Applications to Earthquake Engineering*. Boston: Pearson Education, 4th, global edition ed., 2014.

[7] COMSOL AB, "Response spectrum analysis theory," 2018. COMSOL Multiphysics® 5.4 documentation. Accessed 2025-12-11.

[8] L. Anand and S. Govindjee, *Continuum Mechanics of Solids*. Oxford University Press, 07 2020.

[9] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[10] Wikipedia contributors, "Gram–Schmidt process." https://en.wikipedia.org/wiki/Gram%E2%80%93Schmidt_process. Accessed: 2025-12-08.

[11] SciPy Developers, "scipy.signal.hilbert," 2025. SciPy Signal Processing Reference. Accessed 2025-12-11.