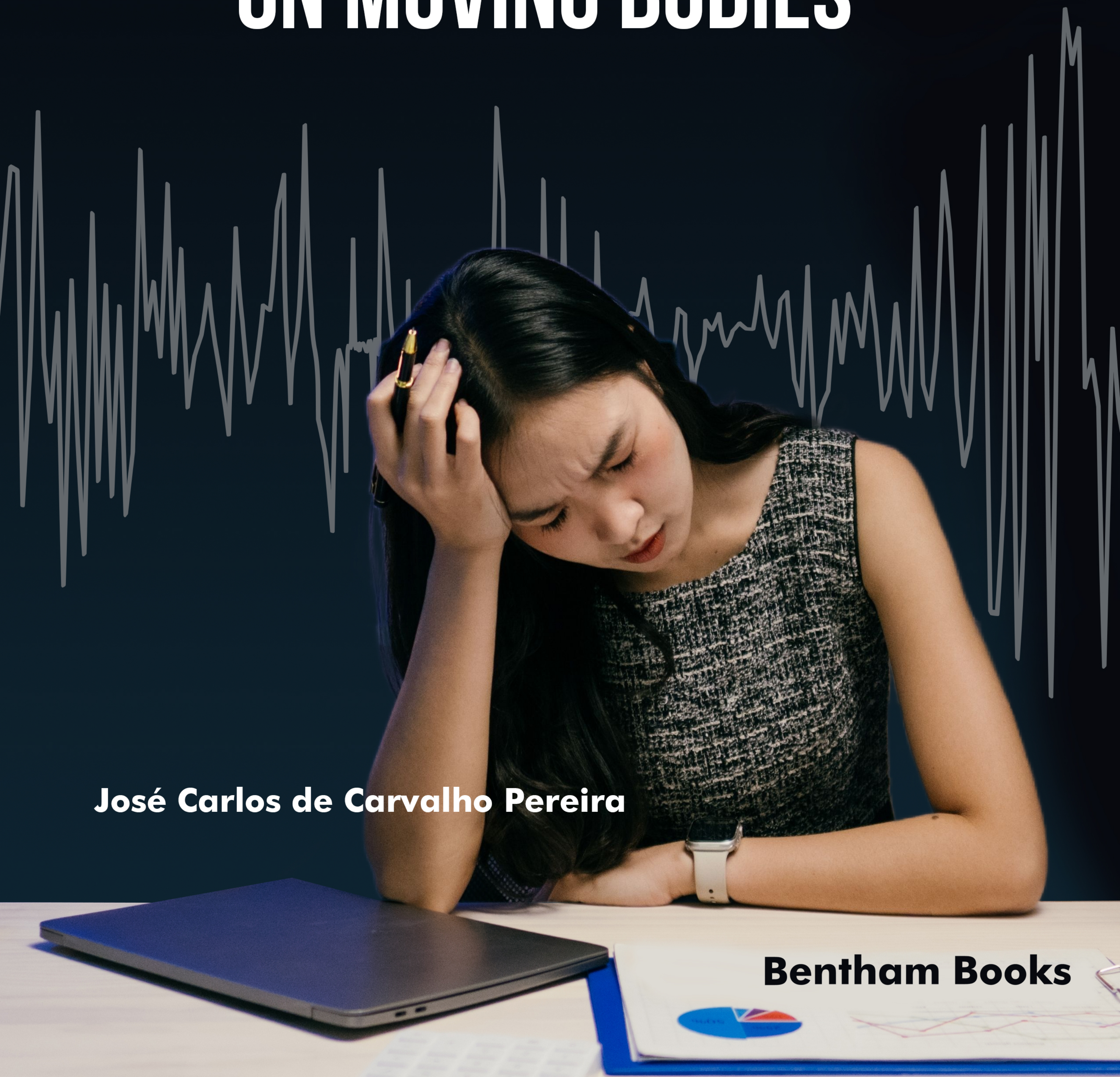
FATIGUE ANALYSIS ON MOVING BODIES



José Carlos de Carvalho Pereira

Bentham Books

Fatigue Analysis on Moving Bodies

Authored by

José Carlos de Carvalho Pereira

Department of Mechanical Engineering
Federal University of Santa Catarina
Florianópolis, Brazil

Fatigue Analysis on Moving Bodies

Author: José Carlos de Carvalho Pereira

ISBN (Online): 978-981-5313-53-6

ISBN (Print): 978-981-5313-54-3

ISBN (Paperback): 978-981-5313-55-0

First published in 2025.

need for a court order if at any point you breach any terms of this License Agreement. In no event will any delay or failure by Bentham Science Publishers in enforcing your compliance with this License Agreement constitute a waiver of any of its rights.

3. You acknowledge that you have read this License Agreement, and agree to be bound by its terms and conditions. To the extent that any other terms and conditions presented on any website of Bentham Science Publishers conflict with, or are inconsistent with, the terms and conditions set out in this License Agreement, you acknowledge that the terms and conditions set out in this License Agreement shall prevail.

Bentham Science Publishers Pte. Ltd.
80 Robinson Road #02-00
Singapore 068898
Singapore
Email: subscriptions@benthamscience.net

CONTENTS

PREFACE

The purpose of this book is to bring some particular aspects of the preliminary design of elastic structural elements that make up dynamic systems, here called moving bodies, which undergo transient load regimes. Within this perspective, the content covered in this book aims to guide undergraduate and postgraduate students of mechanical engineering or related engineering in ordering knowledge for its application in the design of dynamically loaded mechanical components.

This book aims to study dynamic systems closer to those found in our everyday lives, which present some complexities, whether from a geometric point of view or from the point of view of their loading. In this context, the Lagrangian formulation of the problem for obtaining differential equations of motion for these dynamic systems is recurrent throughout the book.

Differential equations of motion are solved in both the frequency domain and the time domain. However, the focus of this book lies on the response in the time domain, as such results allow observing the behavior of dynamic systems within transient regimes such as the startup and shutdown, short circuit, and passage through natural vibration frequencies. The response in the time domain also allows obtaining the evolution of mechanical stresses in elastic structural elements within transient regimes, and therefore, the application of fatigue failure theories.

To understand these concepts discussed in this book, models of real mechanical engineering systems are developed, such as gear trains, vehicle suspension systems, and rotating machines.

Contents:

- Chapter 1. Introduction to the Dynamics of Mechanical System.
- Chapter 2. Lagrange’s Equations of Motion of Mechanical Systems.
- Chapter 3. Energy Formulation of Discrete Systems.
- Chapter 4. Energy Formulation of Continuous Elements.
- Chapter 5. Finite Elements Method Applied to Dynamics.
- Chapter 6. Dynamical Systems Models.
- Chapter 7. Fatigue Failure Mechanism and Analysis Methods.

- Chapter 8. Dynamic Analysis of Structural Elements.
- Appendix. Matlab Routines.

José Carlos de Carvalho Pereira
Department of Mechanical Engineering
Federal University of Santa Catarina
Florianópolis, Brazil

DEDICATION

This book is dedicated to my sons Pedro Vitor and Arthur Gabriel, my wife, Andréa, and my parents Maria Aparecida (in Memorium) and Ozório (in Memorium).

CHAPTER 1

Introduction to the Dynamics of Mechanical Systems

Abstract: Within engineering, mechanical sciences were developed to interpret physical phenomena and describe them using mathematical models. These mathematical models represent important tools within the project area, which serve to predict certain behaviors. Particularly mechanical systems subjected to transient loads must be structurally designed in order to avoid failure modes associated with the alternation of mechanical stresses. This chapter will present some examples of mechanical systems that can be subjected to transient loads during their operation and what are the design criteria used to prevent their failures.

Keywords: Impulsive axial load, Mechanical systems, Product design; Random transverse load, Structural analysis, Transient torque, Transient rotary bending.

INTRODUCTION

The design of a product must meet certain requirements previously established during its initial phases, where the identification of its needs and the definition of its objectives are made. After the steps of synthesis and analysis of all the ideas presented and discussed, a selection of the best alternatives for the product is made. In this more advanced stage of product development, in the preliminary design stage, and perhaps even within the next detailed design stage, mathematical models of the product must be elaborated. These mathematical models must be representations of the physics of the problem described by the best alternatives of the product, and must produce results that allow the choice of the best alternative, or at least, that eliminate the less suitable ones. At this stage of the product design, it is very common to use CAD (Computer Aided Design) or CAE (Computer Aided Engineering) tools. The CAE tool is most often used when the product must meet safety factors or engineering design standards, such as: American National Standards Institute (ANSI), American Society of Mechanical Engineers (ASME), American Society of Testing and Materials (ASTM), International Standards Organization (ISO), and Society of Automotive Engineers (SAE), among others. After an accurate analysis of the reasonableness of the results obtained with the mathematical models in this preliminary design stage, and later, detailed design, it is recommended the manufacture product prototypes and tests for their qualification. Only after going through all these design stages, and with a critical look at all of them, will the product be able to be produced on an industrial scale.

José Carlos de Carvalho Pereira

Returning to the stage of the preliminary design of engineering products, it is important that the problem is very well formulated, that is, that the following are previously defined: a) the specific objectives to be achieved, b) the desired operating hypotheses, c) the data that can be assumed, d) the sketches chosen, e) the most suitable mathematical models to analyze the problem, *etc*. In the case of mechanical systems, within which machines, equipment, structures, mechanical components, complete systems, or sub-systems can be included, all must meet functionality and mechanical strength requirements. In order to evaluate the strength of a structural element, it is necessary to carry out a stress analysis based on an adequate formulation of the problem as reported above. With regard to the use of the most appropriate mathematical models, the identification of the movement or not of these mechanical systems, or the evaluation of the effect of the accelerations involved, is fundamental for the classification of the problem, between the static and dynamic analysis.

For the purpose of this book, the term dynamics will be used to define a load whose magnitude varies with time, keeping its direction and position constant. As a consequence of the action of a dynamic load on the mechanical systems, also called dynamic systems, they will respond dynamically, *i.e*., the response in terms of the displacements of their components will vary with time. Additionally, if the components of dynamic systems are considered elastic or deformable, the quantities related to displacement, such as strains, stresses, internal forces, *etc*., will vary with time. In view of the variation of stresses with time, it is then evident that the analysis step, within the design of a mechanical system that contains elastic elements, does not have a simple solution as in the case of a static analysis.

Depending on the functionality of the mechanical system, the dynamic load acting can be considered periodic or non-periodic. The primary purpose of this book is therefore the calculation of stresses in elastic components of mechanical systems on which a non-periodic dynamic load is applied. In a later step, the fatigue failure analysis of these components will be carried out, when applicable, using traditional criteria. Within this non-periodic load category, transient loads of short duration or random loads of long duration will be considered. To demonstrate the applicability of the procedure adopted in this book, mechanical systems with elastic elements subjected to transient torque, impulsive axial load, random transverse load and unbalanced mass in rotating machines operating in a transient regime will be analyzed. Examples of problems of the types mentioned above are solved using routines developed with the Matlab tool.

MECHANICAL SYSTEMS UNDER TRANSIENT TORQUE

Mechanical systems subjected to transient torque are sub-systems of more complex systems, such as energy generation systems (wind and steam turbines) and vehicles (cars, motorcycles, and bicycles). Examples of these types of mechanical systems are gearbox vehicle transmissions, gear trains of some wind turbines, gear reducers/multipliers-motors, gearboxes of rotating machines such as steam turbines, hydraulic pumps, and so on. This type of mechanical system is characterized by the torsional cycling loads of short-length shafts coupled together by means of gears, belts, *etc*. The description of its movement is made through the angular displacement variable or torsional angle $\theta(t)$ of each of its shafts, and its related variables: angular velocity $\dot{\theta}(t)$ and angular acceleration $\ddot{\theta}(t)$. Thus, the shear strain $\gamma(t)$ and the shear stress $\tau(t)$ can be determined. Figs. (**1** and **2**) illustrate examples of this type of mechanical system, which contain torsional movement of short-length drive train shafts such as wind turbines [1-3].

Fig. (1). Wind turbine plant (source: Secretariat for Economic Development and Tourism - Municipality of Agua Doce, Santa Catarina State, Brazil**).**

CHAPTER 2

Lagrange's Equations of Motion of Mechanical Systems

Abstract: This chapter discusses the concepts involved in the Lagrangian formulation to derive the differential equations of motion of mechanical systems. Here it is demonstrated how the equations of motion from scalar quantities such as work and energy can be obtained in contrast to Newton's second lawprevent their failures.

Keywords: D'Alembert's Principle, Generalized forces, Generalized coordinates, Hamilton's Principle, Lagrange's equations, Virtual works.

INTRODUCTION

For the study of particle dynamics, equations of motion are often obtained by applying Newton's laws of motion, or Newton's second law. As the forces acting on particles are vector quantities, obtaining these forces and their decomposition is not always a simple task. In more complex systems, there are other ways to obtain the equations of motion: Hamilton's Principle, Hamilton's Equations, D'Alembert's Principle, and the most common in the field of mechanical systems, Lagrange's equations, also known as Lagrangean formulation. For the derivation of the Lagrange's equations, some concepts must be presented, such as: generalized coordinates or degrees of freedom, constraints, virtual works, and generalized forces. The differential equations of motion of all mechanical systems treated in this book will be obtained from the Lagrangian formulation or the Lagrange's equations application over all forms of energy of the constituent elements, such as strain energy, kinetic energy and dissipative energy, as well as on the virtual work of the applied forces.

DEGREES OF FREEDOM

The number of equations describing the motion of a system of particles m_i ($i = 1, 2, 3, ..., N$) is equal to the number of degrees of freedom. The number of degrees of freedom (dof) is equal to the number of coordinates that specify the configuration of a system in space, in a plane, or in a direction, minus the number of constraints, or a number of independent equations that constrain its movement.

José Carlos de Carvalho Pereira

As an example of how it is possible to define the number of degrees of freedom, consider a system containing two particles of masses m_1 and m_2 ($N = 2$), positioned in space with coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2), as shown in Fig. (**1**). Particles are connected by a rigid bar of length L, which implies the following restriction equation [1-2]:

$$L^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 \qquad \textbf{(2.1)}$$

In this example, the number of coordinates is 6 and the number of constraints is 1. Therefore, the number of degrees of freedom is 5. Thus, the configuration of this system could be specified by the position of any point (x_1, y_1, z_1) or (x_2, y_2, z_2); or by the position of the gravity center (GC) (x_0, y_0, z_0), and additionally, by the spherical coordinates that indicate the orientation of the bar in space, θ and ϕ.

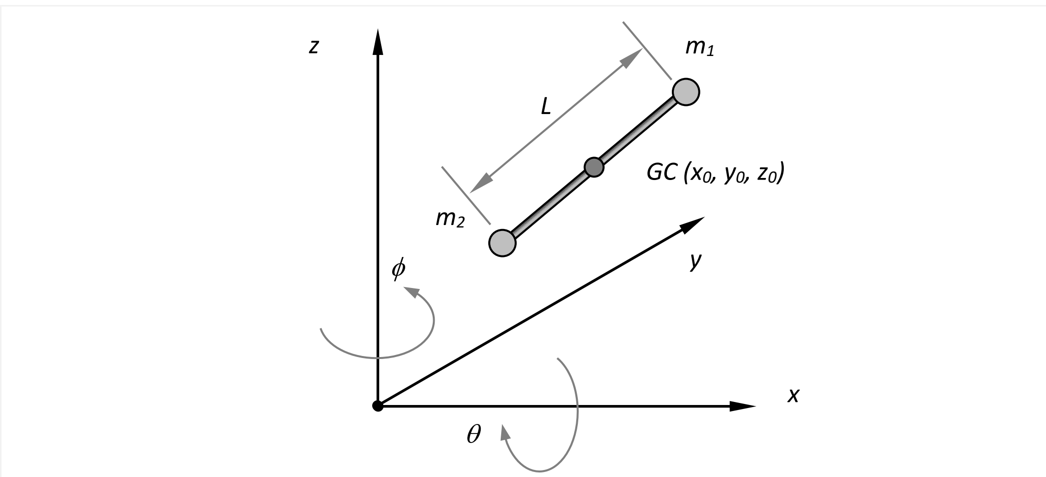


Fig. (1). System with two particles connected by a rigid bar.

GENERALIZED COORDINATES

As mentioned in Fig. (**1**) example, the description in space of the system's configuration is made by the cartesian coordinates or physical coordinates of the two masses of the system (x_1, y_1, z_1) and (x_2, y_2, z_2), minus the restriction imposed by the bar that unites them. As a result of this restriction, the system could be described by the cartesian coordinates of a point on the bar (x_0, y_0, z_0), plus the spherical coordinates given by θ and ϕ. Thus, in order to describe the system's configuration more generally, any independent variables that serve to describe its movement are called generalized coordinates p_i [1]. Consequently, the number of degrees of freedom is equal to the number of generalized coordinates.

In general, for a system with *N* particles moving in three-dimensional space, the number of generalized coordinates p_i, or independent variables, needed to describe the movement of this system is as:

$$p_i = 3N - R \quad i = 1,2,\cdots,n \tag{2.2}$$

where *R* is the number of constraints contained in the systems, or the number of equations that relate the coordinates to each other and n is the number of generalized coordinates.

CONSTRAINTS

According to Greeenwood [1] and Spiegel [3], constraints can be divided into holonomics and non-holonomics.

Holonomics Constrainsts

Constraints of the holonomics type, also called scleronomics, because they do not explicitly depend on time, relate the coordinates of the system's particles through equations that should not be violated when describing their configuration. These equations bring the number of generalized coordinates p_i, or degrees of freedom, to a number smaller than the coordinates of the particles $\left(x_i, y_i \textit{ and } z_i\right)$. A typical example of this restriction is shown in Fig. (**1**).

Non-holonomics Constrainsts

Constraints of the non-holonomics type, unlike holonomics constraints, are explicitly time-dependent. They relate the coordinates of the system particles $\left(x_i, y_i \textit{ and } z_i\right)$ through equations that should not be violated when describing their configuration at a given instant of time. However, they do not necessarily lead to the number of generalized coordinates p_i, to a number smaller than the coordinates of the particles. This type of constraints can be represented by a function of the type shown in eq. (2.3):

CHAPTER 3

Energy Formulation of Discrete Systems

Abstract: This chapter covers discrete systems composed of lumped masses, springs and dampers. The energy formulation of these elements from the application of the Lagrange equations, obtaining the differential equations of motion, is presented. The resolution of the equations of motion in the time domain by step-by-step methods such as the Taylor series, Newmark, Wilson and Houbolt is presented.

Keywords: Discrete systems, Degrees of freedom, Frequency domain response, Modal method, Newmark method, Time domain response.

INTRODUCTION

This chapter covers systems containing discrete elements, rigid masses joined by springs and dampers with 1, 2 and N degrees of freedom (dof). It intends to simply obtain the differential equations of motion through the Lagrangian formulation, evidencing the variables that describe the system's response. Its behavior is evaluated for different requests, as well as the effect of its components on the response. Notions of frequency domain response and time domain response are demonstrated. Special attention is given to obtaining the time domain response when four methods are presented and compared to each other. It will be shown that the time step is an important aspect of getting the response in the time domain accurately. The modal method, which can be used to solve problems with a large number of degrees of freedom, is also presented. To consolidate the knowledge acquired in this chapter, the resolutions of some exercises are carefully detailed, along with some proposed exercises to be solved.

SYSTEMS WITH 1 DEGREE OF FREEDOM

To understand the behavior of dynamically requested mechanical systems, it is initially necessary to study systems with 1 degree of freedom (1dof) as these are relatively simple to be modeled mathematically.

A system with 1 dof can be characterized by discrete elements, such as a mass considered rigid m, a viscous damping with constant c and an elastic spring of rigidity k as shown in Fig. (**1**) [1-5].

José Carlos de Carvalho Pereira

This system is excited by an external force, a function of time, *f(t)*. The answer of this system is given by the vertical displacement, also a function of time, *w(t)*, which is the only degree of freedom of the system.

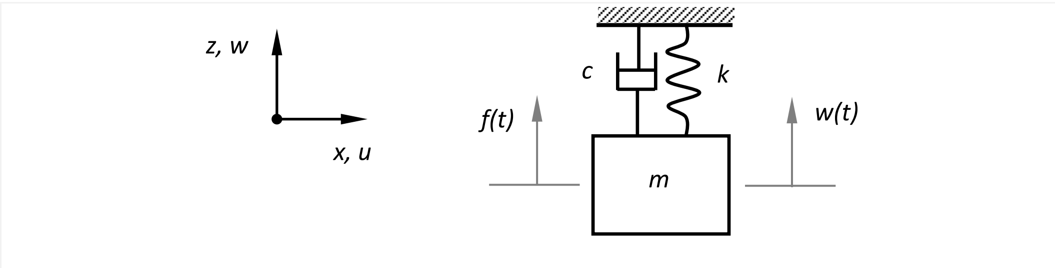


Fig. (1). One degree of freedom system.

The kinetic energy of the mass *T*, the energy dissipated by the damping *D* and the deformation energy of the spring *U* can be calculated as:

$$T = \frac{1}{2} m \dot{w}^2 \tag{3.1}$$

$$D = \frac{1}{2} c \dot{w}^2 \tag{3.2}$$

$$U = \frac{1}{2} k w^2 \tag{3.3}$$

Due to the applied force *f(t)*, the virtual work can be calculated as:

$$\delta W = f(t)\,\delta w \tag{3.4}$$

where δw is the virtual displacement of the mass *m*.

Derivation of the Equation of Motion in 1 dof Systems

The equation of motion of the mass of 1 dof system can be obtained from the application of the Lagrange equation.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{w}}\right) - \frac{\partial T}{\partial w} + \frac{\partial D}{\partial \dot{w}} + \frac{\partial U}{\partial w} = f(t) \tag{3.5}$$

Thus, applying eq. (3.5) on eqs. (3.1), (3.2) and (3.3), we have:

$$m\ddot{w}+c\dot{w}+kw=f(t) \qquad \textbf{(3.6)}$$

Forced Vibrations in 1 dof Systems

The mass of the 1 dof system can be excited by a harmonic force, the differential equation of motion being given by:

$$m\ddot{w}+c\dot{w}+kw=F_o\,sen\Omega t \qquad \textbf{(3.7)}$$

The solution of eq. (3.7) can be placed as:

$$w(t)=W\,sen\left(\Omega t-\beta\right) \qquad \textbf{(3.8)}$$

Where W is the vibration amplitude of mass m, and β is the phase or the delay of the displacement with respect to the excitation force.

Developing the eqs. (3.8), we have:

$$w(t)=W\left(sen\Omega t\cos\beta-\cos\Omega t\,sen\beta\right) \qquad \textbf{(3.9)}$$

Replacing eqs. (3.9) in the equation of motion, (3.7), we obtain:

$$W\left\{\begin{array}{l}\left[-m\Omega^2\,sen\Omega t\cos\beta+m\Omega^2\cos\Omega t\,sen\beta\right]+\\ \left[c\Omega\cos\Omega t\cos\beta+c\Omega\,sen\Omega t\,sen\beta\right]+\\ \left[k\,sen\Omega t\cos\beta-k\cos\Omega t\,sen\beta\right]\end{array}\right\}=F_o\,sen\Omega t \qquad \textbf{(3.10)}$$

Equating the terms in $\cos\Omega t$ and $sen\Omega t$, eqs. (3.10) are subdivided into

$$\begin{array}{l}W\left\{-m\Omega^2\cos\beta+c\Omega\,sen\beta+k\cos\beta\right\}sen\Omega t=F_o\,sen\Omega t\\ W\left\{m\Omega^2\,sen\beta+c\Omega\cos\beta-k\,sen\beta\right\}\cos\Omega t=0\end{array} \qquad \textbf{(3.11)}$$

As $W\neq 0$ and $\cos\Omega t\neq 0$, from eq. (3.11b) we have:

$$sen\beta=\frac{c\Omega}{\left(-m\Omega^2+k\right)}\cos\beta \qquad \textbf{(3.12)}$$

CHAPTER 4

Energy Formulation of Continuous Elements

Abstract: This chapter covers continuous elastic elements, such as bars, beams, and shafts required by axial and torsional efforts, bending and rotary bending. The formulations of kinetic energy and elastic deformation energy of these elements are presented from the application of the Lagrange's equations, obtaining the differential equations of motion.

Keywords: Bending, Kinetic energy, Rotary bending, Strain energy, Tension, Torsion, Work.

INTRODUCTION

This chapter covers elastic continuous elements, such as bars, beams, and axes submitted to axial, bending, and torsional efforts. The formulations of kinetic energy and elastic deformation of these elements are presented in the form of the application of the Lagrange equations, as well as the differential equations of motion [1-3]. The solution of these differential equations using functions that describe the deformed shape of these elements is performed. In the analysis of free vibrations, the natural frequencies of these continuous elements in different vibration modes and different boundary conditions are determined. In the analysis of forced vibrations, displacements and torsional angles are determined, in order to later determine deformations and stresses in the time domain. In addition, the formulations of shafts in rotation are presented to obtain the differential equations of motion, for later resolution of the problem.

BAR ELEMENTS UNDER TENSION

Energy and Work of Bars in Longitudinal Movement

The longitudinal movement of bars, described by variable u, function of space x and time t, can be obtained from the expressions of kinetic energy and strain energy of an infinitesimal element of length dx, cross-sectional area A and density ρ see Fig. (**1**).

José Carlos de Carvalho Pereira

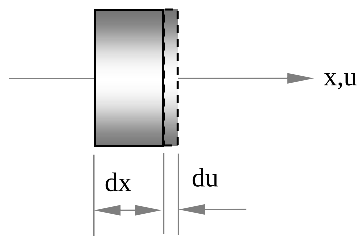


Fig. (1). Bar element in longitudinal movement**.**

The general expression for the strain energy for longitudinal movement is [1-2].

$$U = \frac{1}{2}\int_{V} \sigma_x \, \varepsilon_x \, dV \tag{4.1}$$

where the relation between normal stress and normal strain given by Hooke's Law, within the linear elastic regime, is $\sigma_x = E\,\varepsilon_x$ [3-4].

Thus, within the linear-elastic regime, the strain energy can be placed as:

$$U = \frac{1}{2}\int_{A}\int_{0}^{L} E\,\varepsilon_x^2 \, dA\,dx \tag{4.2}$$

For E and A, constants along the length of the bar element of length L, we have [1-2].

$$U = \frac{E\,A}{2}\int_{0}^{L}\left(\frac{\partial u}{\partial x}\right)^2 dx \tag{4.3}$$

The kinetic energy of an axis element with volumetric density ρ, cross-sectional area A and length L, in an element of infinitesimal volume dV and length dx, has the form [1-2].

$$T = \frac{1}{2}\int_{V} \dot{u}^2 \, \rho \, dV = \frac{1}{2}\int_{A}\int_{0}^{L} \dot{u}^2 \, \rho \, dA\,dx \tag{4.4}$$

For ρ and A, constants along the length of the bar element of length L, we have:

$$T = \frac{\rho A}{2} \int_0^L \dot{u}^2 dx \tag{4.5}$$

The virtual work due to a longitudinal force *f(t)* applied at the end of the bar can be calculated in this way [5-6].

$$\delta W = f(t)\,\delta u \tag{4.6}$$

Where δu is the virtual displacement at the point of application of the force *f(t)*.

Derivation of the Equation of Bars in Longitudinal Movement

It is observed that the displacement variable u in the expressions of strain energy and kinetic energy of the bar is a function dependent on time *t* and position *x*. To obtain the differential equation of motion, it is common to use a classical analytical method of separating variables as follows [5-8]:

$$u(x,t) = \phi(x)\, p(t) \tag{4.7}$$

The function $\phi(x)$ is called the spatial function, which must respect the boundary conditions of the problem, in addition to being derivable as many times as necessary. The function *p(t)*, called the temporal function, must represent the evolution of the displacement in time.

The equation of motion of a bar in longitudinal movement can be obtained from the application of the Lagrange equation, placed in the form [6].

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{p}}\right) - \frac{\partial T}{\partial p} + \frac{\partial D}{\partial \dot{p}} + \frac{\partial U}{\partial p} = F(t) \tag{4.8}$$

Thus, applying eq. (4.8) on eqs. (4.3), (4.5) and (4.6), there is:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{p}}\right) = \frac{d}{dt}\left[\frac{\partial}{\partial \dot{p}}\left(\frac{\rho A}{2}\int_0^L \dot{u}^2 dx\right)\right] = \frac{d}{dt}\left\{\frac{\partial}{\partial \dot{p}}\left[\frac{\rho A}{2}\int_0^L (\phi\, \dot{p})^2\, dx\right]\right\} = \rho A \int_0^L \phi^2 dx\, \ddot{p} \tag{4.9}$$

$$\frac{\partial U}{\partial p} = \frac{\partial}{\partial p}\left[\frac{E\,A}{2}\int_0^L \left(\frac{\partial u}{\partial x}\right)^2 dx\right] = E\,A \int_0^L \left(\frac{\partial \phi}{\partial x}\right)^2 dx\, p \tag{4.10}$$

CHAPTER 5

Finite Elements Method Applied to Dynamics

Abstract: This chapter will describe the procedure for obtaining the differential equations of motion resulting from the discretization of continuous bar elements under tension or torsion and beam elements under bending and rotary bending from the Lagrangian formulation.

Keywords: Bar elements, Bending, Lagrangian formulation, Beams elements, Rigid elements, Tension, Torsion.

INTRODUCTION

In the preliminary analysis and detailed analysis stages during the development of a product that must comply with engineering recommendations, CAE tools play a fundamental role. The advantage of CAE tools is that they address multiphysics problems and that they more accurately represent the application of a given product, as well as more accurately describe its geometric complexity. Within this context, commercial software has provided, in recent decades, an extraordinary advance in these stages of product development. This does not mean that homemade routines developed in scientific language to solve mathematical models of representing a physical problem are discarded. No matter what computing environment one is working in, numerical methods are used to solve mathematical models, and one of the most widely applied methods for solving engineering problems is the finite element method. Within the context proposed by this book, which is the analysis of mechanical systems subjected to dynamic loads and composed of elastic elements, the finite element method presents results in a very accurate way.

This chapter does not intend to present a survey on aspects related to the finite element methods but only describes the procedure for obtaining the elementary matrices of continuous bar elements under tension or under torsion, beam elements and rigid elements, from the lagrangian formulation. The differential equation of motion of a system is obtained by the sum of these elementary matrices. The solution of the differential equation in the time domain for the system operating in the transient regime provides the displacements and inclinations at the nodes of the elements, with which it is possible to calculate the mechanical stresses, and subsequently, apply the most appropriate failure theory as described in Chapters 6 and 8.

José Carlos de Carvalho Pereira

BAR ELEMENT

Element under Tension

Consider an element of a bar of length L, modulus of elasticity E, and cross section A and density ρ, as shown in Fig. (**1**). The two ends are denoted nodal points (or simply nodes) *1* and *2*. Forces (external to the element) P_1 and P_2, respectively, are acting on these nodes. Corresponding to these two forces, there are two axial displacements u_1 and u_2 called degrees of freedom [1-6].

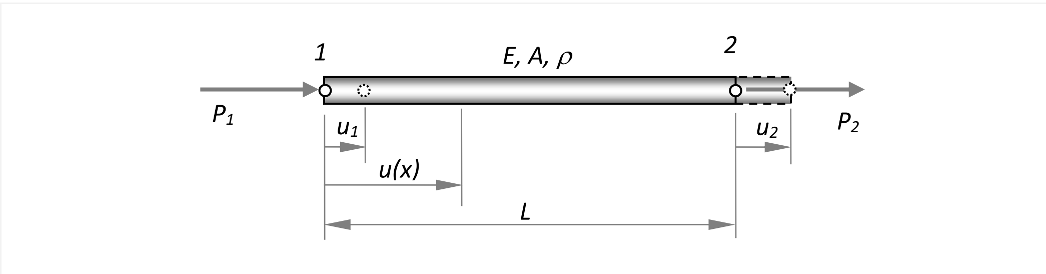


Fig. (1). Finite bar element under tension.

For an element with constant axial stress or constant axial strain, the axial displacement can be assumed to vary linearly in x is:

$$u(x) = a_1 + a_2 x \tag{5.1}$$

where a_1 and a_2 are constants to be determined by the imposition of boundary conditions:

$$\begin{aligned} &p / x = 0, \quad u(x) = u(0) = u_1 = a_1 \\ &p / x = L, \quad u(x) = u(L) = u_2 = a_1 + a_2 L \quad \Rightarrow \quad a_2 = \frac{u_2 - u_1}{L} \end{aligned} \tag{5.2}$$

Replacing the results of a_1 and a_2 from eq. (5.2) in eq. (5.1), the axial displacement can be written as:

$$u(x) = f_1(x) u_1 + f_2(x) u_2 \tag{5.3}$$

where $f_1(x)$ and $f_2(x)$ are the shape functions for bar elements, and are expressed as:

$$f_1(x) = 1 - \frac{x}{L}$$
$$f_2(x) = \frac{x}{L} \tag{5.4}$$

For the case of uniaxial stresses and strains, the strain is defined as:

$$\varepsilon = \frac{\partial u}{\partial x} \tag{5.5}$$

Replacing eqs. (5.3) and (5.4) in eq. (5.5) presents:

$$\varepsilon = \frac{\partial f_1(x)}{\partial x} u_1 + \frac{\partial f_2(x)}{\partial x} u_2 = f_1^{'}(x)\, u_1 + f_2^{'}(x)\, u_2 \tag{5.6}$$

The expression of strain energy for the case of axially loaded bars is shown in eq. (4.3) in Chapter 4, Section 4.1.

$$U = \frac{E\,A}{2} \int_0^L \left(\frac{\partial u}{\partial x} \right)^2 dx \tag{5.7}$$

Replacing eq. (5.6) in eq. (5.7) gives:

$$U = \frac{E\,A}{2} \int_0^L \left[f_1^{'}(x)\, u_1 + f_2^{'}(x)\, u_2 \right]^2 dx \tag{5.8}$$

The kinetic energy of a bar element of volume density ρ, cross-sectional area A and length L has the form shown in eq. (4.5) in Chapter 4, Section 4.1, is as:

$$T = \frac{\rho\,A}{2} \int_0^L \dot{u}^2 dx \tag{5.9}$$

Where the axial velocity is expressed in the form:

$$\dot{u}(x) = f_1(x)\dot{u}_1 + f_2(x)\dot{u}_2 \tag{5.10}$$

Applying Lagrange's equations on the expression of kinetic energy, we have:

CHAPTER 6

Dynamic System Models

Abstract: In this chapter, the development of models of more complex mechanical systems is developed, which have continuous elastic elements joined by elastic spring and viscous damper elements. As will be observed, these mechanical systems can be easily identified with the suspensions of bicycles, motorcycles and cars, gear trains and rotating machines.

Keywords: Continuous elements, Complex mechanical systems, Frequency domain response, Time domain response.

INTRODUCTION

Thus far, the procedures necessary for the development of discrete system models to obtain their response to any requests have been presented in Chapter 3. Moreover, model development with continuous elements as well as proposal presentations for approximate solutions through continuous functions and formulation discretized by finite elements, such as bars, beams, and rotary beams have also been described in Chapter 5. From this chapter onward, it will be possible to develop models of more complex mechanical systems, which have continuous elastic elements joined by elastic springs and viscous damper elements, in addition to lumped mass and rigid elements. As can be seen, these mechanical systems can be easily identified with gear trains, suspensions of bicycles, motorcycles, and cars, and rotating machines. The resolution of the equations of motion is presented in the frequency domain and in the time domain through the Newmark method. As presented in Chapter 3, it is important to determine the first natural frequency of the system in order to stipulate a suitable time step when calculating the time domain response.

SYSTEMS WITH ELEMENTS IN LONGITUDINAL MOVEMENT

Mechanical systems with elements in longitudinal movement are found in the front suspensions of bicycles and motorcycles, similar to the one shown in Figs. (**1** and **8)** into Chapter 1 [1].

The mathematical models commonly developed to describe the behavior of these systems are constituted by continuous bar elements, elastic springs, viscous dampers and lumped masses.

Example 6.1 – Consider a mechanical system in longitudinal movement composed of a bar and a spring, as shown in Fig. (**1**). The properties of the bar are *E*, *A*, *L* and ρ; and the stiffness of the spring is *k*. Determine the natural frequencies of this system using the finite element method. Likewise, we plot the frequency domain response when the force *f(t)* is harmonic of amplitude F_o and frequency Ω.

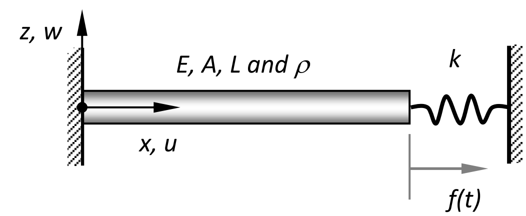


Fig. (1). Mechanical system with bar and spring in longitudinal movement.

Fig. (**2**) shows the finite element mesh of this mechanical system with the bar and spring in longitudinal movement with 1-bar element**.**

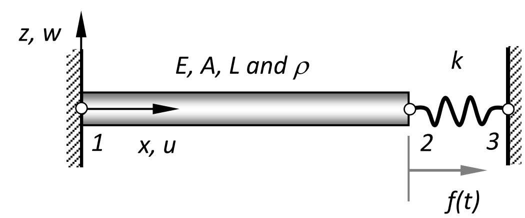


Fig. (2). The finite element mesh with 1-bar element under tension.

The elementary mass and elementary stiffness matrices of element 1-2 were presented previously in Chapter 5, Section 5.2.

The deformation energy due to the spring placed between nodes *2* and *3* is:

$$U = \frac{1}{2} k \left(u_3 - u_2 \right)^2 \tag{6.1}$$

Virtual work due to external force applied at node *2* is expressed as:

$$\delta W = f(t) \delta u_2(t) \tag{6.2}$$

Applying Lagrange's equations to the deformation energy of the spring and virtual work:

$$\frac{\partial U}{\partial u_2} = k\left(u_2 - u_3\right), \qquad \frac{\partial U}{\partial u_3} = k\left(-u_2 + u_3\right) \\ F(t) = f(t) \tag{6.3}$$

Thus, the equations of motion of this discretized system with 3 nodes after the assemblage are written as:

$$\frac{\rho A L}{6}\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{Bmatrix} + \frac{E A}{L}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1+\dfrac{L}{E A}k & -\dfrac{L}{E A}k \\ 0 & -\dfrac{L}{E A}k & \dfrac{L}{E A}k \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} R_1(t) \\ f(t) \\ R_3(t) \end{Bmatrix} \tag{6.4}$$

Where $R_1(t)$ and $R_3(t)$ are reactions on the supports on nodes *1* and *3*. Applying the boundary conditions, $u_1 = 0$ and $u_3 = 0$, the equations of motion of this system become,

$$\frac{\rho A L}{6}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{Bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{Bmatrix} + \frac{E A}{L}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1+\dfrac{L}{E A}k & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ f(t) \\ 0 \end{Bmatrix} \tag{6.5}$$

Therefore, eq. (6.5) reduces to,

$$\frac{\rho A L}{3}\ddot{u}_2 + \frac{E A}{L}\left(1+\frac{L}{E A}k\right)u_2 = f(t) \tag{6.6}$$

The natural frequency of the discretized system with 3 nodes is obtained by assuming the harmonic response as stated in eq. (5.69), in Chapter 5, Section 5.6. Hence:

CHAPTER 7

Fatigue Failure Mechanism and Analysis Methods

Abstract: In this chapter, a brief review of the basic concepts of fatigue failure mechanism in structural elements requested by cyclic loading is presented. Besides, tests to estimate the fatigue resistance curves of materials for different loading conditions, as well as how these curves can be used to design parts are demonstrated.

Keywords: Crack, Fatigue failure, Fatigue strength, Failure mechanism, Stress concentration.

INTRODUCTION

In the previous sections, it was possible to observe some situations of mechanical components subjected to cycling loads and/or inertial efforts (also known as body loads) so that the deformations, and consequently, the mechanical stress are time functions. Therefore, it is concluded that, for the design of these components, an analysis of fatigue strength is necessary. The universally accepted definition of fatigue failure is the process of progressive and cumulate microscopic, localized damage at a notched surface of the mechanical component due to successive loading cycles, which lead to a gradual reduction in the resistance capacity of its material, resulting in an abrupt rupture. Fatigue failure can start from a small crack, already present in the mechanical component, due to manufacturing processes, or it can start in a stress concentration region (notches) from a process called crack nucleation, which will be described later. Since Wöhler's investigation of catastrophic failures reported in his 1870 article, it is suspected that 80% to 90% of failures in mechanical components originated in the fatigue failure mechanism. A characteristic of this type of failure is that the stress level that leads to static failure combined with a number of cycles that reach the order of dozens or hundreds of millions.

This chapter will briefly present the fatigue damage mechanism and how the fatigue strength of materials can be obtained through experimental tests. It will also present how it is possible to estimate the fatigue strength curve of materials for different loading conditions, as well as how these curves can be used for the design of mechanical components. This book is not intended to provide a detailed description of the fatigue failure mechanism. Therefore, it is recommended to read classic textbooks. However, some fundamental concepts about fatigue failure and criteria

José Carlos de Carvalho Pereira

adopted for the design of mechanical components will be reviewed in the next sections.

FATIGUE DAMAGE MECHANISM

The crack initiation mechanism is called crack nucleation, and it occurs from the cyclic movement of the maximum shear stresses acting on the free surface of a mechanical component. Once the shear material strength of the component is overcome, slip planes are generated and oriented according to the directions of these maximum shear stresses. The plastic deformations generated by this cyclic mechanism preferentially start at 45° of the applied load direction, as well as in the more unfavorably oriented grains. These cyclic slips form on the free surface of the component slip bands, creating intrusions and extrusions [1]. These intrusions and extrusions, as seen in Fig. (**1**), show a wedge effect that "opens" the crack and, due to the cyclic movement, gradually propagates to the interior of the component.

The crack nucleation mechanism is very sensitive to surface topography (due to manufacturing processes), residual stresses (tractive is harmful and compressive is beneficial), environmental aggressiveness (accelerated crack nucleation), and other factors.

The fatigue damage mechanism can be divided into 3 phases see Figs. (**2** and **3**) [1-2]:

○**Phase I:** where the nucleation process and microscopic crack propagation occur, predominantly, because of the shear stresses,

○**Phase II:** where crack propagation occurs macroscopically, mainly because of tractive stresses,

○**Phase III:** instantaneous rupture of the component.

Generally, more than 70% of life is spent on the nucleation and propagation of the crack in phase I, with the remainder of life used for its propagation in phase II. Due to the microscopic characteristic in phase I, this process is very sensitive to local differences in microstructure, that is, to the localized properties of the material (changes in the direction of the crystalline planes, grain boundaries, *etc*.). In phase II, as the propagation occurs macroscopically, it is the average properties of the material that are relevant.

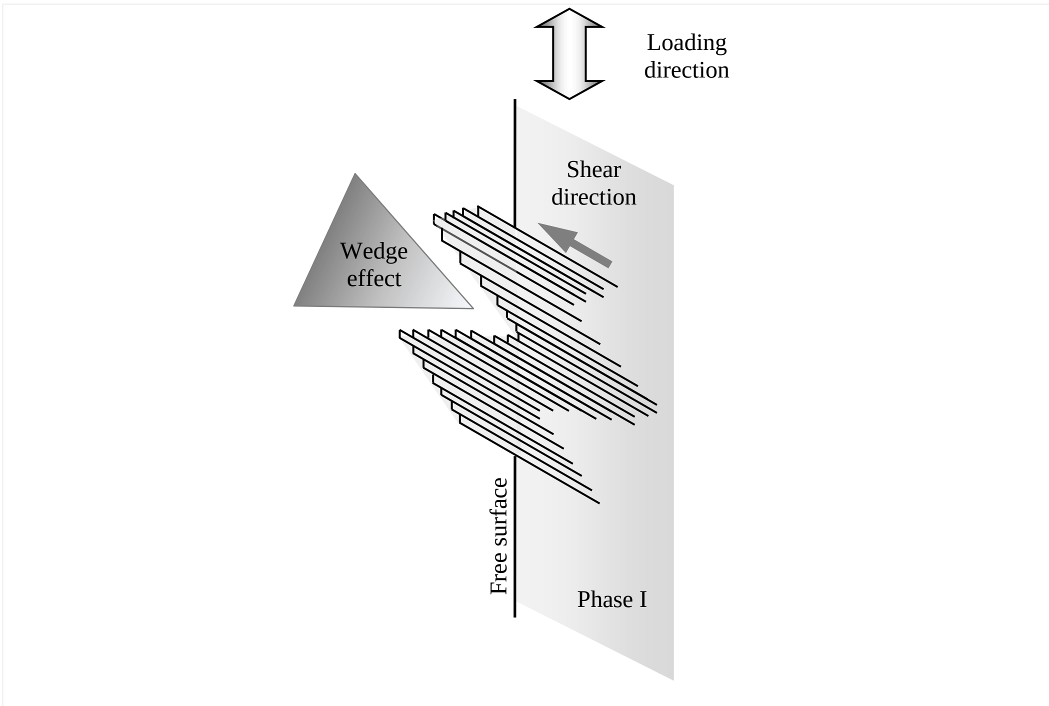


Fig. (1). Slipping bands formed by a cyclical load on a free surface.

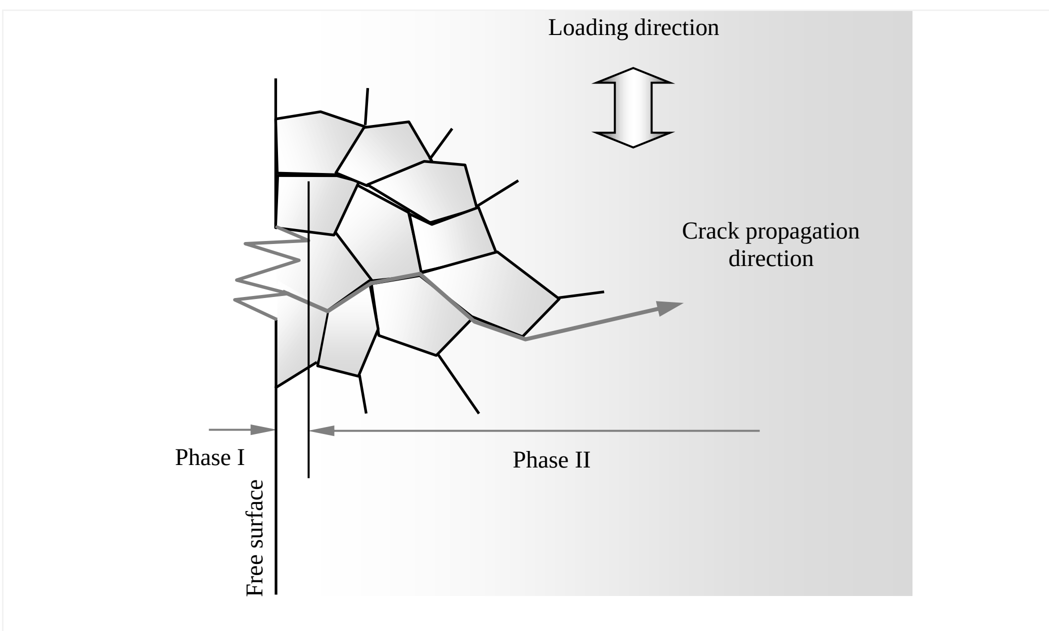


Fig. (2). Phases of Fatigue Crack Propagation.

CHAPTER 8

Dynamic Analysis of Structural Elements

Abstract: This chapter aims to apply the concepts of dynamics and fatigue strength analysis to predict the behavior of dynamically requested mechanical systems, and to design the structural elastic elements that constitute them. This procedure is applied to systems with torsional cycling loads, such as gear trains, systems with transversal cycling loads, such as vehicle suspension, and systems with rotary bending, such as turbines and pumps.

Keywords: Gear train, Rotary bending, Transient regime, Transversal cycling loads, Vehicle suspension.

INTRODUCTION

In Chapter 6, more complex models of mechanical systems were developed, which have continuous elastic elements joined by elastic springs and viscous damper elements. The procedure for the basis of mechanical system models was presented along with their resolution using approximate finite element methods, either with respect to the degrees of freedom, or with the acting stress. In Chapter 7, the adopted fatigue failure mechanism, and the failure criteria were presented to predict, and consequently prevent its occurrence.

This present Chapter 8 aims to apply the concepts of mechanical systems dynamic modeling and fatigue failure analysis, seen earlier, in order to design elastic elements that compose them. Since the interest is in the calculation of stresses resulting from transient loading, this chapter will apply these concepts to systems with torsional cycling loads, such as gear trains; systems with transversal cycling loads, such as suspension systems; and systems with rotary bending, such as turbines and pumps.

SYSTEMS WITH SHAFT UNDER TRANSIENT TORSION

In this section, mechanical systems containing shafts subjected to torsional transient efforts will be analyzed [1-5]. Particularly during start-up and shutdown or short-circuit conditions of rotating machines, variables such as acceleration, nominal rotation, and transient torque intensity can generate an alternating shear stress level in shafts that can reduce of the mechanical system life.

José Carlos de Carvalho Pereira

A typical transient torque during the start-up is related through eq. (8.1) and illustrated in Fig. (**1**).

$$T_{st}\left(t\right)=T_o\left[a+b\,e^{-\chi t}sen\left(\omega t+\alpha\right)\right] \tag{8.1}$$

In eq. (8.1), T_o is the nominal torque, a, b and χ are known transient constants, ω is the frequency of electrical current and α is the delay between the electric current and torque.

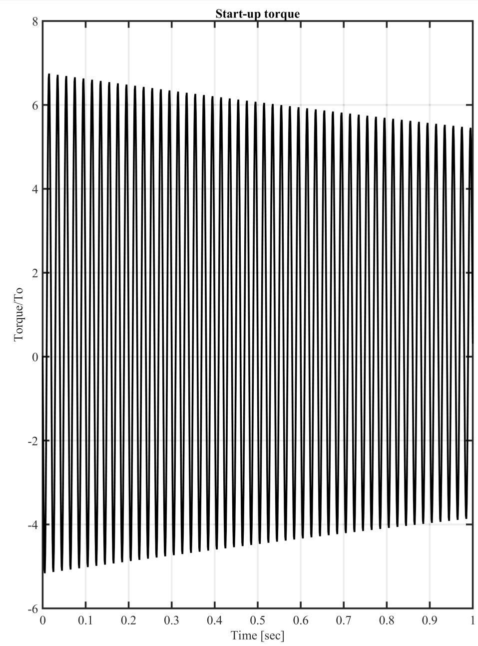


Fig. (1). Transient torque during start-up.

Fig. (**2**) illustrates the resulting alternating shear stress during start-up. As observed, the shear stress resulting from the start-up torque are formed by stress cycling blocks that replicate over time, whose stress level is gradually reduced.

In turn, a typical transient torque during the short circuit is given by eq. (8.2) and illustrated in Fig. (**3**). The first term is related to the air gap torque and the second term is the armature torque [4-5].

$$T_{sh}\left(t\right)=T_o\,m_a\,e^{-\alpha t}sen\,\omega t+T_o\,m_s\,e^{-\beta t}sen\,2\omega t \tag{8.2}$$

In eq. (8.2), T_o is the nominal torque, m_a, α, m_s and β are known transient constants and ω is the frequency of the electrical current.

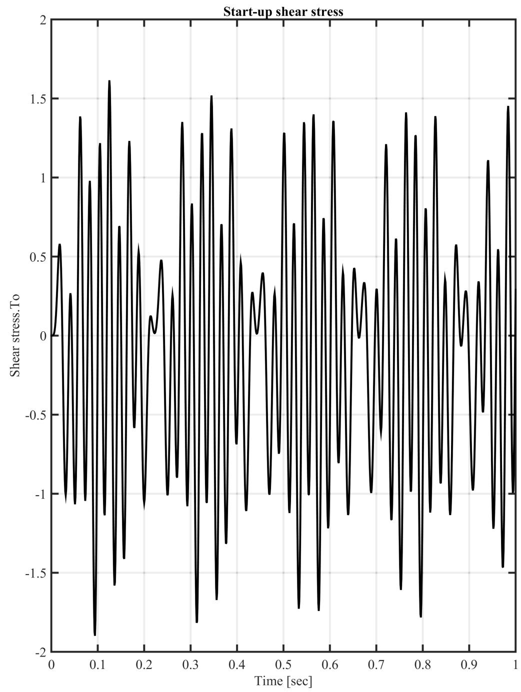


Fig. (2). Shear stress during start-up.

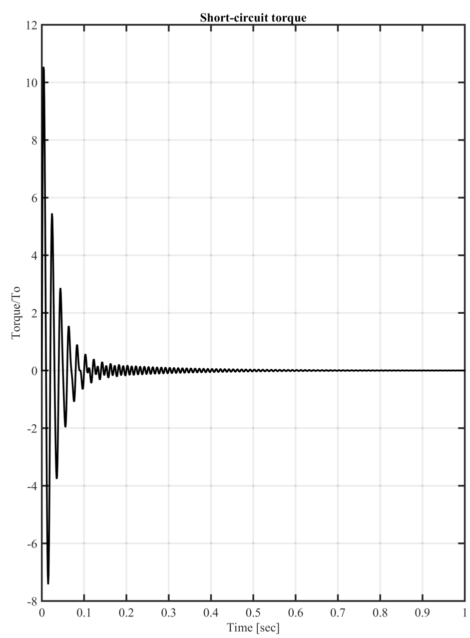


Fig. (3). Transient torque during short-circuit.

APPENDIX

Matlab Routines

Abstract: In this appendix, routines developed with the Matlab tool will be offered for some problems solved in Chapters 3, 6 and 8.

Keywords: Matlab; routines; frequency domain response; time domain response

INTRODUCTION

Some of the Matlab routines presented in this appendix refer to the frequency domain response and others to the time domain response of the examples seen in Chapters 3, 6, and 8. Readers who have some affinity with programming in the Matlab environment will be able to build their routines based on the routines offered in this appendix.

CHAPTER 3

Example 3.7 – Frequency domain response of a system with 2 dofs by the Direct Method.

```
%==========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 06 January 2022
%==========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%--------------------------------------------------------------------------
--
%    PURPOSE
%      Compute frequency response of a 2 GDL system for a harmonic force:
%      Direct Method
%
%    INPUT:
%          m    [kg]      mass
%          c    [N/m/s]   damping
%          k    [N/m]     stiffness
%          f1   [N]       amplitude of the force on mass m1
```

```
%          f2   [N]       amplitude of the force on mass m2
%          fi   [hz]      initial frequency
%          ff   [hz]      final frequency
%          df   [hz]      frequency step
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Amplitude [m] x frequency [rad/sec]
%-----------------------------------------------------------------------
--
%   NOTE: (1) Example 3.7
%-----------------------------------------------------------------------
--
clear
clf
% Data
%-----------------------------------------------------------------------
--
% Mass properties
m1 = 1.0;
m2 = 1.0;
% Damping properties
c1 = 1.0;
c2 = 0.0;
% Stiffness properties
k1 = 100.0;
k2 = 100.0;
% Amplitude of the forces
f1 = 0;
f2 = 10;
% Frequency data
fi = 0;
ff = 100;
df = 0.01;
% DOF number
ndof = 2;
% Compute the total number of points
nt = (ff-fi)/df;
% General matrices inicialization
freq = zeros(nt,1);
F = zeros(nt,ndof);
Re = zeros(nt,ndof);
Im = zeros(nt,ndof);
W = zeros(nt,ndof);
%-----------------------------------------------------------------------
--
% Assembly of global matrices
M = [m1 0
     0 m2];
C = [c1+c2 -c2
     -c2    c2];
K = [k1+k2 -k2
     -k2    k2];
Fo = [f1
```

```
     f2];
%-----------------------------------------------------------------------
--
     for i = 1:nt
     freq(i) = i*df;
     A(1,1) = -m1*freq(i)^2+k1+k2;
     A(1,2) = -k2;
     A(2,1) = -k2;
     A(2,2) = -m2*freq(i)^2+k2;
     W(i,:) = inv(A)*Fo;
     d_w_1(i) = abs(W(i,1));
     d_w_2(i) = abs(W(i,2));
     end
%-----------------------------------------------------------------------
--
%   PLOTs
%-----------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];   % black
    color_curve2 = [ 1.0 0.0 0.0];   % red
    color_curve3 = [ 0.0 1.0 0.0];   % green
    color_curve4 = [ 0.0 0.0 1.0];   % blue
    color_curve5 = [ 1.0 0.0 1.0];   % magenta
    linestyle1 = '-';   % Solid line
    linestyle2 = '--';  % Dashed line
    linestyle3 = ':';   % Dotted line
    linestyle4 = '-.';  % Dash-dot line
    linestyle5 = ':';  % Dash-dot line
    %-------------------------------------------------------------------
--
    figure (1)

semilogy(freq,d_w_1,'LineStyle',linestyle1,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    grid on

    title('Frequency        domain        response        -        mass
1','FontName',font2,'FontWeight','bold', 'FontSize',fontsize_title)
    xlabel('Frequency                        [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Amplitude [m]','FontName',font2, 'FontSize',fontsize_label)
    %-------------------------------------------------------------------
--
```

```
    figure (2)

semilogy(freq,d_w_2,'LineStyle',linestyle1,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    grid on

    title('Frequency         domain         response         -         mass
2','FontName',font2,'FontWeight','bold', 'FontSize',fontsize_title)
    xlabel('Frequency                       [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Amplitude [m]','FontName',font2, 'FontSize',fontsize_label)
    %---------------------------------------------------------------------
--
```

Example 3.8 – Time domain response of a system with 2 dofs by the Direct Method.

```
%=====================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 06 January 2022
%=====================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University od Santa Catarina - UFSC
%                Florianópolis - Brazil
%---------------------------------------------------------------------
--
%    PURPOSE
%      Compute time response of a 2 GDL system for a harmonic force:
%      Direct Method
%
%    INPUT:
%          m    [kg]      mass
%          c    [N/m/s]   damping
%          k    [N/m]     stiffness
%          f1   [N]       amplitude of the force on mass m1
%          f2   [N]       amplitude of the force on mass m2
%          fe   [rad/sec] excitation frequency
%          ti   [hz]      initial time
%          tf   [hz]      final time
%          dt   [hz]      time step
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%---------------------------------------------------------------------
--
%   NOTE: (1) Example 3.8
```

```
%-----------------------------------------------------------------------
--
clear
clf
% Data
%-----------------------------------------------------------------------
--
% Mass properties
m1 = 1.0;
m2 = 1.0;
% Damping properties
c1 = 1.0;
c2 = 0.0;
% Stiffness properties
k1 = 100.0;
k2 = 100.0;
% Amplitude of the forces
f1 = 0;
f2 = 10;
% Start-up conditions
Omo = 20.0;   % nominal angular velocity
fa = 2.0;     % inclination of the angular velocity
% Frequency response
nf1 = 6.18;    % first natural frequency
% Time data
ti = 0;
tf = 40;
npc = 50;      % number of points into a cycle
dt = 1/(nf1/2/pi)/npc;
% DOF number
ndof = 2;
% Compute the total number of points
nt = floor(tf/dt)+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-----------------------------------------------------------------------
--
% Assembly of global matrices
M = [m1 0
     0 m2];
C = [c1+c2 -c2
     -c2    c2];
K = [k1+k2 -k2
     -k2    k2];
Fo = [f1 f2];
%-----------------------------------------------------------------------
--
% Numerical integration - Taylor's method (1)
%-----------------------------------------------------------------------
--
%   Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = 6*M/dt^2+3*C/dt+K;
%-----------------------------------------------------------------------
--
```

```
% Numerical integration - Newmark's method (2)
beta = 1/4;
gama = 1/2;
%-----------------------------------------------------------------------
--
% Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%-----------------------------------------------------------------------
--
% Initialization of matrices
time = zeros(nt,1);        %   time vector
F = zeros(1,ndof);         %   force
a1 = zeros(1,ndof);        %   auxiliar vector a1
a2 = zeros(1,ndof);        %   auxiliar vector a2
displac = zeros(1,ndof);   %
veloc = zeros(1,ndof);     %
accel = zeros(1,ndof);     %
d_w_2 = zeros(nt,ndof);    %   displacement
v_w_2 = zeros(nt,ndof);    %   velocity
a_w_2 = zeros(nt,ndof);    %   aceleration
%-----------------------------------------------------------------------
--
% Loop for nt total numbers of points
     for i = 2:nt
        t = ti+(i-1)*dt;
        time(i) = t;
        Om(i) = Omo*(1-exp(-fa*t/Omo));
        F = Fo*sin(Om(i)*t);

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;

        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

     end
     d_max_1 = max(d_w_2(:,1))
     d_max_2 = max(d_w_2(:,2))
     v_max_1 = max(v_w_2(:,1))
     v_max_2 = max(v_w_2(:,2))
     a_max_1 = max(a_w_2(:,1))
     a_max_2 = max(a_w_2(:,2))
%-----------------------------------------------------------------------
--
%   PLOTs
```

```
%-----------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = ':';  % Dash-dot line
%-----------------------------------------------------------------------
    figure (1)


plot(time,d_w_2(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,2),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Displacement                              [m]','FontName',font2,
'FontSize',fontsize_label)
    legend('mass 1', 'mass 2')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (2)


plot(time,Om,'LineStyle',linestyle1,'Color',color_curve1,'LineWidth',line
width_curve);
    grid on

    title('Excitation     frequency','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
```

```
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Frequency                         [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    %-----------------------------------------------------------------------
--
```

CHAPTER 6

Example 6.2 – Frequency domain response of a system with 1 bar element and shock absorber element by the Direct Method.

```
%=========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%=========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University od Santa Catarina - UFSC
%                Florianópolis - Brazil
%-------------------------------------------------------------------------
--
%    PURPOSE
%      Compute frequency response of a longitudinal movement system by FEM:
%       1 bar element and shock absorber element - Direct Method
%
%    INPUT:
%          ro   [kg/m3]   mass density
%          A    [m2]      cross section area
%          L    [m]       lenght of the bar
%          E    [N/m2]    Young's modulus
%          k    [N/m]     stiffness of the spring
%          c    [N/m/s]   viscous damping
%          lm   [kg]      lumped mass
%          f    [N]       amplitude of the force on the bar
%          fi   [hz]      initial frequency
%          ff   [hz]      final frequency
%          df   [hz]      frequency step
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%-------------------------------------------------------------------------
--
%   NOTE:  (1) Example 6.2
%-------------------------------------------------------------------------
--
clear
```

```
clf
% Data
%-----------------------------------------------------------------------
--
% Beams material properties
ro = 7800;
E = 200.0e9;
% Bar cross section property
A = 1.13e-4;
% Bar lenght
L = 0.4;
% Stiffness property
k = 1.0e4;
% Viscous damping property
c = 20;
% Lumped mass
lm = 2;
% External excitation
f = 1000;
% Frequency data
fi = 0;
ff = 2000;
df = 1;
% DOF number
ndof = 3;
% Compute the total number of points
nt = (ff-fi)/df;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
% General matrices inicialization
freq = zeros(nt,1);
F = zeros(nt,ndof);
U = zeros(nt,ndof);
% Boundary conditions
u1 = 0;
% Constants of the beam element matrices
mb = ro*A*L/6;
kb = E*A/L;
%-----------------------------------------------------------------------
--
% Assemblage of global matrices after boundary conditions application
M = [1      0     0
     0    2*mb    0
     0      0    lm];
C = [0      0     0
     0      c    -c
     0     -c     c];
K = [1      0     0
     0    kb+k   -k
     0     -k     k];
Fo = [0 0 f];
 %----------------------------------------------------------------------
---
     for i = 1:nt
```

```
     freq(i) = fi+(i-1)*df;
     KM = K-freq(i)^2*M;
     IKM = inv(KM);
     U(i,:) = abs(Fo*IKM);
     end
%-------------------------------------------------------------------------
--
%   PLOTs
%-------------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = ':';  % Dash-dot line
    %--------------------------------------------------------------------
--
    figure (1)

semilogy(freq,U(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

semilogy(freq,U(:,2),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

semilogy(freq,U(:,3),'LineStyle',linestyle3,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Frequency                                              domain
response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Frequency                        [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Amplitude [m]','FontName',font2, 'FontSize',fontsize_label)
```

```
    legend('node 1', 'node 2', 'node 3')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %----------------------------------------------------------------------
--
```

Example 6.4 – Time domain response of a system in transversal movement with 2 beam elements and a shock absorber element by the Direct Method.

```
%=========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%=========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%-------------------------------------------------------------------------
--
%    PURPOSE
%      Compute time response of a transversal movement system by FEM:
%      2 beam element - Direct Method
%
%    INPUT:
%          ro   [kg/m3]   mass density
%          A    [m2]      cross section area
%          L    [m]       lenght of the beam
%          E    [N/m2]    Young's modulus
%          h    [m]       height of the cross section
%          Iy   [m4]      inertia moment of the cross section
%          k1   [N/m]     stiffness of the spring 1
%          k2   [N/m]     stiffness of the spring 2
%          c    [N/m/s]   viscous damping
%          lm   [kg]      lumped mass
%          fo   [N]       amplitude of the ramp_type force
%          to   [sec]     time for constant ramp force
%          ti   [hz]      initial time
%          tf   [hz]      final time
%          dt   [hz]      time step
%          nslx []        node stress location in x direction
%          nslz []        node stress location in z direction
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%-------------------------------------------------------------------------
--
%   NOTE: (1) Example 6.4
%         (2) Half-wave type force
```

```
%         (3) Newmark's method
%-----------------------------------------------------------------------
--
clear
clf
% Data
%-----------------------------------------------------------------------
--
% Beams material properties
ro = 7800;
E = 200.0e9;
% Beams cross section properties
A = 1.0e-3;
h = 100.0e-3;
Iy = 3.33e-8;
% Beams lenght
L = 0.5;
% Stiffness properties
k1 = 1.0e3;
k2 = 1.0e3;
% Viscous damping property
c = 200;
% Lumped mass
lm = 20;
% External excitation
fo = 100;
to = 1.0;
% Frequency response
nf1 = 300.6;    % first natural frequency
% Time data
ti = 0;
tf = 5.0;
npc = 50;      % number of points into a cycle
dt = 1/(nf1/2/pi)/npc;
% Point coordinates to calculate stress
nslx = L;
nslz = h/2;
% DOF number
ndof = 8;
% Compute the total number of points
nt = floor(tf/dt)+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-----------------------------------------------------------------------
--
% Boundary conditions
w1 = 0;
w4 = 0;
w5 = 0;
% Constants of the beam element matrices
mb = ro*A*L/420;
kb = E*Iy/L^3;
%-----------------------------------------------------------------------
--
```

```
% Assemblage of global matrices after boundary conditions application
M = [1      0          0        0            0            0      0   0
     0 4*L^2*mb 13*L*mb -3*L^2*mb         0            0      0   0
     0  13*L*mb 2*156*mb    0          54*mb      -13*L*mb  0   0
     0 -3*L^2*mb    0    2*4*L^2*mb     13*mb     -3*L^2*mb  0   0
     0      0     54*mb   13*L*mb   (156*mb+lm)   -22*L*mb  0   0
     0      0   -13*L*mb -3*L^2*mb   -22*L*mb     4*L^2*mb  0   0
     0      0       0        0           0            0      1   0
     0      0       0        0           0            0      0   1];
C = [0      0       0        0           0            0      0   0
     0      0       0        0           0            0      0   0
     0      0       c        0           0            0      0   0
     0      0       0        0           0            0      0   0
     0      0       0        0           0            0      0   0
     0      0       0        0           0            0      0   0
     0      0       0        0           0            0      1   0
     0      0       0        0           0            0      0   0];
K = [1      0       0        0           0            0      0   0
     0 4*L^2*kb -6*L*kb 2*L^2*kb         0            0      0   0
     0 -6*L*kb (24*kb+k1)  0          -12*kb       6*L*kb    0   0
     0 2*L^2*kb     0    8*L^2*kb    -6*L*kb      2*L^2*kb   0   0
     0      0    -12*kb   -6*L*kb   (12*kb+k2)    -6*L*kb   0   0
     0      0    6*L*kb 2*L^2*kb     -6*L*kb      4*L^2*kb  0   0
     0      0      0        0           0            0      1   0
     0      0      0        0           0            0      0   1];
Fo = [0     0      0        0           fo           0      0   0];
%-----------------------------------------------------------------------
--
% Numerical integration - Newmark's method (2)
beta = 1/4;
gama = 1/2;
%-----------------------------------------------------------------------
--
%   Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%-----------------------------------------------------------------------
--
% Initialization of matrices
F = zeros(1,ndof);          %   force
a1 = zeros(1,ndof);         %   auxiliar vector a1
a2 = zeros(1,ndof);         %   auxiliar vector a2
displac = zeros(1,ndof);    %
veloc = zeros(1,ndof);      %
accel = zeros(1,ndof);      %
d_w_2 = zeros(nt,ndof);     %   displacement
v_w_2 = zeros(nt,ndof);     %   velocity
a_w_2 = zeros(nt,ndof);     %   aceleration

% Loop for nt total numbers of points
     for i = 2:nt
         t = ti+(i-1)*dt;
        time(i) = t;
        if  (t<=to)
          F = Fo*sin(pi*t/to);
```

```
        else
          F = 0.0;
        end

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;

        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

    end
%------------------------------------------------------------------------
--
% Calcule normal stress
S_x = zeros(nt,ndof);      %   normal stress

df1 = (-6/L^2 + 12/L^3*nslx);
df2 = (-4/L + 6/L^2*nslx);
df3 = (6/L^2 - 12/L^3*nslx);
df4 = (-2/L + 6/L^2*nslx);

% Loop for nt total numbers of points
     for i = 2:nt
%       node 2
        S_x(i,1)    =    E*nslz*(df1*d_w_2(i,1)    +    df2*d_w_2(i,2)    +
df3*d_w_2(i,3) + df4*d_w_2(i,4)); %   Pa
        S_x(i,1) = S_x(i,1)/1.0e6;       %   MPa
%       node 3
        S_x(i,2)    =    E*nslz*(df1*d_w_2(i,3)    +    df2*d_w_2(i,4)    +
df3*d_w_2(i,5) + df4*d_w_2(i,6)); %   Pa
        S_x(i,2) = S_x(i,1)/1.0e6;       %   MPa
     end
%------------------------------------------------------------------------
--
     % Calcule reaction on supports
React = zeros(nt,ndof);      %   reaction in supports

% Loop for nt total numbers of points
     for i = 2:nt
%       node 1
        React(i,1)    =    mb*(54*a_w_2(i,3)-13*L*a_w_2(i,4))    +    kb*(-
12*d_w_2(i,3)-6*L*d_w_2(i,4)); %   [N]
%       node 4
        React(i,7) = kb*(-k1/kb*d_w_2(i,3)+k1/kb*d_w_2(i,7)); %   [N]
%       node 5
```

```
        React(i,8) = kb*(-k2/kb*d_w_2(i,5)+k2/kb*d_w_2(i,8)); %    [N]
     end
%------------------------------------------------------------------------
--
%    PLOTs
%------------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = '-';    % Solid line
    %--------------------------------------------------------------------
--
    figure (1)


plot(time,d_w_2(:,3),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,5),'LineStyle',linestyle4,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Displacement                            [m]','FontName',font2,
'FontSize',fontsize_label)
    legend({'w_2', 'w_3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %--------------------------------------------------------------------
--
    figure (2)
```

```
plot(time,S_x(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on

%plot(time,S_x(:,2),'LineStyle',linestyle4,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    %hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Normal             stress             [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Node 2', 'Node 3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (3)


plot(time,React(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Reaction       in       supports       [N]','FontName',font2,
'FontSize',fontsize_label)
    legend({'R_1z'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (4)


plot(time,React(:,7),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,React(:,8),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Reaction       in       supports       [N]','FontName',font2,
'FontSize',fontsize_label)
    legend({'R_4z', 'R_5z'},'Location','northeast')
```

```
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
```

Example 6.5 – Frequency domain response of a system in torsional movement with 2 bar elements and 2 rigid disks by the Direct Method.

```
%========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%------------------------------------------------------------------------
--
%    PURPOSE
%      Compute frequency response of a torsional movement system by FEM:
%      2 bar element and 2 rigid disks - Direct Method
%
%    INPUT:
%          ro    [kg/m3]    mass density
%          R     [m]        radius of the shaft cross section
%          L     [m]        lenght of the element
%          G     [N/m2]     Shear modulus
%          Jd1   [kgm2]     inertia mass of rigid disk 1
%          Jd2   [kgm2]     inertia mass of rigid disk 2
%          f     [Nm]       amplitude of the torque on the shaft
%          fi    [hz]       initial frequency
%          ff    [hz]       final frequency
%          df    [hz]       frequency step
%          ndof  []         number of degree of freedom
%
%    OUTPUT:
%           plot = Amplitude [m] x Frequency [rad/sec]
%------------------------------------------------------------------------
--
%   NOTE:  (1) Example 6.5
%------------------------------------------------------------------------
--
clear
clf
% Data
%------------------------------------------------------------------------
--
% Shaft material properties
ro = 7800;
G = 75.0e9;
% Shaft cross section properties
```

```
R = 0.02;
% Shaft lenghts
L = 0.5;
% Rigid disk properties
Jd1 = 0.08;
Jd2 = 0.0225;
% External torque on rigid disk
f = 100;
% Frequency data
fi = 0;
ff = 2000;
df = 0.1;
% DOF number
ndof = 3;
% Compute the total number of points
nt = (ff-fi)/df;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
% General matrices inicialization
freq = zeros(nt,1);
F = zeros(nt,ndof);
U = zeros(nt,ndof);
% Boundary conditions
u1 = 0;
% mass moment of inertia of the shaft
J = pi*R^4/2;
% Constants of the beam element matrices
mb = ro*J*L/6;
kb = G*J/L;
%-------------------------------------------------------------------------
--
% Assemblage of global matrices after boundary conditions application
M = [1      0       0
     0   4*mb+Jd1   mb
     0     mb   2*mb+Jd2];
C = [0      0      0
     0      0      0
     0      0      0];
K = [1      0      0
     0    2*kb   -kb
     0    -kb     kb];
To = [0    0      f];
 %------------------------------------------------------------------------
---
     for i = 1:nt
     freq(i) = fi+(i-1)*df;
     KM = K-freq(i)^2*M;
     IKM = inv(KM);
     U(i,:) = abs(To*IKM);
     end
%-------------------------------------------------------------------------
--
%   PLOTs
```

```
%-----------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = ':';  % Dash-dot line
    %-------------------------------------------------------------------
--
    figure (1)

semilogy(freq,U(:,2),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

semilogy(freq,U(:,3),'LineStyle',linestyle3,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Frequency                                              domain
response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Frequency                        [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Amplitude [m]','FontName',font2, 'FontSize',fontsize_label)
    legend('node 2', 'node 3')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-------------------------------------------------------------------
--
```

Example 6.7 – Time domain response of a system in torsional movement with 3 bar elements and 5 rigid disks by the Direct Method.

```
%=======================================================================
==
```

```
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update:  17 February 2022
%=========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%-------------------------------------------------------------------------
--
%    PURPOSE
%      Compute time response of a torsional movement system by FEM:
%      3 bar element and 5 rigid disks - Direct Method
%
%    INPUT:
%          ro   [kg/m3]   mass density
%          R    [m]       radius of the shaft cross section
%          L    [m]       lenght of the element
%          G    [N/m2]    Shear modulus
%          Jd1  [kgm2]    inertia mass of rigid disk 1
%          Jd2  [kgm2]    inertia mass of rigid disk 2
%          f    [Nm]      amplitude of the torque on the shaft
%          fe   [rad/sec] excitation frequency
%          ti   [hz]      initial time
%          tf   [hz]      final time
%          dt   [hz]      time step
%          nslx []        node stress location in x direction
%          nslz []        node stress location in z direction
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%-------------------------------------------------------------------------
--
%   NOTE: (1) Example 6.7
%         (2) Harmonic type force
%         (3) Newmark's method
%-------------------------------------------------------------------------
--
clear
clf
% Data
%-------------------------------------------------------------------------
--
% Shaft material properties
ro = 7800;
G = 75.0e9;
% Shaft cross section properties
R1 = 0.025;
R2 = 0.025;
R3 = 0.025;
```

```
% Shaft lenghts
L1 = 0.3;
L2 = 0.3;
L3 = 0.4;
% Rigid disk properties
Jd1 = 0.24811;
Jd2 = 0.11965;
Jd3 = 0.94203;
Jd4 = 0.01938;
Jd5 = 0.08974;
% Coupling ratio
cr = -3;
% External torque on rigid disk
f1 = 0;
f2 = 1.0;
f3 = 0;
f4 = 0;
f5 = 0;
% Frequency of the eletric current
fe = 100*pi;
% Torque data
a = 0.8;
b = -6;
xi = 0.25;
alf = 0.1;
% Frequency response
nf1 = 506.8;    % first natural frequency
% Time data
ti = 0;
tf = 1.0;
npc = 50;       % number of points into a cycle
dt = 1/(nf1/2/pi)/npc;
% DOF number
ndof = 4;
% Compute the total number of points
nt = floor(tf/dt)+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-----------------------------------------------------------------------
--
% Boundary conditions
u1 = 0;
% mass moment of inertia of the shafts
J1 = pi*R1^4/2;
J2 = pi*R2^4/2;
J3 = pi*R3^4/2;
% Constants of the beam element matrices
m1 = ro*J1*L1/6;
m2 = ro*J2*L2/6;
m3 = ro*J3*L3/6;
k1 = G*J1/L1;
k2 = G*J2/L2;
k3 = G*J3/L3;
```

```
%-----------------------------------------------------------------------
--
% Assemblage of global matrices after boundary conditions application
M = [2*m1+Jd1   m1              0                    0
     m1   2*m1+2*m2+Jd2        m2                    0
     0          m2  2*m2+2*cr^2*m3+Jd3+cr^2*Jd4  -cr*m3
     0          0           -cr*m3              2*m3+Jd5];
C = [0          0              0                    0
     0          0              0                    0
     0          0              0                    0
     0          0              0                    0];
K = [k1        -k1             0                    0
     -k1      k1+k2          -k2                    0
     0         -k2       k2+cr^2*k3               cr*k3
     0          0           cr*k3                   k3];
To = [f1       f2         f3-cr*f4                  f5];
%-----------------------------------------------------------------------
--
% Numerical integration - Newmark's method (2)
beta = 1/4;
gama = 1/2;
%-----------------------------------------------------------------------
--
%   Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%-----------------------------------------------------------------------
--
% Initialization of matrices
F = zeros(1,ndof);          %   force
Ft = zeros(nt,ndof);          %   force
edif = zeros(1,ndof);       %   force
a1 = zeros(1,ndof);         %   auxiliar vector a1
a2 = zeros(1,ndof);         %   auxiliar vector a2
displac = zeros(1,ndof);    %
veloc = zeros(1,ndof);      %
accel = zeros(1,ndof);      %
d_w_2 = zeros(nt,ndof);     %   displacement
v_w_2 = zeros(nt,ndof);     %   velocity
a_w_2 = zeros(nt,ndof);     %   aceleration
% Initial conditions
a_w_2(1,:) = To*(a+b*exp(-xi*0)*sin(fe*0+alf))/Jd2;
% Loop for nt total numbers of points
     for i = 2:nt
        t = ti+(i-1)*dt;
        time(i) = t;
        F = To*(a+b*exp(-xi*t)*sin(fe*t+alf));

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;

        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
```

```
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

     end
    %--------------------------------------------------------------------
------
% Calcule normal stress
S_x = zeros(nt,ndof);      %   normal stress

% Loop for nt total numbers of points
     for i = 2:nt
%       element 1-2
        S_x(i,1) = G*R1/L1*(d_w_2(i,2) - d_w_2(i,1)); %   Pa
        S_x(i,1) = S_x(i,1)/1.0e6;       %   MPa
%       element 2-3
        S_x(i,2) = G*R2/L2*(d_w_2(i,3) - d_w_2(i,2)); %   Pa
        S_x(i,2) = S_x(i,2)/1.0e6;       %   MPa
%       element 4-5
        S_x(i,3) = G*R3/L3*(d_w_2(i,4) - d_w_2(i,3)*(-cr)); %   Pa
        S_x(i,3) = S_x(i,3)/1.0e6;       %   MPa
     end
%-------------------------------------------------------------------------
--
%   PLOTs
%-------------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = '-';    % Solid line
```

```
    %-----------------------------------------------------------------
--
    figure (1)


plot(time,d_w_2(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,2),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,3),'LineStyle',linestyle3,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,4),'LineStyle',linestyle4,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Angular         displacement         [rad]','FontName',font2,
'FontSize',fontsize_label)
    legend({'t_1', 't_2', 't_3', 't_5'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------
--
    figure (2)


plot(time,S_x(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Shear            stress            [MPa]/To','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 1-2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------
--
    figure (3)


plot(time,S_x(:,2),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
```

```
    grid on

    title('Time  domain  response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Shear            stress            [MPa]/To','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 2-3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (4)


plot(time,S_x(:,3),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Time  domain  response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Shear            stress            [MPa]/To','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 4-5'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
```

CHAPTER 8

Example 8.2 – Time domain response of a geared steam turbine system in torsional movement subjected to a short-circuit torque by the Direct Method.

```
%========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%------------------------------------------------------------------------
--
%    PURPOSE
%      Compute time response of a torsional movement system by FEM:
%      3 bar element and 5 rigid disks - Direct Method
%
%    INPUT:
%          ro   [kg/m3]   mass density
```

```
%          R    [m]       radius of the shaft cross section
%          L    [m]       lenght of the element
%          G    [N/m2]    Shear modulus
%          Jd1  [kgm2]    inertia mass of rigid disk 1
%          Jd2  [kgm2]    inertia mass of rigid disk 2
%          Jd3  [kgm2]    inertia mass of rigid disk 3
%          Jd4  [kgm2]    inertia mass of rigid disk 4
%          cr   []        coupling relation
%          f    [Nm]      amplitude of the torque on the shaft
%          fe   [rad/sec] excitation frequency
%          ti   [hz]      initial time
%          tf   [hz]      final time
%          dt   [hz]      time step
%          nslx []        node stress location in x direction
%          nslz []        node stress location in z direction
%          ndof []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%-----------------------------------------------------------------------
--
%   NOTE: (1) Example 8.2 (Turbine Blade Life Estimation - Rao, J. S.)
%         (2) Short-circuit torque
%         (3) Newmark's method
%-----------------------------------------------------------------------
--
clear
clf
% Data
%-----------------------------------------------------------------------
--
% Shaft material properties
ro = 7800;
G = 75.0e9;
Su = 600.0;
kt = 2.0;
k1 = 0.75;
% Swinging arm material: limit fatigue stress
SF3 = 0.9*Su;
SE = 0.5*Su*k1;
% Shaft cross section properties
diameter1 = 0.062;
R1 = diameter1/2;
diameter2 = 0.125;
R2 = diameter2/2;
diameter3 = 0.125;
R3 = diameter3/2;
% Shaft lenghts
L1 = 0.2931;
L2 = 0.2876;
L3 = 0.2409;
% Rigid disk properties
```

```
Jd1 = 0.149;
Jd2 = 0.004;
Jd3 = 113.9;
Jd4 = 6.230;
Jd5 = 32.20;
md1 = 00.0;
md3 = 880.5;
md4 = 2.0;
md5 = 225.0;
% Coupling ratio
cr = -14.26;
% External torque on rigid disk
f1 = 0.0;
f2 = 0.0;
f3 = 0.0;
f4 = 0.0;
%f5 = 1.0;
f5 = 6366.6;
% Frequency of the eletric current
fe = 100*pi;
% Torque data
b = 6;
xi = 1/0.03;
% Frequency response
nf1 = 128.7;    % first natural frequency
% Time data
ti = 0;
tf = 1.0;
npc = 50;      % number of points into a cycle
dt = 1/(nf1/2/pi)/npc;
%dt = 0.001;
% DOF number
ndof = 4;
% Compute the total number of points
nt = floor(tf/dt)+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-------------------------------------------------------------------------
--
% Boundary conditions
%u1 = 0;
% mass moment of inertia of the shafts
J1 = pi*R1^4/2;
J2 = pi*R2^4/2;
J3 = pi*R3^4/2;
% Constants of the beam element matrices
m1 = ro*J1*L1/6;
m2 = ro*J2*L2/6;
m3 = ro*J3*L3/6;
k1 = G*J1/L1
k2 = G*J2/L2
k3 = G*J3/L3
%-------------------------------------------------------------------------
--
```

```
% Assemblage of global matrices after boundary conditions application
M = [2*m1+Jd1+md1            m1                    0              0
     m1     2*m1+2*cr^2*m2+Jd2+cr^2*Jd3+cr^2*md3   -cr*m2         0
     0                     -cr*m2          2*m2+2*m3+Jd4+md4      m3
     0                       0                     m3       2*m3+Jd5+md5];

C = [0                       0                     0              0
     0                       0                     0              0
     0                       0                     0              0
     0                       0                     0              0];

K = [k1                     -k1                    0              0
     -k1                   k1+cr^2*k2             cr*k2           0
     0                      cr*k2                 k2+k3          -k3
     0                       0                     -k3            k3];

To = [f1                   f2-cr*f3               f4             f5];
%------------------------------------------------------------------------
--
% Numerical integration - Newmark's method (2)
beta = 1/4;
gama = 1/2;
%------------------------------------------------------------------------
--
%   Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%------------------------------------------------------------------------
--
% Initialization of matrices
F = zeros(1,ndof);         %   force
Tsc = zeros(nt,ndof);     %   torque short-circuit
a1 = zeros(1,ndof);        %   auxiliar vector a1
a2 = zeros(1,ndof);        %   auxiliar vector a2
displac = zeros(1,ndof);   %
veloc = zeros(1,ndof);     %
accel = zeros(1,ndof);     %
d_w_2 = zeros(nt,ndof);    %   displacement
v_w_2 = zeros(nt,ndof);    %   velocity
a_w_2 = zeros(nt,ndof);    %   aceleration
% Initial conditions

% Loop for nt total numbers of points
     for i = 2:nt
        t = ti+(i-1)*dt;
        time(i) = t;
        %F = To*(b*exp(-xi*t)*sin(fe*t));
        F          =          To*(12.353*exp(-33.97*t)*sin(fe*t)+0.5*exp(-
5.45*t)*sin(2*fe*t));
        Tsc(i,:) = F;

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;
```

```
        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

     end
%------------------------------------------------------------------------
--
% Calcule internal torque
Torq = zeros(nt,ndof);     %   torque
% Calcule shear stress
S_x = zeros(nt,ndof);      %   shear stress

% Loop for nt total numbers of points
     for i = 2:nt
%       element 1-2
        S_x(i,1) = G*R1/L1*(d_w_2(i,2) - d_w_2(i,1)); %   Pa
        S_x(i,1) = S_x(i,1)/1.0e6;       %   MPa
        Torq(i,1) = (d_w_2(i,2) - d_w_2(i,1))*k1;     %   Nm
%       element 2-3
        S_x(i,2) = G*R2/L2*(d_w_2(i,3) - d_w_2(i,2)*(-cr)); %   Pa
        S_x(i,2) = S_x(i,2)/1.0e6;       %   MPa
        Torq(i,2) = (d_w_2(i,3) - d_w_2(i,2)*(-cr))*k2;     %   Nm
%       element 3-4
        S_x(i,3) = G*R3/L3*(d_w_2(i,4) - d_w_2(i,3)); %   Pa
        S_x(i,3) = S_x(i,3)/1.0e6;       %   MPa
        Torq(i,3) = (d_w_2(i,4) - d_w_2(i,3))*k3;     %   Nm
     end
%------------------------------------------------------------------------
--
%   PLOTs
%------------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
```

```
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = '-';    % Solid line
    %-----------------------------------------------------------------------
--
    figure (1)


plot(time,d_w_2(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,2),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,3),'LineStyle',linestyle3,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Angular         displacement         [rad]','FontName',font2,
'FontSize',fontsize_label)
    legend({'t_1', 't_2', 't_4'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (2)


plot(time,S_x(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Short-circuit                                              shear
stress','FontName',font2,'FontWeight','bold', 'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    %ylabel('Shear                         stress.To','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Shear             stress              [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 1-2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (3)
```

```
plot(time,S_x(:,2),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Short-circuit                                             shear
stress','FontName',font2,'FontWeight','bold', 'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    %ylabel('Shear                        stress.To','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Shear             stress             [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 2-3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------
--
    figure (4)


plot(time,S_x(:,3),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Short-circuit                                             shear
stress','FontName',font2,'FontWeight','bold', 'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    %ylabel('Shear                        stress.To','FontName',font2,
'FontSize',fontsize_label)
    ylabel('Shear             stress             [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Element 3-4'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------
--
    figure (5)


plot(time,Tsc(:,4),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Short-circuit    torque','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Torque/To','FontName',font2, 'FontSize',fontsize_label)
    %-----------------------------------------------------------------
--
    figure (6)
```

```
plot(time,Torq(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    hold on
    grid on

    title('Internal                       torque                        in
elements','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Torque/To','FontName',font2, 'FontSize',fontsize_label)
    legend({'Element 1-2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (7)


plot(time,Torq(:,2),'LineStyle',linestyle1,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    hold on
    grid on

    title('Internal                       torque                        in
elements','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Torque/To','FontName',font2, 'FontSize',fontsize_label)
    legend({'Element 2-3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (8)


plot(time,Torq(:,3),'LineStyle',linestyle1,'Color',color_curve1,'LineWidt
h',linewidth_curve);
    hold on
    grid on

    title('Internal                       torque                        in
elements','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Torque/To','FontName',font2, 'FontSize',fontsize_label)
    legend({'Element 3-4'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
```

Example 8.4 – Time domain response of a rear suspension model of a motocycle travelling on a roughness road by the Direct Method.

```
%======================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%======================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%----------------------------------------------------------------------
--
%    PURPOSE
%      Compute time response of a transversal movement system by FEM:
%      2 beam element - Direct Method
%
%    INPUT:
%          ro    [kg/m3]   mass density
%          A     [m2]      cross section area
%          L     [m]       lenght of the beam
%          E     [N/m2]    Young's modulus
%          h     [m]       height of the cross section
%          Iy    [m4]      inertia moment of the cross section
%          k     [N/m]     stiffness of the shock absorber
%          c     [N/m/s]   viscous damping of the shock absorber
%          lm    [kg]      lumped mass
%          v_car [m/s]     driving speed of the vehicle
%          road_class      road class
%          k_t   [N/m]     tire stifness
%          fo    [N]       amplitude of the force
%          ti    [s]       initial time
%          tf    [s]       final time
%          dt    [s]       time step
%          nslx  []        node stress location in x direction
%          nslz  []        node stress location in z direction
%          ndof  []        number of degree of freedom
%
%    OUTPUT:
%           plot = Displacement [m] x Time [sec]
%           plot = Velocity [m/s] x Time [sec]
%           plot = Acceleration [m/s] x Time [sec]
%----------------------------------------------------------------------
--
%   NOTE: (1) Example 8.4
%         (2) surface rougness type force
%         (3) Newmark's method
%----------------------------------------------------------------------
--
clear
clf
% Data
%----------------------------------------------------------------------
--
```

```
% Swinging arm material properties
ro = 7800;
E = 200.0e9;
Su = 500.0;
kt = 1.2;
k1 = 0.75;
% Swinging arm material: limit fatigue stress
SF3 = 0.9*Su;
SE = 0.5*Su*k1;
% Swinging arm cross section properties
base = 20.0e-3;
height = 50.0e-3;
thickness = 2.0e-3;
A = base*height-(base-2*thickness)*(height-2*thickness);
Iy = base*height^3/12-(base-2*thickness)*(height-2*thickness)^3/12;
% Swinging arm cross lenghts
L1 = 0.4;
L2 = 0.2;
% Shock absorber properties
k = 1.4e4;
c = 2.3e3;
% Rear wheel mass
lm = 20;
% Motorcycle driving speed
v_car = 15;
% Road class
road_class = 'D';
% Tyre stiffness
k_t = 1.85e4;
% Unit force from the tyre
fo = 1.0;
% Time data
ti = 0;
tf = 10.0;
dt = 0.001;
% Stress calculation data
nslx = L1;
nslz = height/2;
% Total number of degree
ndof = 7;
% Compute the total number of points
nt = tf/dt+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-----------------------------------------------------------------------
--
% Boundary conditions
w1 = 0;
w4 = 0;
% Constants of the beam element matrices
mb1 = ro*A*L1/420;    % constant of the mass matrix element 1
kb1 = E*Iy/L1^3;      % constant of the stiffness matrix element 1
mb2 = ro*A*L2/420;    % constant of the mass matrix element 2
kb2 = E*Iy/L2^3;      % constant of the stiffness matrix element 2
```

```
%-----------------------------------------------------------------------
--
% Assemblage of global matrices after boundary conditions application
M = [1      0                 0                     0                  0
0       0
     0  4*L1^2*mb1         13*L1*mb1             -3*L1^2*mb1            0
0       0
     0  13*L1*mb1      156*mb1+156*mb2    -22*L1*mb1+22*L2*mb2      54*mb2
-13*L2*mb2  0
     0 -3*L1^2*mb1  -22*L1*mb1+22*L2*mb2  4*L1^2*mb1+4*L2^2*mb2  13*L2*mb2
-3*L2^2*mb2  0
     0      0              54*mb2                13*L2*mb2        156*mb2+lm
-22*L2*mb2  0
     0      0           -13*L2*mb2             -3*L2^2*mb2       -22*L2*mb2
4*L2^2*mb2  0
     0      0                 0                     0                  0
0       1];

C = [0      0                 0                     0                  0
0       0
     0      0                 0                     0                  0
0       0
     0      0                 c                     0                  0
0       0
     0      0                 0                     0                  0
0       0
     0      0                 0                     0                  0
0       0
     0      0                 0                     0                  0
0       0
     0      0                 0                     0                  0
0       1];

K = [1      0                 0                     0                  0
0       0
     0   4*L1^2*kb1        -6*L1*kb1             2*L1^2*kb1             0
0       0
     0  -6*L1*kb1     12*kb1+12*kb2+k     -6*L1*kb1+6*L2*kb2       -12*kb2
6*L2*kb2   0
     0  2*L1^2*kb1  -6*L1*kb1+6*L2*kb2  4*L1^2*kb1+4*L2^2*kb2    -6*L2*kb2
2*L2^2*kb2  0
     0      0             -12*kb2               -6*L2*kb2           12*kb2
-6*L2*kb2    0
     0      0           6*L2*kb2              2*L2^2*kb2         -6*L2*kb2
4*L2^2*kb2  0
     0      0                 0                     0                  0
0      1];

Fo = [0     0                 0                     0                 fo
0      0];
%-----------------------------------------------------------------------
--
% Numerical integration - Newmark's method (2)
beta = 1/4;
```

```
gama = 1/2;
%-----------------------------------------------------------------------
--
%   Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%-----------------------------------------------------------------------
--
% Initialization of matrices
F = zeros(1,ndof);          %   force
a1 = zeros(1,ndof);         %   auxiliar vector a1
a2 = zeros(1,ndof);         %   auxiliar vector a2
displac = zeros(1,ndof);    %
veloc = zeros(1,ndof);      %
accel = zeros(1,ndof);      %
d_w_2 = zeros(nt,ndof);     %   displacement
v_w_2 = zeros(nt,ndof);     %   velocity
a_w_2 = zeros(nt,ndof);     %   aceleration
%
%-----------------------------------------------------------------------
--
%   Road Class excitation
%-----------------------------------------------------------------------
--
%   White Noise Function, WN = wgn(nt,1,power)
WN = wgn(nt,1,30);         %   Generate white Gaussian noise samples
%w_r = w_inp;
%   first order filter application
framelen = (nt-1)/100+1;           % odd number
WN = sgolayfilt(WN,3,framelen);    %   white noise function zero mean value
for the road
%-----------------------------------------------------------------------
--
%   road class
%-----------------------------------------------------------------------
--
if road_class == 'A'   %   Very good road class
    Gq = 16;
end
if road_class == 'B'   %   Good road class
    Gq = 64;
end
if road_class == 'C'   %   Average road class
    Gq = 256;
end
if road_class == 'D'   %   Poor road class
    Gq = 1024;
end
if road_class == 'E'   %   Very poor road class
    Gq = 4096;
end
if road_class == 'F'
    Gq = 16384;
end
```

```
%-----------------------------------------------------------------------
--
% Initialization
%-----------------------------------------------------------------------
--
alpha = 2*pi*0.0628;                              %   alpha = 2*pi*f0
delta   =   2*pi*0.1*(v_car*Gq*10^(-6))^(1/2);           %      delta   =
2*pi*n0*(Gq(n0)*v(t))^1/2

%   matrices inicialization of the road_TAYLOR
road_d = zeros(1,nt);    %   inicialization of displacement matrix of the
road
road_v = zeros(1,nt);    %   inicialization of velocity matrix of the road
- right
A1_road = zeros(1,nt);        %   3/dt*p(t)+2*v(t)+dt/2*a(t)
%-----------------------------------------------------------------------
--

% Loop for nt total numbers of points
     for i = 2:nt

%   Road Class input
    A1_road(i)=3/dt*road_d(i-1)+2*road_v(i-1);
    road_d(i)=(delta*WN(i)+A1_road(i))/(3/dt+alpha);
    road_v(i)=3/dt*road_d(i)-A1_road(i);

   %   Road Class input - end
        t = ti+(i-1)*dt;
        time(i) = t;
        F = Fo*k_t*road_d(i);

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;

        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

    end
%-----------------------------------------------------------------------
--
% Calcule normal stress
S_x = zeros(nt,ndof);      %   normal stress

df1 = (-6/L1^2 + 12/L1^3*nslx);
df2 = (-4/L1 + 6/L1^2*nslx);
```

```
df3 = (6/L1^2 - 12/L1^3*nslx);
df4 = (-2/L1 + 6/L1^2*nslx);

% Loop for nt total numbers of points
     for i = 2:nt
%       node 2
        S_x(i,1)    =    E*nslz*(df1*d_w_2(i,1)    +    df2*d_w_2(i,2)    +
df3*d_w_2(i,3) + df4*d_w_2(i,4)); %   Pa
        S_x(i,1) = S_x(i,1)/1.0e6;       %   MPa
     end
%-------------------------------------------------------------------------
--
% Calcule reaction on supports
React = zeros(nt,ndof);      %   reaction in supports

% Loop for nt total numbers of points
     for i = 2:nt
%       node 1
        React(i,1)   =   mb1*(54*a_w_2(i,3)-13*L1*a_w_2(i,4))   +   kb1*(-
12*d_w_2(i,3)-6*L1*d_w_2(i,4)); %   [N]
%       node 4
        React(i,7) = kb1*(-k/kb1*d_w_2(i,3)+k/kb1*d_w_2(i,7)); %   [N]
     end

%-------------------------------------------------------------------------
--
%   Compute the mean Me_r and standard deviation Sd_r of the road roughness
%   Mean of the displacement of the road
Me_r = mean(road_d);
%   Standard deviation of the displacement of the road (Sd1 = 68%)
Sd_r = std(road_d);
%   Variance of the displacement of the road
Va_r = var(road_d);
%   Mean square of the displacement of the road
Rms_r = rms(road_d);
fprintf ('\n1. Road roughness\n')
fprintf ('Road roughness mean = %.3f m\n', Me_r)
fprintf ('Road roughness standard deviation = %.3f m\n', Sd_r)
fprintf ('Road roughness variance = %.3f m\n', Va_r)
fprintf ('Road roughness Mean square = %.3f m\n', Rms_r)
%-------------------------------------------------------------------------
--
%   Compute the mean Me_S and standard deviation Sd_S of the normal stress
%   Nominal mean stress
Me_S_2 = mean(S_x(:,1));    % mean stress at node 2
%   Nominal standard deviation of the normal stress (Sd = 68%)
Sd_S_2 = std(S_x(:,1));      % standard deviation of the normal stress at
node 2
%   Nominal stress
fprintf ('\n2. Mean stress\n')
fprintf ('Mean stress at node 2 = %.2f MPa\n', Me_S_2)
fprintf ('\n3. standard deviation\n')
fprintf ('standard deviation of the normal stress at node 2 = %.2f MPa\n',
Sd_S_2)
```

```
%   Stress at the notch
fprintf ('\n4. Mean stress at the notch\n')
fprintf ('Mean stress at node 2 = %.2f MPa\n', Me_S_2*kt)
fprintf ('\n5. standard deviation at the notch\n')
fprintf ('standard deviation of the normal stress at node 2 = %.2f MPa\n',
Sd_S_2*kt)

if Me_S_2 <= 0
   Me_S_2 = 0.0;
end

%   Sa x N curve
fprintf ('\n6. Sa x N\n')
m_curve = -1/3*log10(SF3/SE);
fprintf ('m_curve = %.3f \n', m_curve)
C_curve = SF3*1000^(-m_curve);
fprintf ('C_curve = %.1f \n', C_curve)
fprintf ('\n7. Number of cycles (Life)\n')
N_Sdt_1 = ((Su/(Su-Me_S_2*kt)*Sd_S_2*kt*1)/C_curve)^(1/m_curve);
fprintf ('N_Sdt_1 = %.2e cycles\n', N_Sdt_1);
N_Sdt_2 = ((Su/(Su-Me_S_2*kt)*Sd_S_2*kt*2)/C_curve)^(1/m_curve);
fprintf ('N_Sdt_2 = %.2e cycles\n', N_Sdt_2);
N_Sdt_3 = ((Su/(Su-Me_S_2*kt)*Sd_S_2*kt*3)/C_curve)^(1/m_curve);
fprintf ('N_Sdt_3 = %.2e cycles\n', N_Sdt_3);
%-----------------------------------------------------------------------
--
%   PLOTs
%-----------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
    color_curve3 = [ 0.0 1.0 0.0];    % green
    color_curve4 = [ 0.0 0.0 1.0];    % blue
    color_curve5 = [ 1.0 0.0 1.0];    % magenta
    linestyle1 = '-';    % Solid line
    linestyle2 = '--';   % Dashed line
    linestyle3 = ':';    % Dotted line
    linestyle4 = '-.';   % Dash-dot line
    linestyle5 = '-';    % Solid line
%-----------------------------------------------------------------------
--
    figure (1)
```

```
plot(time,d_w_2(:,3),'LineStyle',linestyle2,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,5),'LineStyle',linestyle4,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Displacement                              [m]','FontName',font2,
'FontSize',fontsize_label)
    legend({'w_2', 'w_3'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %------------------------------------------------------------------
--
    figure (2)


plot(time,S_x(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Normal             stress             [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Node 2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %------------------------------------------------------------------
--
    figure (3)


plot(time,React(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Reaction       in       supports       [N]','FontName',font2,
'FontSize',fontsize_label)
    legend({'R_1z'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %------------------------------------------------------------------
--
    figure (4)
```

```
plot(time,React(:,7),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Reaction       in       supports       [N]','FontName',font2,
'FontSize',fontsize_label)
    legend({'R_4z'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %---------------------------------------------------------------------
--
    figure (5)


plot(time,road_d,'LineStyle',linestyle1,'Color',color_curve1,'LineWidth',
linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Road             roughness             [m]','FontName',font2,
'FontSize',fontsize_label)
    legend({'w_r'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %---------------------------------------------------------------------
--
```

Example 8.5 – Time domain response of a rotary bending system supported on rigid bearings by the Direct Method.

```
%=========================================================================
==
%   Author: José Carlos de Carvalho Pereira
%   Email: j.c.carvalho.p@ufsc.br
%   Last Update: 17 February 2022
%=========================================================================
==
% Copyright (c)  Analysis and Mechanical Design Group - GRANTE
%                Department of Mechanical Engineering - EMC
%                Federal University of Santa Catarina - UFSC
%                Florianópolis - Brazil
%-------------------------------------------------------------------------
--
%    PURPOSE
%      Compute time stress of a transversal movement system by FEM:
%      3 beam element - Direct Method
```

```
%
%    INPUT:
%           ro   [kg/m3]   mass density
%           E    [N/m2]    Young's modulus
%           L1   [m]       lenght of the first element
%           L2   [m]       lenght of the second element
%           ds   [m]       shaft diameter
%           mrd  [kg]      mass of the rigid disk
%           Ird  [kgm2]    moment of inergia of the rigid disk
%           Jrd  [kgm2]    polar moment of inergia of the rigid disk
%           md   [kg]      unbalanced mass
%           d    [m]       excentricity
%           ky1  [N/m]     stiffness on y direction at bearing 1
%           kz1  [N/m]     stiffness on z direction at bearing 1
%           ky2  [N/m]     stiffness on y direction at bearing 2
%           kz2  [N/m]     stiffness on z direction at bearing 2
%           cy1  [N/m/s]   viscous damping on y direction at bearing 1
%           cz1  [N/m/s]   viscous damping on z direction at bearing 1
%           cy2  [N/m/s]   viscous damping on y direction at bearing 2
%           cz2  [N/m/s]   viscous damping on z direction at bearing 2
%           Omo  [rad/sec] nominal rotation
%           fa   [rad/sec2]excitation rotation acceleration
%           ti   [s]       initial time
%           tf   [s]       final time
%           dt   [s]       time step
%           nslx []        node stress location in x direction
%           nslz []        node stress location in z direction
%           ndof []        number of degree of freedom
%
%    OUTPUT:
%            plot = Displacement [m] x Time [sec]
%            plot = Velocity [m/s] x Time [sec]
%            plot = Acceleration [m/s] x Time [sec]
%------------------------------------------------------------------------
--
%   NOTE: (1) Example 8.5
%         (2) Hydraulic pump supported by rigid bearings
%         (3) Syncronous unbalanced type force
%         (4) Newmark's method
%------------------------------------------------------------------------
--
clear
clf
% Data
%------------------------------------------------------------------------
--
% Shaft material properties
ro = 7800;
E = 200.0e9;
Su = 500.0;
kt = 1.2;
k1 = 0.75;
% Swinging arm material: limit fatigue stress
SF3 = 0.9*Su;
```

```
SE = 0.5*Su*k1;
% Shaft cross section properties
diameter = 9.525e-3;
% Shaft lenghts
L1 = 0.0762;
L2 = 0.0889;
L3 = 0.0889;
% Rigid disk properties
mrd = 4.53592;
Ird = 0.014631971;
Jrd = 0.023411153;
% Unbalanced mass properties
md = 0.0001;
d = 0.15;
% Balancing mass properties
mb = 0.0001;
b = 0.10;
bet = 0;             % balancing phase - degree
bet = bet*pi/180;    % balancing phase - radians
% Bearing properties
ky1 = 1.75e12;
kz1 = 1.75e12;
ky2 = 1.75e12;
kz2 = 1.75e12;
cy1 = 00;
cz1 = 00;
cy2 = 00;
cz2 = 00;
% Start-up conditions
Omo = 290.0;    % nominal angular velocity
fa = 150.0;     % inclination of the angular velocity
% Frequency response
nf1 = 243.7;    % first natural frequency
% Time data
ti = 0;
tf = 50.0;
npc = 50;       % number of points into a cycle
dt = 1/(nf1/2/pi)/npc;
% Location point of the normal stress calculation
nslx = L1;
nslz = diameter/2;
% DOF number
ndof = 20;
% Compute the total number of points
nt = floor(tf/dt)+1;
% General matrices inicialization
time = zeros(nt,1);    %   time vector
%-----------------------------------------------------------------------
--
% Boundary conditions
v5 = 0;
w6 = 0;
v7 = 0;
w8 = 0;
```

```
% cross section of the shaft properties
A = pi*diameter^2/4;
I = pi*diameter^4/64;
% Constants of the beam element matrices
mb1 = ro*A*L1/420;  % constant of the mass matrix element 1
mb2 = ro*A*L2/420;  % constant of the mass matrix element 2
mb3 = ro*A*L3/420;  % constant of the mass matrix element 3
kb1 = E*I/L1^3;      % constant of the stiffness matrix element 1
kb2 = E*I/L2^3;      % constant of the stiffness matrix element 2
kb3 = E*I/L3^3;      % constant of the stiffness matrix element 3
%------------------------------------------------------------------------
--
% Assemblage of global matrices after boundary conditions application
%    v1            psy1      w1        fi1          v2                  psy2
w2           fi2                   v3                 psy3              w3
f3           v4           psy4     w4      f4        v5   w6  v7  w8
M = [156*mb1   -22*L1*mb1    0        0        54*mb1             13*L1*mb1
0             0                    0                  0                 0
0            0            0        0        0         0    0   0   0
    -22*L1*mb1   4*L1^2*mb1    0          0        -13*L1*mb1             -
3*L1^2*mb1           0              0                   0                 0
0                 0             0            0        0        0         0    0
0   0
       0             0       156*mb1  22*L1*mb1       0                     0
54*mb1      -13*L1*mb1                0                 0                 0
0            0            0        0        0         0    0   0   0
       0             0      22*L1*mb1  4*L1^2*mb1     0                     0
13*L1*mb1     -3*L1^2*mb1              0                0                 0
0            0            0        0        0         0    0   0   0
    54*mb1     -13*L1*mb1      0          0  156*mb1+156*mb2+mrd  22*L1*mb1-
22*L2*mb2        0              0                  54*mb2         13*L2*mb2
0                 0             0            0        0        0         0    0
0   0
   13*L1*mb1    -3*L1^2*mb1        0               0    22*L1*mb1-22*L2*mb2
4*L2^2*mb1+4*L2^2*mb2+Ird  0              0               -13*L2*mb2      -
3*L2^2*mb2                  0                  0              0             0
0       0         0    0    0   0
       0            0         54*mb1   13*L1*mb1       0                    0
156*mb1+156*mb2+mrd   -22*L1*mb1+22*L2*mb2       0                        0
54*mb2          -13*L2*mb2         0            0        0        0         0
0   0   0
       0            0     -13*L1*mb1  -3*L1^2*mb1     0                     0
-22*L1*mb1+22*L2*mb2  4*L1^2*mb1+4*L2^2*mb2+Ird  0                        0
13*L2*mb2         -3*L2^2*mb2       0            0        0        0         0
0   0   0
       0            0           0         0           0                     0
0             0              156*mb2+156*mb3 22*L2*mb2-22*L3*mb3         0
0          54*mb3    13*L3*mb3   0        0         0    0   0   0
       0           0           0         0      13*L2*mb2         -3*L2^2*mb2
0             0         22*L2*mb2-22*L3*mb3 4*L2^2*mb2+4*L2^3*mb3        0
0       -13*L3*mb3  -3*L3^2*mb3  0        0         0    0   0   0
       0           0           0         0           0                      0
54*mb2      13*L2*mb2             0              0         156*mb2+156*mb3
-22*L2*mb2+22*L3*mb3  0            0      54*mb3 -13*L3*mb3 0    0   0   0
```

```
       0          0          0        0          0                  0
-13*L2*mb2     -3*L2^2*mb2                 0                   0      -
22*L2*mb2+22*L3*mb3  4*L2^2*mb2+4*L3^2*mb3  0            0   13*L3*mb3  -
3*L2^2*mb2 0    0   0   0
       0          0          0        0          0                  0
0            0                 54*mb3           13*L3*mb3            0
0        156*mb3    22*L3*mb3   0       0       0    0    0    0
       0          0          0        0          0                  0
0            0              13*L3*mb3       -3*L3^2*mb3              0
0       22*L3*mb3 4*L3^2*mb3    0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                   0                0              54*mb3
13*L2*mb2       0          0    156*mb3 -22*L3*mb3  0    0    0    0
       0          0          0        0          0                  0
0             0                    0                  0                 -
13*L3*mb3         -3*L2^2*mb2       0           0 -22*L3*mb3 4*L3^2*mb3  0
0   0   0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       1    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    1    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    1    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    1];

C = [ cy1         0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0         cz1       0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
       0          0          0        0          0                  0
0            0                    0                 0               0
0           0           0       0       0       0    0    0    0
```

```
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0          cy2          0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0      cz2      0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        1    0    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    1    0    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    1    0
       0           0           0         0           0                     0
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    1];

%    v1            psy1      w1        fi1         v2                   psy2
w2            fi2                    v3                 psy3             w3
f3            v4            psy4      w4      f4        v5   w6   v7   w8
K = [12*kb1+ky1 -6*L1*kb1    0         0        -12*kb1             -6*L1*kb1
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
    -6*L1*kb1 4*L1^2*kb1     0         0       6*L1*kb1            2*L1^2*kb1
0             0                    0                  0                    0
0           0           0       0       0        0    0    0    0
      0              0    12*kb1+kz1   6*L1*kb1       0                    0
-12*kb1         6*L1*kb1             0                  0                  0
0           0           0       0       0        0    0    0    0
      0              0      6*L1*kb1  4*L1^2*kb1      0                    0
-6*L1*kb1       2*L1^2*kb1            0                  0                  0
0           0           0       0       0        0    0    0    0
  -12*kb1       6*L1*kb1       0         0     12*kb1+12*kb2       6*L1*kb1-
6*L2*kb2          0              0                -12*kb2          -6*L2*kb2
```

```
0                  0            0            0        0        0         0    0
0    0
 -6*L1*kb1     2*L1^2*kb1           0               0      6*L1*kb1-6*L2*kb2
4*L1^2*kb1+4*L2^2*kb2        0                 0                   6*L2*kb2
2*L2^2*kb2             0                 0            0            0        0
0        0     0    0    0
      0             0       -12*kb1   -6*L1*kb1       0                    0
12*kb1+12*kb2 -6*L1*kb1+6*L2*kb2          0                0             -
12*kb2            6*L2*kb2         0            0        0        0         0
0    0    0
      0             0       6*L1*kb1 2*L1^2*kb1      0                     0
-6*L1*kb1+6*L2*kb2 4*L1^2*kb1+4*L2^2*kb2          0                        0
-6*L2*kb2          2*L2^2*kb2       0            0        0        0        0
0    0    0
      0            0           0         0        -12*kb2            6*L2*kb2
0              0               12*kb2+12*kb3   6*L2*kb2-6*L3*kb3          0
0         -12*kb3    -6*L3*kb3   0        0        0    0    0    0
      0            0          0         0       -6*L2*kb2         2*L2^2*kb2
0              0              6*L2*kb2-6*L3*kb3 4*L2^2*kb2+4*L3^2*kb3      0
0         6*L3*kb3  2*L3^2*kb3   0        0        0    0    0    0
      0             0           0         0           0                    0
-12*kb2      -6*L2*kb2              0              0          12*kb2+12*kb3
-6*L2*kb2+6*L3*kb3   0           0     -12*kb3 6*L3*kb3    0     0    0    0
      0             0           0         0           0                    0
6*L2*kb2       2*L2^2*kb2                0                   0             -
6*L2*kb2+6*L3*kb3 4*L2^2*kb2+4*L3^2*kb3 0          0  -6*L3*kb3 2*L3^2*kb3
0    0    0    0
      0             0           0         0           0                    0
0              0                -12*kb3           6*L3*kb3                 0
0       12*kb3+ky2   -6*L3*kb3    0        0        0    0    0    0
      0             0           0         0           0                    0
0              0               -6*L3*kb3        2*L3^2*kb3                 0
0       -6*L3*kb3    4*L3^2*kb3   0        0        0    0    0    0
      0             0           0         0           0                    0
0             0                 0                0                  -12*kb3
-6*L3*kb3          0           0 12*kb3+kz2 6*L3*kb3    0     0    0    0
      0             0           0         0           0                    0
0             0                 0                0                  6*L3*kb3
2*L3^2*kb3         0           0  6*L3*kb3 4*L3^2*kb3   0     0    0    0
      0             0           0         0           0                    0
0              0                 0                0                        0
0            0           0       0        0       1    0    0    0
      0             0           0         0           0                    0
0              0                 0                0                        0
0            0           0       0        0       0    1    0    0
      0             0           0         0           0                    0
0              0                 0                0                        0
0            0           0       0        0       0    0    1    0
      0             0           0         0           0                    0
0              0                 0                0                        0
0            0           0       0        0       0    0    0    1];
```

```
Fo = [0             0           0         0           1                    0
1               0                    0                 0                   0
0            0           0       0        0       0    0   0   0];
%-------------------------------------------------------------------------
--
% Numerical integration - Newmark's method (2)
beta = 1/4;
gama = 1/2;
%-------------------------------------------------------------------------
--
% Governing equation, d_w = mck\(ff+M*a2+C*a1)
MCK = M/beta/dt^2+gama*C/beta/dt+K;
%-------------------------------------------------------------------------
--
% Initialization of matrices
F = zeros(1,ndof);          %   force
Ft = zeros(nt,ndof);          %   force
edif = zeros(1,ndof);       %   force
a1 = zeros(1,ndof);         %   auxiliar vector a1
a2 = zeros(1,ndof);         %   auxiliar vector a2
displac = zeros(1,ndof);    %
veloc = zeros(1,ndof);      %
accel = zeros(1,ndof);      %
d_w_2 = zeros(nt,ndof);     %   displacement
v_w_2 = zeros(nt,ndof);     %   velocity
a_w_2 = zeros(nt,ndof);     %   aceleration
r_c = zeros(nt,1);          %   amplitude of the center

% Calcule normal stress
S_x = zeros(nt,ndof);       %   normal stress

dg1 = (-6/L1^2 + 12/L1^3*nslx);
dg2 = (-4/L1 + 6/L1^2*nslx);
dg3 = (6/L1^2 - 12/L1^3*nslx);
dg4 = (-2/L1 + 6/L1^2*nslx);

df1 = (-6/L1^2 + 12/L1^3*nslx);
df2 = (4/L1 - 6/L1^2*nslx);
df3 = (6/L1^2 - 12/L1^3*nslx);
df4 = (2/L1 - 6/L1^2*nslx);

% Loop for nt total numbers of points
     for i = 2:nt
        t = ti+(i-1)*dt;
        time(i) = t;
        Om(i) = Omo*(1-exp(-fa*t/Omo));
        dOmdt(i) = fa*(-exp(-fa*t/Omo));
% Unbalanced mass excitation
        Fo(5)         =          -md*d*2*dOmdt(i)*cos(Om(i)*t)          +
md*d*(dOmdt(i)*t+Om(i))*Om(i)*sin(Om(i)*t);
        Fo(7)        =              md*d*2*dOmdt(i)*sin(Om(i)*t)        +
md*d*(dOmdt(i)*t+Om(i))*Om(i)*cos(Om(i)*t);

% Balancing mass - one plan compensation
```

```
        Fo(5)    =    Fo(5)    -    mb*d*2*dOmdt(i)*cos(Om(i)*t+bet)    +
mb*b*(dOmdt(i)*t+Om(i))*Om(i)*sin(Om(i)*t+bet);
        Fo(7)    =    Fo(7)    +    mb*d*2*dOmdt(i)*sin(Om(i)*t+bet)    +
mb*b*(dOmdt(i)*t+Om(i))*Om(i)*cos(Om(i)*t+bet);

        F = Fo;

        displac = (F+a2*M+a1*C)/MCK;
        veloc = gama/(beta*dt)*displac-a1;
        accel = 1/(beta*dt^2)*displac-a2;

        a1    =    gama/(beta*dt)*displac    +    (gama/beta-1)*veloc    +
(gama/(2*beta)-1)*dt*accel;
        a2  =  1/(beta*dt^2)*displac  +  1/(beta*dt)*veloc  +  (1/(2*beta)-
1)*accel;

        for j = 1:ndof
        d_w_2(i,j) = displac(j);
        v_w_2(i,j) = veloc(j);
        a_w_2(i,j) = accel(j);
        end

%       amplitude of the center - node 2
        r_c(i) = sqrt(d_w_2(i,5)^2 + d_w_2(i,7)^2);

%       node 2
        S_x_v = -E*nslz*(dg1*d_w_2(i,1) + dg2*d_w_2(i,2) + dg3*d_w_2(i,5)
+ dg4*d_w_2(i,6))*sin(Om(i)*t);
        S_x_w = -E*nslz*(df1*d_w_2(i,3) + df2*d_w_2(i,4) + df3*d_w_2(i,7)
+ df4*d_w_2(i,8))*cos(Om(i)*t);

        S_x(i,1) = S_x_v + S_x_w;         %   Pa

        S_x(i,1) = S_x(i,1)/1.0e6;        %   MPa

     end
%-------------------------------------------------------------------------
--
%   PLOTs
%-------------------------------------------------------------------------
--
    %   Size plot config
    set(0,'defaultfigureposition',[500 200 600 400])

    %   Font config
    font1 = 'Times New Roman';
    font2 = 'Arial';
    fontsize_title = 13;
    fontsize_label = 12;
    fontsize_axis = 11;
    linewidth_curve = 2;
    fontsize_leg = 10;
    color_curve1 = [ 0.0 0.0 0.0];    % black
    color_curve2 = [ 1.0 0.0 0.0];    % red
```

```
    color_curve3 = [ 0.0 1.0 0.0];   % green
    color_curve4 = [ 0.0 0.0 1.0];   % blue
    color_curve5 = [ 1.0 0.0 1.0];   % magenta
    linestyle1 = '-';   % Solid line
    linestyle2 = '--';  % Dashed line
    linestyle3 = ':';   % Dotted line
    linestyle4 = '-.';  % Dash-dot line
    linestyle5 = '-';   % Solid line
    %-----------------------------------------------------------------------
--
    figure (1)


plot(time,d_w_2(:,5),'LineStyle',linestyle1,'Color',color_curve1,'LineWid
th',linewidth_curve);
    hold on

plot(time,d_w_2(:,7),'LineStyle',linestyle2,'Color',color_curve2,'LineWid
th',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Displacement                           [m]','FontName',font2,
'FontSize',fontsize_label)
    legend({'v_2', 'w_2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (2)


plot(time,S_x(:,1),'LineStyle',linestyle1,'Color',color_curve1,'LineWidth
',linewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Normal             stress             [MPa]','FontName',font2,
'FontSize',fontsize_label)
    legend({'Node 2'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %-----------------------------------------------------------------------
--
    figure (3)


plot(time,Om,'LineStyle',linestyle1,'Color',color_curve1,'LineWidth',line
width_curve);
    grid on
```

```
    title('Excitation     frequency','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Frequency                         [rad/sec]','FontName',font2,
'FontSize',fontsize_label)
    %------------------------------------------------------------------
--
    figure (4)


plot(time,r_c,'LineStyle',linestyle1,'Color',color_curve1,'LineWidth',lin
ewidth_curve);
    hold on
    grid on

    title('Time   domain   response','FontName',font2,'FontWeight','bold',
'FontSize',fontsize_title)
    xlabel('Time [sec]','FontName',font2, 'FontSize',fontsize_label)
    ylabel('Amplitude     of     the     center     [m]','FontName',font2,
'FontSize',fontsize_label)
    legend({'r'},'Location','northeast')
    set(legend,'FontName',font2, 'FontSize',fontsize_leg);
    %------------------------------------------------------------------
--
```

REFERENCE

[1] The MathWorks Inc., MATLAB version: 8.5.0 (R2015a), Natick, Ma.: The MathWorks Inc., 2015.

SUBJECT INDEX

A

B

C

José Carlos de Carvalho Pereira

D

E

L

M

N

O

P

R

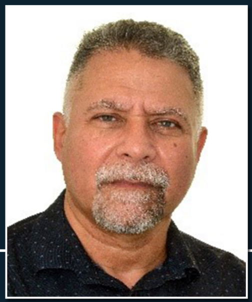
U

V

W

This work addresses the problem of machine dynamics in a clear and objective manner, using Lagrangian formulations, with special emphasis on transient regimes during start-up and shutdown. The topics covered are aimed at those seeking solutions to solve new and complex engineering problems.

Lauro Cesar Nicolazzi, Dr. Eng.
Professor of Mechanical
Engineering Course of
Universidade Federal de Santa Catarina
Florianópolis – SC – Brazil



José Carlos de Carvalho Pereira

The author holds a degree in Mechanical Engineering from the Federal University of Uberlândia, Brazil (1987); a master's degree in Mechanical Engineering from the Federal University of Uberlândia (1990); a master's degree in Mechanical Engineering from the Institut National des Sciences Appliquées - Lyon, France (1992); and a PhD in Mechanical Engineering from the Institut National des Sciences Appliquées - Lyon, France (1996). He worked as a researcher at the Aeronautics and Space Institute of the Department of Aerospace Science and Technology from 1996 to 1998. He has been a professor in the Department of Mechanical Engineering at the Federal University of Santa Catarina since 1998 and a full professor since 2017. He has experience in Mechanical Engineering, with an emphasis on the Dynamics of Mechanical Systems, Analysis of Structures in Composite Materials, and Finite Elements.