NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Dynamic Analysis of Rigid-Link Open-Chain Robot Manipulators Using Cartesian Tensor Methods

Constantinos A. Balafoutis

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

July, 1989

ISBN 0-315-51344-6

ABSTRACT

Dynamic Analysis of Rigid-Link Open-Chain Robot Manipulators Using Cartesian Tensor Methods

Constantinos A. Balafoutis Ph. D.,
Concordia University, 1989.

This thesis is concerned with developing computationally efficient algorithms for solving some basic problems of robot dynamics. In particular, the following problems of rigid-link open-chain robot manipulator dynamics are considered : *i*) the problem of computing inverse dynamics, *ii*) the problem of computing forward dynamics, and *iii*) linearization of the equations of motion for the above mentioned class of robot manipulators. Computationally efficient solutions of these problems are prerequisites for real-time robot applications, and are therefore necessary for flexible automation in a dynamically changing environment. The algorithms presented in this thesis can be used to solve in a computationally efficient manner these fundamental problems of robotic manipulators. These algorithms rely heavily on Cartesian tensor analysis. In this thesis, it is shown that by exploiting the relationships between second order Cartesian tensors and their vector invariants, a number of new tensor identities can be obtained. These identities enrich the theory of Cartesian tensors and allow us to manipulate complex Cartesian tensor equations effectively. Moreover, a geometric characterization for second order skew-symmetric Cartesian tensors is provided which gives a physical interpretation and a deeper insight into tensor algebraic equations in general and rotation tensors in particular. Thus, based on a new understanding of Cartesian tensor analysis, a conceptually simple easy to implement and computationally efficient tensor methodology is proposed in this thesis for studying classical rigid body dynamics.

Application of this tensor methodology to the dynamic analysis of rigid-link open-chain robot manipulators is simple and leads us to an efficient formulation of the

dynamic equations of motion. Moreover, the use of generalized and augmented links enables us to devise modeling schemes which are very much suited for the dynamic analysis of the aforementioned class of robot manipulators, since they allow us to compute as many as possible manipulator's configuration independent dynamic parameters off-line. By using Cartesian tensor analysis and the ideas of generalized and augmented links, we propose algorithms for solving the problem of inverse dynamics, the problem of forward dynamics, and the linearization of the equations of motion of rigid-link open-chain robot manipulators which are computationally the most efficient non-customized algorithms presently available.

To My Parents

Anastasio And Georgia

Στους Γονεις Μου

Αναστασιο Και Γεωργια

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor R. V. Patel, for his continued guidance, suggestions and encouragement during the course of this research. Professor R. V. Patel besides being a great teacher is also an excellent man. Working in association with him has been a pleasant experience for me and I thank him for that.

I consider myself fortunate to have made the acquaintance of so many wonderful people over the last few years. It is with great pleasure that I acknowledge the support and motivation provided by my colleagues Pradeep, Murthy, Zheng, Mohamed, Nat, Shahrokh and Babu.

Last but not the least, it is with pride that I acknowledge the support and inspiration of my parents Anastasio and Georgia, my brothers Nick and Gianni and my sisters Andrianna and Gianna — back home. Also, I would like to express my heartfelt thanks to all my relatives and friends here in Montreal for their encouragement and the good times I had with them.

This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. A1345 awarded to Prof. R. V. Patel.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

n	Number of degrees-of-freedom of a manipulator
m_i	Mass of the i-th link
$\overline{m}_i$	Composite mass of links i to n
o_i	The point which denotes the origin of the i-th link coordinate system
c_i	The point which denotes the center of mass (c.m.) of the i-th link
$\mathbf{1}$	The unity (identity) tensor
$\mathbf{I}_{C_i}^k$	The inertia tensor of the i-th link about c_i expressed in the k-th coordinate system orientation
$\mathbf{J}_{C_i}^k$	The pseudo-inertia tensor of the i-th link about c_i expressed in the k-th coordinate system orientation
$\mathbf{K}_{O_i}^k$	The inertia tensor of the i-th augmented link about o_i expressed in the k-th coordinate system orientation
$\hat{\mathbf{K}}_{O_i}^k$	The pseudo-inertia tensor of the i-th augmented link about o_i expressed in the k-th coordinate system orientation
$\mathbf{E}_{O_i}^k$	The inertia tensor of the i-th generalized link about o_i expressed in the k-th coordinate system orientation
$\mathbf{D}$	The joint space generalized inertia tensor of a robot manipulator
$\mathbf{u}_{O_i}^i$	The first moment of the i-th augmented link about o_i, expressed in the i-th coordinate system orientation
$\mathbf{U}_{O_i}^i$	The first moment of the i-th generalized link about o_i, expressed in the i-th coordinate system orientation
$\mathbf{F}_{C_i}^i$	Force vector acting on c_i expressed in the i-th coordinate system orientation

$\mathbf{M}_{c_i}^{i}$ Moment vector about c_i expressed in the i-th coordinate system orientation

$\mathbf{r}_{i,j}^{i}$ Position vector from o_i to c_j expressed in the i-th coordinate system orientation

$\mathbf{s}_{i,j}^{i}$ Position vector from o_i to o_j expressed in the i-th coordinate system orientation

$\dot{\mathbf{r}}$ $(\ddot{\mathbf{r}})$ The absolute velocity (acceleration) of vector $\mathbf{r}$

ω_i^i $(\dot{\omega}_i^i)$ Absolute angular velocity (acceleration) of the i-th coordinate system expressed in the i-th coordinate system orientation

Ω_i^i The angular acceleration tensor of the i-th link, expressed in the i-th coordinate system orientation

$\mathbf{q}$ $(\dot{\mathbf{q}}, \ddot{\mathbf{q}})$ Joint space position (velocity, acceleration) vector

χ $(\dot{\chi}, \ddot{\chi})$ Cartesian space position (velocity, acceleration) vector

τ Joint space generalized force vector

A_i The 3×3 coordinate (or the 4×4 homogeneous) transformation matrix from the i-th frame to the $(i-1)$-th frame

W_i The 3×3 coordinate (or the 4×4 homogeneous) transformation matrix from the i-th frame to the base frame

$dual(\cdot)$ A tensor-valued vector operator (or a vector-valued tensor operator)

$\tilde{\mathbf{v}}$ Skew-symmetric tensor which denotes the action of the *dual* operator on a vector $\mathbf{v}$

$\mathbf{D}^o$ The inertia force-acceleration sensitivity tensor of a linearized robot model

$\mathbf{V}^o$ The centrifugal and Coriolis force-velocity sensitivity tensor of a linearized robot model

$\mathbf{P}^o$ The force-position sensitivity tensor of a linearized robot model

□	Denotes the end of a proof
IDP	Inverse Dynamics Problem
FDP	Forward Dynamics Problem

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 ROBOT MANIPULATORS : AN OVERVIEW

The science of robotics began less than thirty years ago, when the first computer-controlled manipulator was demonstrated by Unimation Inc. Since that time, scientists and engineers have designed hundreds of different manipulators and the study of robotics has become a highly complex and interdisciplinary field which encompasses a number of topics taken from other "classical" fields such as : mathematics, mechanical and electrical engineering, computer science, etc. Today, with the advances made over the last decade, robots have come to symbolize high-level automation in almost every aspect of human activities. Applications of robots can be found almost everywhere : from hazardous environments such as in space and oceans, to more pleasant home environment [1-5]. However, by far the majority of applications of robots to date has been in the automotive manufacturing and metalworking industries [6,7]. A few typical applications of the so called *industrial robots* include : *spray painting, welding, material handling, machine loading, assembly,* etc.

Exactly what constitutes an industrial robot is still debatable not only from the view point of social science experts, but even from that of robotics specialists. For example, the Robot Institute of America defines [7] an industrial robot as :

> *"A reprogrammable multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks".*

On the other hand, the Japan Industrial Robot Industry Association uses a broader definition of an industrial robot :

"An all-purpose machine equipped with a memory device and a terminal, and capable of rotation and of replacing human labor by automatic performance of movements."

This debate on the definition of an industrial robot simply indicates the continuous evolution which the field of robotics is undergoing year after year.

Independent of any specific definition, robotics specialists agree that a *robot manipulator*, which is the most important form of industrial robots, consists of the following physical components : a *mechanical system*, *sensors* and a *controller*. In the mechanical system the basic components are the *arm*, the *end-effector* and the *actuating mechanisms*. The arm usually consists of six rigid-links connected together in an open kinematic chain by revolute or prismatic joints, and allows the robot to position the end-effector in different locations in the workspace. The end-effector (gripper, welding torch, electromagnet, etc.) provides the means of manipulating objects or performing various other tasks in the workspace. The actuating mechanisms consist of power source(s), actuators (electric, hydraulic, pneumatic) and drive mechanisms (chains, gears, etc.). The sensors (visual, acoustics, force) measure and determine the state (positions, orientations, velocities) of the manipulator links and the end-effector. Furthermore, sensors measure and determine forces and moments exerted by the manipulated object on the manipulator. Finally, the controller is the device which supervises and regulates the programmed motion.

From a mathematical point of view, the study of robot manipulators includes topics such as : *modeling and design; robot arm kinematics, dynamics and control; trajectory planning; sensors; robot vision; robot control languages;* etc. Each of these topics can be studied on its own in great depth, as part of the education of a robotics specialist, or as an application area in different aspects of engineering. However, although each one of these topics is very important in robotics applications, a deep study of manipulator kinematics and dynamics is the cornerstone of successful utilization of today's robots

and those which are going to be used in the future.

Robot manipulator kinematics deals with the geometry and the time-dependent manipulator motion without consideration of forces and/or moments that cause the motion. In other words, it deals with the spatial configuration of the manipulator in the physical space. In particular, kinematics of robot manipulators is concerned with *configuration* and *motion* kinematic analysis. Configuration kinematic analysis deals with possible mathematical descriptions of the manipulator's spatial configuration as a function of time; and motion kinematic analysis deals with the first and second time derivatives of these configuration functions. The dynamics of a robot manipulator deals with the relation between actuator torques or forces and the manipulator's motion, considering its mass and inertial properties. These relations define the *dynamic equations of motion* of a robot manipulator which are fundamental to any robotic application. In particular, in the dynamic analysis of robot manipulators we deal with the following basic problems.

1.2 Basic Problems in Robot Manipulator Dynamics

As is well known, the dynamical performance of an n degrees-of-freedom system of rigid bodies can generally be described by n second order, usually coupled nonlinear differential equations which can be represented by a second order n-dimensional (n-D), coupled and nonlinear, vector differential equation. These differential equations are known as the *dynamic equations of motion* of the system and denote its *dynamic model.*

In a dynamic model of a system there are two main aspects with which one is concerned : *motion* and *forces.* The motion of a system is called its *trajectory* and consists of a sequence of desired *positions, velocities,* and *accelerations* of some point or points in the system. Forces are usually characterized as *internal* (or *constraint*) forces and *external* (or applied) forces. The external forces are the ones which cause motion.

In robotics, a dynamic robot model usually describes relationships between robot motion and forces causing that motion, so that given one of these quantities, we can determine the other. There are, therefore, the following two problems to be considered.

i) Forward Dynamics

Forward or *direct dynamics* problem is one where the forces which act on a robot are given and we wish to solve for the resulting motion. In its simplest form, the forward dynamics problem (FDP) can be expressed symbolically as a vector differential equation of the form

$$\ddot{\mathbf{q}} = \mathbf{h}\,(\mathbf{q}\,,\,\dot{\mathbf{q}}\,,\,\tau,\,\text{manipulator parameters}\,) \tag{1.2.1}$$

where, $\mathbf{q}$ is the vector of generalized coordinates (joint variables), $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are its derivatives with respect to time, τ is the (input) generalized force vector, i.e., the vector of joint torques and/or joint forces and the "manipulator parameters" are all those parameters which characterize the particular geometry and dynamics of a robot manipulator.

The importance of forward dynamics in robotics stems mainly from its use in simulation [8]. Simulation of robot motion is a way of testing control strategies or manipulator designs prior to the expensive task of working with the actual manipulator. In general, as we shall see later, equation (1.2.1) is not a simple equation for which an analytic solution can be provided easily. For a general robot manipulator, equation (1.2.1) is very complex since it is highly nonlinear with strong coupling between the joint variables. Hence, the solution of (1.2.1) for $\mathbf{q}$ requires complex procedures for evaluating $\mathbf{h}$ and for performing numerical integration. Fortunately, a solution for equation (1.2.1) is rarely required in practical applications. More often, we are interested in the following converse problem.

ii) Inverse Dynamics

The *inverse robot dynamics*, or simply *inverse dynamics* problem, is one in which

we need to determine the generalized forces that will produce a known motion trajectory. The inverse dynamics problem (IDP) can be described, mathematically, by an equation of the form

$$\tau = \mathbf{f}\ (\mathbf{q}\ ,\ \dot{\mathbf{q}}\ ,\ \ddot{\mathbf{q}}\ ,\ \text{manipulator parameters}\ ) \qquad (1.2.2)$$

where, as in (1.2.1), the manipulator parameters describe the particular robot manipulator, τ is the vector of the unknown generalized forces and $(\mathbf{q}\ ,\ \dot{\mathbf{q}}\ ,\ \ddot{\mathbf{q}}\ )$ is the given manipulator trajectory.

Inverse dynamics is very important in practical robot applications because it enables us to determine the profile of the generalized forces necessary to achieve a desired robot trajectory. Efficient computation of the inverse dynamics becomes particularly important when τ has to be evaluated online. This can arise in several practical situations, e.g., when the robot payload varies, or when the desired trajectory has to be modified online (e.g. for collision avoidance). Also, inverse dynamics plays an important role in many advanced robot control strategies where the inverse dynamics are used in the feedforward or feedback paths and may need to be computed online [9]. Moreover, to ensure convergence of the control scheme, the inverse dynamics computations may have to be performed very frequently. Consequently, the formulation and evaluation of these equations of motion affect the servo rate of the robot controller and partially determine the feasibility of implementing many control schemes online.

The forward and inverse dynamics problems are two problems which constitute what is usually known [9,10] as *robot manipulator dynamics*. However, since both problems are described by highly nonlinear and dynamically coupled equations, it can be of great assistance in many robotic applications if we have available the *linearized dynamic equations* of the robot manipulator. Thus, besides forward and inverse dynamics we may also include the following linearization problem in robot manipulator dynamics.

iii) Linearized Robot Dynamics

As is well known, the linearized dynamics of a nonlinear system can be described by the following first order vector differential equation (state-space form)

$$\delta\dot{\mathbf{x}} = \mathbf{A}(t)\delta\mathbf{x} + \mathbf{B}(t)\delta\mathbf{u} \tag{1.2.3}$$

where, the matrices $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are functions of time and $\delta\mathbf{u}$ and $\delta\mathbf{x}$ denote small perturbations in the input $\mathbf{u}$ and state $\mathbf{x}$, respectively, about some nominal (given) trajectory. Equation (1.2.3) describes the perturbed motion (for sufficiently small perturbations) of a dynamical system and is usually derived from the actual nonlinear dynamic equation (1.2.2) using a Taylor series expansion about a nominal trajectory [11,12]. The Taylor series expansion is applicable to nonlinear robot dynamics because, as can be easily shown, the nonlinearities in robot dynamics are analytic functions of their arguments. Therefore, the derivation of (1.2.3) from a nonlinear dynamic robot model at least in principle does not present any problems. However, applying the Taylor series expansion to a nonlinear system which has the complexity of a general robot manipulator is a challenging problem, especially if one attempts to derive efficient *computational* algorithms for determining the coefficient matrices of the linearized model.

Linearized robot dynamics may be used in manipulator control. This is best illustrated by the following example. In ideal situations, equation (1.2.2) provides the generalized forces which will drive a manipulator along a desired trajectory. However, in practice, because of perturbations resulting from modeling errors, unpredicted working conditions or payload variations, this cannot be achieved without the application of some control strategies, which are designed to compensate against these perturbations. Currently, there are many well established control strategies in the linear systems area. However, direct application of these linear control strategies to robotics is not possible, since, as we have already mentioned, the dynamic robot models defined by equation (1.2.2) are dynamically coupled and highly nonlinear. Therefore, one way in which these linear control schemes can be used is by obtaining linearized robot dynamic models

derived from equation (1.2.2) [13]. Another application of linearized robot dynamic models is in carrying out parameter sensitivity analysis of robot manipulator dynamics for the purpose of efficient manipulator design [12].

1.3 General Remarks on Robot Manipulator Dynamics

In principle, solving forward or inverse dynamics for rigid-link robot manipulators presents no difficulty. A robot manipulator is just a system of rigid bodies, and the equations of motion of such systems have been known for a long time. The real problem in robot dynamics is a practical one, namely, that of finding formulations for the equations of motion that lead to efficient computational algorithms. To derive these equations, we can use well established procedures from classical mechanics [14,15] such as those based on the equations of *Newton* and *Euler*, *Euler* and *Lagrange*, *Kane*, etc. However, the choice of a particular procedure determines the nature of the analysis and the amount of effort needed to state the equations of motion in the form of a computational algorithm. For example, in the Newton-Euler approach, the derivation of the equations *of motion* is based on direct application of Newton's and Euler's laws, while in the Lagrangian approach, the equations of motion are derived from two scalar quantities, namely, the *kinetic* and *potential energy*. Moreover, in the Newton-Euler approach, physical coordinate systems (usually Cartesian) are employed to express the equations of motion. Some of the coordinates may not be independent but related to others by kinematic constraints which are employed simultaneously with the equations of motion. In contrast, the Lagrangian approach usually employs linearly independent generalized coordinates. Therefore, the analysis and, consequently, the effort needed to derive the equations varies. Irrespective of the approach, the equations of motion for rigid-link open-chain robot manipulators can be stated in the following forms :

In a *closed-form* formulation, the equations of motion are usually described by the equation

$$\tau = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{1.3.1}$$

where τ is the vector of generalized forces, $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ denotes the joint trajectories, $\mathbf{D}(\mathbf{q})$ is the generalized inertia tensor of the manipulator, and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{G}(\mathbf{q})$ are the Coriolis and centrifugal, and gravitational coefficient vectors respectively. A closed-form representation, such as (1.3.1), can be used directly for solving the IDP, or it can be adapted easily for the FDP by solving for $\ddot{\mathbf{q}}$. This is probably the most attractive feature of closed-form formulations for the equations of motion of a robot manipulator. But, since these formulations are computationally inefficient [9,10], it is preferable to use more efficient recursive formulations for the equations of motion in practical real-time robot applications.

In a *recursive formulation*, the equations of motion of a robot manipulator are expressed implicitly in terms of recurrence relations between quantities describing various properties of the robotic system. Recursive formulations do not have the compact representation of the closed-form one but they too solve the IDP directly and, what is more important, they can be implemented in a very efficient manner. However, it is not possible to solve the FDP with the same recursive equations, without major modifications. But this is not a drawback, because even with major modifications we can solve the FDP efficiently. From the foregoing, it is not surprising that most of the research effort for solving manipulator dynamics has been directed at deriving efficient recursive formulations.

From the work that has been done to date on developing algorithms for computing manipulator dynamics it appears that there is a misconception that the computational efficiency of the algorithms depends on the formulations used for their derivation. Thus for example, it has been believed for some time that the algorithms derived from the Newton-Euler formulation are computationally more efficient than those derived using the Lagrangian formulation. It is felt that this confusion results from a lack of deeper understanding of the mathematical representations used to describe the equations of

motion. For example, in the Newton-Euler approach, the time variation in the orientation is generally represented by the *angular velocity vector*, and in the Lagrangian approach it is represented by the time derivative of a *rotation tensor*. But it can be shown [16], that the Lagrangian formulation will yield a similar algorithm to that obtained using the Newton-Euler formulation, if an equivalent representation of angular velocity is employed. Obviously, this result should be expected because the Lagrange equations can be derived from the Newton-Euler equations based on arguments of virtual work.

Therefore, in computing efficient robot manipulator dynamics, the issue is not which procedure from classical mechanics to use in the analysis. With proper analysis we can derive [17] exactly the same computational algorithms for solving manipulator dynamics. The real issue, in terms of computational efficiency, is which mathematical representation to use for expressing various physical quantities, when the nature of the quantities allows us to use more than one representation. Obviously, a particular representation dictates a certain mathematical analysis which leads to descriptions of the basic dynamic equations whose structure corresponds to that particular analysis Then, since the implementation of an algorithm depends on such structure, it follows that the computational efficiency of a particular algorithm will depend on the mathematical representation used to describe these physical quantities. Therefore, in searching for efficient computational algorithms to solve problems in robot manipulator dynamics, we have to search for a mathematical representation of the basic physical quantities of motion which will allow us to describe rigid body motion more efficiently.

1.4 OBJECTIVES AND MOTIVATION

Among the problems of robot manipulator dynamics, the IDP is the more important one. An efficient solution of this problem is a prerequisite for real-time robot applications, which in turn is necessary for flexible automation in a dynamically changing

environment. Therefore, the main objectives of this thesis are the analysis of the computational cost of solving the inverse dynamics problem and the development of algorithms with significantly reduced computational complexity.

In the last decade, a large number of algorithms has been proposed for solving inverse dynamics. The emphasis in most of these algorithms is placed on reducing their computational complexity by using analytical organization procedures and customization [18-20]. However, particular analytical organization procedures and customization are generally used in implementing the set of equations of an algorithm and not for deriving them. Moreover, in many cases, analytic procedures and customization are restricted to robot manipulators with a specific geometry. In this thesis, the emphasis is placed in improving the computational efficiency of the said algorithms through a more efficient formulation of the dynamic equations of motion and not through better implementation of existing formulations. Thus, our intention is to devise a new methodology for analysis and formulation of the dynamic equations of rigid body motion. The methodology has to be conceptually simple, easy to implement, and computationally efficient. To this end, we shall apply this methodology for solving in a computationally efficient manner the problems of inverse and forward dynamics of rigid-link, open-chain robot manipulators. Also, the methodology will be used for the derivation of linearized robot dynamic models in a computationally efficient manner.

In the dynamic analysis of rigid body motion we deal with relationships between the motion of a rigid body and forces and/or torques which cause (or result) from the motion. As we mentioned above, the representation of the physical quantities which are involved in the formulation of the equations of motion of a rigid bodies system, determines the kind of mathematical analysis that will be used in deriving these equations. Thus, in order to devise a new methodology, we must have a better understanding of the mathematical representations used to describe basic physical quantities. For example, in the classical Newtonian formulation of rigid body dynamics (which has been

applied successfully in deriving computational algorithms for solving inverse dynamics [21]), vectors are normally used to represent most of the physical quantities and, therefore, vector analysis is used for deriving the equations of rigid body motion. However, vector analysis, although powerful, has some major drawbacks. For example, one of its basic vector operations, namely, the vector cross product, is not associative and this limits our ability to manipulate effectively the equations of motion for more efficient solutions. To free ourselves from such limitations, we have to abandon vector analysis in preference to other more powerful mathematical procedures.

Vector analysis is imposed on classical Newtonian dynamics from the consideration that *angular rates* (i.e., linearly independent rates of change of rigid body orientation) constitute the components of a vector quantity, the *angular velocity vector*. This consideration also assigns a vector character to other physical quantities which are defined in terms of the angular velocity vector such as : *angular acceleration, angular momentum, external torque*, etc. Therefore, in searching for other mathematical procedures for describing classical Newtonian dynamics, we have to examine other alternatives in representing angular velocity which is one of the basic physical quantities involved in rigid body dynamics.

As is well known [22], angular velocity can also be described by a second order skew-symmetric Cartesian tensor, the *angular velocity tensor*. Obviously then, the tensor representation of angular velocity calls for Cartesian tensor analysis to be applied in rigid body dynamics. However, application of Cartesian tensor analysis (within the framework of the Newtonian approach) in rigid body dynamics requires that all the other physical quantities which are defined in terms of the angular velocity be treated as Cartesian tensors instead of vectors. Thus, we have to examine if a Cartesian tensor representation of basic physical quantities such as : angular acceleration, angular momentum, external torque, etc., simplifies the equations of rigid body motion.

The use of tensor analysis is obviously known in rigid body dynamics. However, most of the time the analysis is performed in the *configuration space* of the rigid bodies system which is generally a *Riemanian space*, i.e., a non-Euclidean *manifold* [23,24]. In this thesis, we shall use tensors to analyze the motion of a rigid bodies system, but the analysis will be carried out in Euclidean space instead of on nonlinear manifolds, i.e., we shall use *Cartesian tensor analysis* [25,26]. To do this, we shall need to review basic results from Cartesian tensor analysis and, in particular, we shall need to understand the relations between three dimensional vectors and second order skew-symmetric Cartesian tensors.

Finally, almost all existing algorithms which solve forward or inverse manipulator dynamics have a structure which requires that all quantities involved in these algorithms be computed online (except the configuration independent geometric and dynamic parameters of the individual links). This obviously is a consequence of the underlying modeling schemes which have been used to derive the algorithms, because the structure of an algorithm depends on the underlying scheme. However, from a computational point of view it is desirable to devise algorithms which allow us to compute *off-line* as many quantities as possible and at the same time, to keep the *online* computations as simple as possible. Therefore, in order to derive computationally efficient algorithms for solving the inverse and forward manipulator dynamics problems, we have to examine if it is possible to devise a modeling scheme for the robot manipulators which allows us to compute as many configuration independent kinematic or dynamic parameters of the robot manipulator off-line as possible.

1.5 Outline of the Thesis

This thesis presents a new methodology for the analysis and formulation of computationally efficient algorithms for solving basic problems of robot manipulator dynamics. The layout of the thesis is as follows : Chapter III is concerned with Cartesian tensor

analysis, based on which the new methodology is devised, and Chapter IV demonstrates how this theory can be applied to rigid body motion. New algorithms for solving the problems of inverse and forward dynamics of rigid-link open-chain robot manipulators are proposed in Chapters V and VI, respectively, while Chapter VII deals with linearized dynamic robot models. In particular, the main contents of each chapter are as follows :

Chapter II : Preliminaries.

This chapter introduces the notation to be used throughout the thesis. The configuration kinematic analysis of rigid bodies is briefly reviewed. Also, this chapter presents some relevant robot manipulator terminology as well as a configuration kinematic analysis of rigid-link open-chain robot manipulators.

Chapter III : Cartesian Tensor Analysis.

This chapter introduces relevant definitions and some basic algebraic Cartesian tensor operations and outlines the structural symmetries of second order Cartesian tensors. Also, based on tensor vector invariants, operators between vectors and skew-symmetric Cartesian tensors are defined and a geometric characterization of second order skew-symmetric Cartesian tensors is given. Furthermore, based on these operators, important propositions are stated which establish some basic tensor identities.

Chapter IV : Cartesian Tensors and Rigid Body Motion.

Based on Cartesian tensor analysis, kinematic and dynamic aspects of rigid body motion are considered in this chapter. In particular, the angular acceleration tensor is introduced which together with the angular velocity tensor is shown to be a very powerful tool for describing the motion of a rigid body. Then, by using these two tensors, a tensor representation for the angular momentum and external torque surfaces naturally and leads to a Cartesian tensor description for the Newtonian formulation of rigid body motion. To this end, a tensor formulation for the generalized Euler equation, which describes pure rotational motion, is derived. This new formulation of the Euler equation

has the same simplicity as the classical vector formulation but can be implemented far more efficiently.

Chapter V : Inverse Dynamics of Rigid-Link Open-Chain Robot Manipulators.

A brief survey of existing methods for solving the inverse dynamics problem for rigid-link open-chain robot manipulators is followed by some observations and remarks on various issues concerning the computational efficiency of some "classical" algorithms for solving this problem. It is then shown that by using the Cartesian tensor description of the Newtonian formulation of rigid body motion and utilizing two different modeling schemes, computationally efficient algorithms for solving the IDP can be devised. The computational complexity of these algorithms is shown to be reduced significantly when compared with other algorithms which are based on the classical vector formulation of rigid body motion. Also, it is demonstrated that these algorithms can be cast in a form where their computational efficiency is actually independent of the particular procedure of classical mechanics which has been used for their derivation.

Chapter VI : Forward Dynamics of Rigid-Link Open-Chain Robot Manipulators.

The Cartesian tensor analysis and the modeling scheme which has been proven successful in solving efficiently the IDP are used in this chapter to facilitate the solution of the FDP for rigid-link open-chain robot manipulators. After a brief review of the composite rigid body method [8], we introduce a new algorithm for computing the generalized inertia tensor of rigid-link open-chain robot manipulators. By combining this algorithm with algorithms for solving efficiently the IDP, we improve significantly the computational efficiency of solving the problem of forward dynamics.

Chapter VII : Linearized Dynamic Models for Rigid-Link Open-Chain Robot Manipulators.

This chapter is concerned with the linearization of the dynamic equations of motion for rigid-link open-chain robot manipulators. Using a Taylor series expansion, we derive the

associated linearized dynamic robot models of the nonlinear dynamic models presented in Chapter V. It is then shown that the coefficient sensitivity matrices of these linearized dynamic models can be computed efficiently based on appropriate Cartesian tensor formulations. Also, in this chapter Cartesian space descriptions of the equations of motion for the rigid-link open-chain robot manipulators are reviewed and a method for deriving their associated Cartesian space linearized dynamic models is proposed.

Chapter VIII : Conclusions and Future Work.

Basic contributions of the research described in this thesis are summarized in this chapter. Also, possible extensions of the results to specific, or perhaps, new problems are discussed. A brief section on topics that have *not* been treated in this thesis is included as suggestions for those who might wish to contribute to this exciting area of research.

1.6 REFERENCES

[1] A. Cohen, and J. D. Erickson, "Future Uses of Machine Intelligence and Robotics for the Space Station and Implications for the U.S. Economy", *IEEE J. Robotics and Automation,* RA-1, No. 3, pp. 117-123, 1985.

[2] A. K. Bejczy, and Z. Szakaly, "Universal Computer Control System (UCCS) for Space Telerobots", Proc. *1987 IEEE Int. Conf. on Robotics and Automation,* pp. 318-125, Raleigh, NC, March 31-April 3, 1987.

[3] K. Edahiro, "Development of Underwater Robot Cleaner for Marine Live Growth in Power Station", Proc. *'89 ICAR Int. Conf. on Advanced Robotics,* pp. 99-106, Tokyo, Japan, Sept. 1983.

[4] K. G. Engelhardt, "Applications of Robots to Health and Human Services", Conf. Proc. *Robots 9 : Current Issues, Future Concerns,* pp. 14-48 to 14-65, Detroit, Michigan, June, 1985.

[5] G. N. Saridis, "Robotic Control to Help the Disabled" *Recent Advances in Robotics,* C. Ben and S. Hackwood, *Eds.,* John Wiley, New York, 1985.

[6] V. Shimon, *Eds., Handbook of Industrial Robotics,* John Wiley, New York, 1985.

[7] R. K. Miller, *Industrial Robot Handbook,* Fairmont Press, Indian Trail, NY, 1987.

[8] M. W. Walker, and D. E Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", *ASME J. Dynamic Systems, Measurement and Control,* Vol. 104, pp. 205-211, 1982.

[9] M. Brady *et al., Eds., Robot Motion : Planning and Control,* Cambridge, MA, MIT Press, 1982.

[10] R. Featherstone, *Robot Dynamics Algorithms,* Kluwer Academic Publishers, Boston MA, 1987.

[11] C. A. Balafoutis, P. Misra, and R. V. Patel, "Recursive Evaluation of Linearized Dynamic Robot Models", *IEEE J. Robotics and Automation,* RA-2, pp. 146-155, 1986.

[12] C. P. Neuman, and J. J. Murray, "Linearization and Sensitivity Functions of Dynamic Robot Models", *IEEE Trans. Systems, Man, and Cybernetics,* Vol. SMC-14, pp. 805-818, 1984.

[13] P. Misra, R. V. Patel, and C. A. Balafoutis, "Robust Control of Linearized Dynamic Robot Models", *Robot Manipulators : Modeling, Control and Education,* M. Jamshidi, J. Y. S. Luh, and M. Shahinpur, *Eds.,* North-Holland Publishing Co., New York, 1986.

[14] H. Goldstein, *Classical Mechanics 2nd Ed.,* Addison-Wesley, Reading, MA, 1981.

[15] T. R. Kane, P. W. Likins, and D. A. Levinson, *Spacecraft Dynamics,* McGraw-Hill, New York, 1983.

[16] W. M. Silver, "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators", *Int. J. Robotics Research,* Vol. 1, pp. 60-70, 1982.

[17] C. A. Balafoutis, R. V. Patel, and J. Angeles, "A Comparative Study of Lagrange, Newton-Euler and Kane's Formulation for Robot Manipulator Dynamics" *Robotics and Manufacturing : Recent Trends in Research, Education, and Applications,* M. Jamshidi, J. Y. S. Luh, H Seraji, and G. P. Starr, *Eds.,* ASME Press, New York, 1988.

[18] J. W. Burdick, "An Algorithm for Generation of Efficient Manipulator Dynamic Equations", *Proc. 1986 IEEE Int. Conf. Robotics and Automation,* pp. 212-218, San Francisco, CA, Apr. 1986.

[19] C. J. Li, "A Fast Computational Method of Lagrangian Dynamics for Robot Manipulators", *Int. Journal of Robotics and Automation*, Vol. 3, No. 1, pp. 14-20, 1988.

[20] J. J. Murray, and C. P. Neuman, "Organizing Customized Robot Dynamics Algorithms for Efficient Numerical Evaluation", in *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-18, No. 1, pp. 115-125, 1988.

[21] J. Y. S. Luh, M. W. Walker, and R. P. Paul, "On-Line Computational Scheme for Mechanical Manipulators", *ASME J. Dynamic Systems, Measurement and Control*, Vol. 102, pp. 69-79, 1980.

[22] O. Bottema, and B. Roth, *Theoretical Kinematics*, North-Holland Publishing Co., Amsterdam, 1978.

[23] L. Brillouin, *Tensors in Mechanics and Elasticity*, Academic Press, New York, 1964.

[24] I. S. Sokolnikoff, *Tensor Analysis : Theory and Applications to Geometry and Mechanics of Continua*, John Wiley & sons, New York, 1965.

[25] H. Jeffreys, *Cartesian Tensors*, Cambridge University press, Cambridge, 1961.

[26] A. M. Goodbody, *Cartesian Tensors : With Applications to Mechanics, Fluid Mechanics and Elasticity*, Ellis Horwood, England, 1982.

CHAPTER II

PRELIMINARIES

This chapter introduces the notation, presents some basic concepts from *rigid body kinematics*, defines relevant robot terminology, and deals with the *configuration* kinematic analysis of *rigid-link, open-chain robot manipulators*. The chapter has two main sections : Section 2.2 contains results from rigid body kinematic analysis. In particular, the *configuration* of rigid bodies in the real world or physical space is defined and the *finite displacement* of rigid bodies in this space is reviewed. Section 2.3 is concerned with the geometric description of rigid-link open-chain robot manipulators and defines the *joint* and *Cartesian* space descriptions for their configuration.

2.1 NOTATION

Throughout the text, boldface lower case roman letters are used to denote position vectors. Subscripts indicate, in order, the tail and the head of a position vector, and a superscript indicates the coordinate system with respect to which the position vector is expressed. Upper case boldface roman letters are used to denote second order tensors or vectors of forces and moments. From the context it will be clear if a tensor or a vector is considered. A second order skew-symmetric Cartesian tensor associated with a vector will be denoted with a tilde ($\tilde{}$) above the boldface lower or upper case roman letter denoting this vector. Subscripts denote a point on a link with respect to which the tensors (or the force and moment vectors) are defined, and superscripts denote the coordinate system with respect to which the tensors (or the force and moment vectors) are expressed. The superscript for tensors or vectors expressed in the base frame (inertial frame) is omitted. The *coordinate matrix*, associated with a tensor or a vector, will be denoted by the corresponding lower or upper case italic letter.

2.2 Rigid Bodies And Their Finite Displacement

The main objective in robotics is to manipulate objects in a static or dynamically changing environment, and one of the basic requirements for achieving this goal is to describe effectively, i.e., simply and accurately, objects such as *points* and *rigid bodies* relative to some coordinate system. In this section, we deal with the configuration kinematic analysis of points and rigid bodies. In particular, we review some of the possible approaches for describing the location and displacement of points and rigid bodies in physical space.

2.2.1 The Configuration of Points and Rigid Bodies in Physical Space

A description of the static or instantaneous location of an object in a space relative to a reference coordinate system is referred to as the *configuration* of the object [1]. The configuration is usually expressed as a function of a number of linearly independent variables which are known as *generalized coordinates*. The number of the generalized coordinates defines how many *degrees-of-freedom* the object has relative to the reference coordinate system.

In general, the complexity of the function, which defines the configuration of an object in terms of the generalized coordinates, depends on the reference coordinate system. The reference coordinate system characterizes the space where the object belongs and can be *linear* or *curvilinear*. Usually, the configuration of an object, in a curvilinear reference coordinate system is very complex, as opposed to a linear or Cartesian reference coordinate system where its configuration has usually a simpler form. In robotics, it is of prime concern to describe the configuration of objects in the real world and, to our advantage, the real world or physical space can be modeled as a *three dimensional Euclidean space*. This allows us to consider a Cartesian coordinate system as a reference system relative to which, as we shall demonstrate later, the configuration of points or rigid bodies assumes relatively simple forms.

The configuration of a point in a general space is completely specified by its point-coordinates relative to a reference coordinate system. For points in the three dimensional physical space, their point-coordinates are functions of three generalized coordinates. Moreover, in the physical space which is a Euclidean space, we can identify points with vectors, since in this space, transformations of point-coordinates are identical to transformations of vector components [10]. This identification allows us to specify the configuration (i.e., the location) of a point in the physical space by using the components of a vector. This vector is referred to as the *position vector* of the point under consideration. Therefore, the configuration of a point in the physical space can be described, in an orthogonal Cartesian coordinate system, by a three dimensional vector, its position vector.

A description for the configuration of a rigid body in the physical space is more involved, compared to that of a point. As is well known [1-4], a rigid body in the physical space possesses six degrees-of-freedom. This implies that its configuration will be described in a reference coordinate system in terms of six generalized coordinates. Now, if we consider the configuration of a rigid body to be described by a vector (as we did in the case of a point), this vector must have six independent components. It is obvious then, that this six dimensional vector does not belong to the physical space. It belongs to a six dimensional space which is not Euclidean and is known as the *configuration space*. Therefore, by assuming a vector description for the configuration of a rigid body we have to use a non-Euclidean space and hence curvilinear reference coordinate systems. This approach of describing the configuration of a rigid body leads us, in general, to very complex functional expressions.

Conventionally, we overcome these difficulties by grouping the six generalized coordinates, which describe the configuration of a rigid body into two sets. The first set contains the generalized coordinates which describe the *orientation* of the rigid body. The other set contains the remaining three generalized coordinates and they describe the

configuration (position) of a point on the rigid body.

To implement this scheme, we first associate a *frame* with the rigid body. A frame is a representation for a coordinate system so that the representation includes the possibility that the coordinate system may be displaced (translated) and/or rotated with respect to an other coordinate system. In other words a frame contains a coordinate system whose orientation defines the "frame orientation" and is known as the *frame coordinate system*, and a position vector, which defines the *origin* of the frame coordinate system. The frame coordinate system is assumed to have a fixed relationship with the rigid body. For that, sometimes, the frame coordinate system is refer to as the *body coordinate system.* This allows us to identify the orientation of the rigid body with that of the frame. Moreover, to simplify the description, we consider the frame coordinate system to be an *orthogonal Cartesian system.* Therefore, it is obvious that the configuration of the rigid body will be completely specified, relative to a reference coordinate system, if we specify the orientation of an orthogonal Cartesian coordinate system (the frame coordinate system) and its position vector.

As we noted above, position vectors can be described easily. Therefore, to complete the description for the configuration of a rigid body, we need to describe the orientation of a frame coordinate system relative to a reference one. This can be easily accomplished by considering an intermediate coordinate system which has the same orientation as the reference system, but whose origin is the same as that of the frame coordinate system. Then, we need only to describe the orientation of the frame coordinate system relative to the intermediate one, i.e., we need to describe the relative orientation of two Cartesian coordinate systems with a common origin.

There are many ways of specifying the orientation of a Cartesian coordinate system relative to another one with a common origin. As is well known [1], the orientation of two Cartesian coordinate systems with a common origin is described by a linear transformation. Here, since the coordinate systems are orthogonal, the linear

transformation will be an orthogonal one. Moreover, from physical considerations (-the coordinate systems represent orientations of rigid bodies), the linear transformation is *proper* (i.e., it has determinant equal to one) and so it represents a rotation. Therefore, this results in the problem of how to describe a *rotation.*

One of the most common methods to be found in the literature [1-6] which describes a rotation, is that of using a *real orthogonal* 3×3 matrix. The entries of this matrix are the *direction cosines* which relate the axes of the two coordinate systems -the reference and the rotated one. A set of nine direction cosines completely specify the rotation between any two Cartesian systems. Of course, the set of the nine direction cosines does not form a set of independent generalized coordinates, since as is well known [1] they satisfy six orthogonality relationships. However, the use of direction cosines to describe the orientation of one Cartesian coordinate system with respect to another has a number of important advantages. The most obvious is that they permit the use of Cartesian coordinate systems in describing the orientation of a rigid body, therefore avoiding the need for a curvilinear coordinate system for describing the configuration of a rigid body.

As we mentioned above, the nine direction cosines of a 3×3 real orthogonal matrix have only three degrees-of-freedom which may be specified in terms of three linearly independent parameters. However, it is a well known fact [5], that there is no 1-1 global representation for a rotation matrix in terms of three independent variables (generalized coordinates). Nevertheless, in many practical applications and for a restricted domain it is possible to find three linearly independent variables, which can serve as generalized coordinates to describe a rotation. These generalized coordinates can be chosen in a number of ways. A common approach is to choose a particular sequence of rotation angles (α, β, γ) about the axes of an orthogonal coordinate system. The *roll, pitch and yaw* or the **z –y –x** *Euler angles* are examples of this approach. Finally, an alternative way of describing rotations is to use more than three variables, which obviously will not

be linearly independent and usually the description will not be 1–1. For example, we can use *quaternions, spinors, Pauli spin matrices, special unity* 2×2 and 3×3 matrices [1-8], *geometric* or *frame invariants* [9], etc.

When we use generalized coordinates to describe the configuration of a rigid body, for notational convenience, we sometimes consider the three "generalized" angles α, β, γ which describe a rotation (over a restricted domain) as the components of an "orientation vector". This "orientation vector" is then combined with the position vector $[\, x \;\; y \;\; z \,]^T$ of a point to produce a six dimensional "vector" $\chi = [\, x \;\; y \;\; z \;\; \alpha \;\; \beta \;\; \gamma \,]^T$, which describes the configuration of a rigid body. We usually refer to χ as the *Cartesian configuration vector* for the rigid body. The set of all Cartesian configuration vectors then defines the *Cartesian (configuration) space* for the rigid body.

Remark 2.1 : As mentioned above, the "orientation vector" is created merely for notational convenience. It is not a valid representation for a rotation. Mathematically, a rotation is a *second order tensor* which is not a skew-symmetric tensor. Therefore, it is impossible to be represented by a vector, which is a first order tensor (see Chapter III). Another, probably simpler way to see that the "generalized" angles (α, β, γ) does not form the components of a vector is the following. The composition (multiplication) of finite rotations is known to be associative but not commutative. Now, the only vector operations which produce a vector are addition and vector cross product. But vector addition is commutative and the vector cross product is not associative. Therefore, it is obvious that vectors do not represent finite rotations, because neither vector addition nor vector cross product is compatible with the composition of rotations. Hence, the "orientation vector" does not exist. Therefore, the Cartesian configuration vectors are not "real" vectors. This implies that the terminology "Cartesian space" or "Cartesian vector" is used in robotics in a broader sense of that used in linear algebra.

In the following section, we shall analyze briefly the relationships between the configurations of the same rigid body in two different locations in the physical space.

2.2.2 On the Finite Displacement of a Rigid Body

As is well known, the difference between the position vectors for the same point on a rigid body at two different locations of this body in the physical space is referred to as the displacement of that point in space. In this thesis, we shall define displacement in a broader sense to also include the difference between two orientations of the same rigid body.

A finite *displacement* in the physical space is expressed mathematically as a transformation of the 3-D Euclidean space $\mathbf{E}^3$ into itself, with the property that it preserves the Euclidean distance. To elaborate, if $\mathbf{W} : \mathbf{E}^3 \rightarrow \mathbf{E}^3$ is a transformation and

$$\mathbf{p}' = \mathbf{W}\,\mathbf{p} \tag{2.2.1}$$

denotes the action of $\mathbf{W}$ on a point $\mathbf{p}$ in $\mathbf{E}^3$, then $\mathbf{W}$ defines a displacement if $(\mathbf{p}' - \mathbf{r}')^2 = (\mathbf{p} - \mathbf{r})^2$ for all $\mathbf{p}$ and $\mathbf{r}$ in $\mathbf{E}^3$. Clearly, $\mathbf{W}$ has been defined as a point-transformation, but since in an Euclidean space points are identified by their position vectors, we can view $\mathbf{W}$ as a vector transformation of $\mathbf{E}^3$ into itself.

It can be shown [2] that displacements form a group. It is a subgroup of the group of transformations and it consists of those transformations which leave the distance of any two points *invariant*. It can also be shown [2] that displacements are angle-preserving transformations. In particular, right angles correspond to right angles. This implies that not only the distance between two points but also the distance between a point and a linear subspace, and the distance between two parallel subspaces are invariant under displacement. Finally, if for a certain displacement $\mathbf{W}$, a point $\mathbf{p}$ coincides with its image $\mathbf{p}'$, this point is called a *fixed* point of $\mathbf{W}$.

The concept of displacement is fundamental to rigid body kinematics. It provides the mathematical apparatus for the study of rigid body motion. To see this, we notice that a rigid body is defined as a system of mass points subject to the holonomic constraints that the distances between all pairs of points remain constant throughout any

motion. Now, since displacement is a distance and angle preserving transformation, it is obvious that rigid body motion can be described, mathematically, as a displacement between two distinct configurations of the rigid body.

A general displacement or motion of a rigid body is best analyzed by considering the following two special displacements.

Translation : The transformation T_d, which is defined by the equation

$$\mathbf{p}' = T_d\,\mathbf{p} \triangleq \mathbf{p} + \mathbf{d} \tag{2.2.2}$$

where $\mathbf{d}$ is a fixed vector, is obviously a displacement. It is the simplest displacement and is called a *translation.* The vector $\mathbf{d}$ in (2.2.2) is called the *vector of translation.* As we can see from (2.2.2), if $\mathbf{p}$ and $\mathbf{r}$ are two points, then the vector $\mathbf{w}' = \mathbf{p}' - \mathbf{r}'$ not only has the same length as $\mathbf{w} = \mathbf{p} - \mathbf{r}$ but is also parallel to it. Also, it is obvious from (2.2.2) that translations form a commutative group, which is a subgroup of the group of displacements. Moreover, from equation (2.2.2) it can also be seen that a translation is not a linear transformation (it does not map the origin of the space in to itself) and has no fixed points.

The second special displacement is the familiar rotation which can be defined as follows.

Rotation : A dispacement $\mathbf{A}$ for which a point o is a fixed point is called a *rotation* about o.

It is well known [2], that rotations about a point o constitute a subgroup of the displacement group. The rotation group is not commutative. Also, in contrast with translations, rotations are linear transformations.

A rotation, in general, can be interpreted in two ways. First, we consider a rotation $\mathbf{A}$ as an *operator* which acts on a vector $\mathbf{p}$ and produces another vector $\mathbf{p}'$. This is an *active* point of view. In this approach the space is described in an invariant coordinate system, relative to which all vectors are rotated. So, $\mathbf{p}$ and $\mathbf{p}'$ are two different

vectors, expressed in the same coordinate system. In the second approach, we consider the same rotation $\mathbf{A}$ as a transformation which acts on a reference coordinate system $\{ \mathbf{e} \}$ and produces a new reference coordinate system $\{ \mathbf{e}' \}$. This is a *passive* point of view. In this approach, the actual vector remains invariant and only the reference coordinate system is rotated. An invariant vector is represented by $\mathbf{p}$ relative to the old coordinate system, and by $\mathbf{p}'$ relative to the new one. Both interpretations of a rotation define the same action on a vector and both interpretations are described mathematically using the same algebra. We express the action of a rotation $\mathbf{A}$ by writing

$$\mathbf{p}' = \mathbf{A}\,\mathbf{p} \tag{2.2.3}$$

The dimension of the Euclidean space $\mathbf{E}^n$, where a rotation is defined is fundamental to the analysis of rotations. Thus for n even there are in general no fixed points different from o. For n odd ($n = 2m + 1$, $m = 0, 1, \cdots$) we always have at least one line of fixed points, the *axis of the rotation.* A consequence of the general theory of rotations in odd dimensional Euclidean spaces is the following Theorem.

Theorem 2.1 (Euler) : If a rigid body undergoes a displacement leaving fixed one of its points, o, then a set of points of the body, lying on a line that passes through o, remains fixed as well.

A corollary of Euler's theorem, sometimes called *Chasles' theorem*, states the following result.

Theorem 2.2 (Chasles) : The most general displacement of a rigid body is a translation plus a rotation.

Proofs for these theorems can be found in any book on classical mechanics, e.g. see [1-4]. An important consequence of Chasles' Theorem is that a general displacement of a rigid body can be written as

$$\mathbf{p}' = \mathbf{A}\,\mathbf{p} + \mathbf{d} \tag{2.2.4}$$

where $\mathbf{A}$ is a pure rotation and $\mathbf{d}$ is the translation vector of a pure translation. Equation (2.2.4) is very important in rigid body kinematics. It expresses a general

displacement explicitly in terms of a rotation $\mathbf{A}$ about a point o and a translation of o by $\mathbf{d}$. However, a compact representation for a general displacement, as that in equation (2.2.1) is more appealing, especially when one deals with a series of displacements, as is often the case in robotics. To achieve a compact representation for a general displacement, in terms of a rotation and a translation, we proceed as follows.

Let $\mathbf{W}$ be a general displacement, o an arbitrary point and o′ its image under $\mathbf{W}$. Now, if $\mathbf{T_d}$ is a translation with a vector $\mathbf{d}$ which transfers o into o′, then $\mathbf{T_d^{-1}W}$ is clearly a rotation $\mathbf{A}$ about o. Here, $\mathbf{T_d^{-1}}$ denotes the inverse translation of $\mathbf{T_d}$. Now, if we write $\mathbf{T_d^{-1}W} = \mathbf{A}$ it follows that

$$\mathbf{W} = \mathbf{T_d A} \tag{2.2.5}$$

i.e., a general displacement can be written as the product of a rotation and a translation. We usually refer to a transformation which may not only change the orientation but also the origin of a coordinate system as a *homogeneous transformation*. From the foregoing, equation (2.2.5) defines a homogeneous transformation. Obviously, the homogeneous transformation $\mathbf{W}$ as a product of a nonlinear and a linear transformation is a nonlinear transformation.

Homogeneous transformations are best analyzed in terms of *homogeneous coordinates* [11, 12]. The coordinates of a point, line or plane are called homogeneous if the entity they determine is not altered when the coordinates are multiplied by the same scalar. As is well known, a three dimensional vector $\mathbf{p}$ in a coordinate system has a (3×1) column matrix representation. In a homogeneous coordinate system a three dimensional vector has a (4×1) column matrix representation. The last entry of the column contains a scaling factor which can be chosen to be equal to 1. With the scaling factor equal to 1, the homogeneous coordinate matrix of $\mathbf{p}$ is given by [12]

$$p = [\, p_x \;\; p_y \;\; p_z \;\; 1 \,]^T \tag{2.2.6}$$

where $p = [\, p_x \;\; p_y \;\; p_z \,]^T$ is the coordinate matrix of $\mathbf{p}$ in three dimensional Euclidean space.

A translation homogeneous transformation $\mathbf{T_d}$ with vector of translation $\mathbf{d}$ has a (4×4) matrix representation in homogeneous coordinates given by [12]

$$T = \left[\begin{array}{ccc|c} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.2.7}$$

where $d = [\, d_x \;\; d_y \;\; d_z \,]^T$ is the coordinate matrix of $\mathbf{d}$ in three dimensional Euclidean space.

Similarly, a rotation homogeneous transformation $\mathbf{A}$ has a (4x4) matrix representation in homogeneous coordinates given by

$$A = \left[\begin{array}{c|c} A & 0 \\ \hline 0^T & 1 \end{array}\right] \tag{2.2.8}$$

where, A is the usual (3×3) matrix representation (via direction cosines) of a rotation, and $\mathbf{0}$ is the three dimensional zero vector. Note that we use the same notation for the (3x3) matrix and homogeneous matrix representations of a rotation. We shall rely on the context to distinguish between the two representations.

Now, as we can see by using equation (2.2.7) and (2.2.8), the homogeneous transformation $\mathbf{W}$, defined by (2.2.5), has a (4x4) homogeneous matrix representation given by,

$$W = \left[\begin{array}{c|c} A & d \\ \hline 0^T & 1 \end{array}\right] \tag{2.2.9}$$

Equation (2.2.9) allows us to express the general displacement of a vector (i.e., equation (2.2.1)) in homogeneous coordinates as follows.

$$\left[\begin{array}{c} p' \\ \hline 1 \end{array}\right] = \left[\begin{array}{c|c} A & d \\ \hline 0^T & 1 \end{array}\right] \left[\begin{array}{c} p \\ \hline 1 \end{array}\right] \tag{2.2.10}$$

Equation (2.2.10) is equivalent to the two equations

$$p' = Ap + d \tag{2.2.11a}$$

$$1 = 1 \qquad (2.2.11b)$$

where obviously (2.2.11a) gives the matrix representation in the three dimensional space of equation (2.2.4).

From the foregoing, we have two ways of analyzing a general rigid body displacement transformation : either equation (2.2.4) or equation (2.2.1) can be used.

As we shall see in later chapters, equation (2.2.4), with a matrix representation given by equation (2.2.11a), is suitable for kinematic analysis of rigid body motion when computational issues are of main concern. Equation (2.2.1), with a matrix representation given by (2.2.10), leads to compact representations, but with significantly higher computational complexity. The two representations are, of course, equivalent and equations (2.2.10) and (2.2.11) provide the bridge between them.

Remark 2.2 : Besides describing general displacements, homogeneous transformations are often used in robotics [12] to represent coordinate frames, i.e., relative configurations of rigid bodies. Thus, for example, equation (2.2.9) can be used to define the homogeneous coordinate matrix representation for the configuration of a rigid body relative to another reference coordinate system.

Based on these preliminaries, we can now introduce the physical system on whose dynamic analysis this thesis is focussed.

2.3 Robot Manipulators

As we mentioned in Chapter I, robot manipulators or robot arms are the most important form of robotic systems in use today. The dynamic analysis of such robots is therefore of practical importance. A general description for the physical components of a robot manipulator has been given in Chapter I. In this section, we provide in more details a "geometric" description for the arm of a robot manipulator and introduce some relevant terminology.

2.3.1 Description of Robot Manipulators

A robot manipulator is essentially a mechanical device that can be programmed to automatically manipulate objects in physical space (the real world). The arm or articulate portion of a robot is usually constructed as a series of coupled bodies, known as *links*, which together constitute what is called a *kinematic chain*. If every link is connected to at least two other links, the kinematic chain is said to be *closed*, and such a mechanism is called a *linkage*. If, however, some of the links are connected to only one other link, then the kinematic chain is said to be *open* or *serial-type* and such a mechanism is called a *manipulator*. Therefore, depending on their articulate portion, we can have robots with closed or open kinematic chains. However, since the kinematic and dynamic analyses of closed kinematic chains is more involved [15-17], most industrial robots have open kinematic chains i.e., a manipulator, with some form of *end-effector* attached to the final link. Depending on the intended applications, the end-effector can be a *gripper*, a *welding torch* or other device. We usually refer to this class of industrial robots as *robot manipulators*.

Although, in reality, all mechanical devices are flexible to a degree, the links of present days industrial robot manipulators are made of quite heavy and rigid material and are usually modeled as rigid bodies. This provides a realistic approximation which allows us to simplify considerably their kinematic and dynamic analyses. However, in recent years, some research has been directed towards modeling and analysis of robot manipulators with flexible links [18-21]. Flexible link manipulators are made of light weight material and may be useful for space applications but have not yet become popular in industry.

A *kinematic pair* is the coupling of two adjacent links. In current industrial manipulators the most frequently encountered kinematic pairs (and the simplest ones) are the *revolute pair*, which allows only relative *rotational motion* about a single axis (the *joint*), and the *prismatic pair*, which allows only relative *translational motion* along a single axis

(the joint). For these kinematic pairs since motion is allowed in a single direction only, one parameter (variable) is sufficient for specifying the relative motion between two adjacent links. This implies that revolute or prismatic pairs are characterized by one *degree of freedom.* The corresponding variable which measures the linear or rotational relative motion of a kinematic pair is referred to as the *generalized coordinate* of that kinematic pair or joint.

As we saw in Section 2.2, there are six degrees-of-freedom associated with the configuration of a rigid object. Therefore, if the links of a manipulator are connected by only revolute and/or prismatic joints (as is the case with most current industrial manipulators), then there must be at least six such links (and hence joints) if the manipulator is to be capable of arbitrarily positioning objects in a three dimensional space. Otherwise stated, any manipulator must have at least six degrees-of-freedom (links and/or joints) in order for it to achieve arbitrary real world configurations. There are, however, many manipulators that have fewer than six degrees-of-freedom because they are designed to perform tasks which do not require such freedom. Also, there are robot manipulators which have been designed to have more than six degrees-of-freedom. Such manipulators are called *redundant* arm manipulators. These manipulators are particularly useful in environments where collision avoidance [22] is important. However, in this thesis we shall be concerned only with the kinematic and dynamic analyses of non redundant robot manipulators with rigid links.

To be able to identify the links and the joints of a robot manipulator, we number the links (and implicitly the joints) from zero to n successively - zero being the first link, which is known as the *base*, which is fixed and n the last one which corresponds to the free end. With this scheme, tne joint which connects the $(i-1)$-th and the i-th links is referred to as the i-th joint. The end-effector (if it exists) is usually considered as the $(n+1)$-th link which is rigidly attached to the n-th link with no joint between them.

Also, to be able to specify the configuration, of each link relative to an *inertial* or any other coordinate system, we associate with each link a frame which we denote by $\mathbf{e}_i \equiv \{ \mathbf{x}_i \ \mathbf{y}_i \ \mathbf{z}_i \ \mathbf{o}_i \}$ $i = 0,1,2 \ldots n$, and refer to as the i-th link frame. The i-th frame is composed of the frame coordinate system, which we denote by $\mathbf{e}_i \equiv \{ \mathbf{x}_i \ \mathbf{y}_i \ \mathbf{z}_i \}$, and refer to as the i-th *link coordinate system*, and the position vector of the origin of this coordinate system relative to the inertial or any other coordinate system. The i-th link coordinate system is rigidly attached to the i-th link, and so the i-th frame defines the configuration of the i-link relative to the inertial or any other reference system. The frame which is associated with link 0 is often referred to as the *base frame*. When the base frame is considered as an inertial reference frame (which is often the case) it serves as a universal frame relative to which everything we discuss can be referred. On the end-effector, we attach a frame to which we assign the number $n+1$ and which we call the *tool frame*. Note that the tool frame has a constant configuration relative to the n-th frame. The configuration of the tool frame, relative to an inertial frame, is usually considered as the configuration of the robot manipulator. This is justified, since the end-effector is that part of a robot which is designed to make contact with the environment for the purpose of executing some task. However, in the actual kinematic and dynamic analyses of a robot manipulator, we consider the configuration of its last link as the configuration of the robot manipulator. This is acceptable for two reasons : First, the end-effector is always attached rigidly to the last link, and thus has a constant configuration relative to that of the last link; and secondly, we want the analysis to be general and not specific to a particular end-effector. Moreover, to make the analysis independent of a particular environment where the robot may be used, we choose the inertial or reference coordinate system to be attached to the base of the robot. Thus, we usually choose the basis frame $\{ \mathbf{e}_0 \}$ to be the inertial or universal reference frame. Therefore, in this thesis, unless indicated otherwise, we shall take the configuration of a robot manipulator to mean the configuration of its last link relative to the base coordi-

nate frame $\{ e_0 \}$. Also, when a quantity such as a vector or a coordinate system is defined relative to the inertial reference frame, the term *absolute* will be used. When the reference frame is another link frame, we shall use the term *relative.*

2.3.2 Geometric Description of a Link

To obtain the proper kinematic and dynamic equations for a robot manipulator, it is important to know the exact geometric characteristic of each link. This will enable us to define the absolute or relative configuration of any link in the articulated portion of a robot manipulator.

A geometric description of a link is mainly concerned with what relationship exists between two neighboring joint axes of a manipulator. Mathematically, joint axes are defined by lines in the three dimensional space. Thus the joint axis i is defined to be the line or vector direction in space about which the i-th link rotates or translates relative to the $(i-1)$-th link. Note that this definition for the joint axes implies that link i $(i \neq n)$ connects two joint axes, the i-th and the $(i+1)$-th. In particular the joint axis i about or along which link i moves is called its *proximal joint*, and the joint axis $(i+1)$ which connects link i with link $(i+1)$ is the *distal joint* associated with link i.

The relative location (configuration) of two axes in the three dimensional space is defined by specifying the following two quantities :

Link length : For any two axes in the three dimensional space, there exists a well-defined distance between them. This distance is measured along a line which is mutually perpendicular to both axes. This distance always exists and is unique except when the two axes are parallel, in which case there are many mutually perpendicular lines of equal length.

As we have mentioned, the i-th link is associated with the i-th and the $(i+1)$-th joints. Therefore, the mutually perpendicular line between the i-th and the $(i+1)$-th joints allows us to define what is called the *length* of the i-th link.

Link twist : Between two axes in the three dimensional space, we can always define an angle. There are two cases to examine. In the first case, the axes are assumed to be parallel, and we consider that a zero angle exists between them. In the second case, we assume that the axes are not parallel and we define an angle between them as follows : We consider a plane normal to the mutually perpendicular line which exists between the two non-parallel axes. Now, by considering the projection of these two axes on the normal plane, we obtain two non-parallel lines on the plane. At the point of their intersection we can choose one of the four angles to be the *twist angle* of the two axes. Based on this angle, we shall define the twist angle of a link in the next section.

2.3.3 Description of Link Connections and the Configuration of a Robot Manipulator

The primary purpose of this section is to describe the "configuration" transformation, which defines the relative displacement of two neighboring links of a robot manipulator, as well as to derive a matrix representation for it.

A special description for the configuration transformation of two neighboring members of a spatial kinematic chain, has been established over the years and is known as the Denavit-Hartenberg (D-H) description or convention. The D-H convention has proven to be very practical in robotics because it allows for a systematic description of a spatial kinematic chain, in particular when it is of an open-loop structure. The D-H convention was first introduced by Denavit and Hartenberg [23] (in 1955) for the purpose of analyzing spatial linkages, and was specialized to open-loop spatial kinematic chain by Kahn [25] (in 1969). The usual D-H convention, as originally designed for kinematic analysis, has some disadvantages and can lead to ambiguities when it is used in robots with links having more than two joints [26]. The scheme which we describe here is well suited for open loop spatial kinematic chains and can be easily adapted for spatial kinematic chains with tree or closed-loop structures [26].

The D-H convention associates a frame rigidly with every link (or joint). In particular, the i-th frame is associated with the i-th link and its coordinate system defines the orientation of the i-th link (frame) relative to the $(i-1)$-th link (frame). Also its position vector defines the origin displacement of the i-th coordinate system relative to the $(i-1)$-th coordinate system. In the D-H convention, to assign the i-th coordinate frame on the i-th link, we use the following two basic assumptions :

i) The $\mathbf{z}_i$ basis unit vector of the i-th frame coordinate system is always parallel to either the proximal or the distal axis of the i-th link.

ii) The $\mathbf{x}_i$ basis unit vector of the i-th frame coordinate system is always parallel to the mutual perpendicular between the i-th and the $(i+1)$-th joint axes

Based on these assumptions we define the i-th link coordinate system. We assume that the $\mathbf{z}_i$ basis vector is parallel to the proximal joint of the link, i.e., it is parallel to the i-th joint axis. The origin o_i of the i-th frame coordinate system is located at the intersection of the i-th joint axis and the mutual perpendicular between the i-th and the $(i+1)$-th joint axes. When this point is not unique (in the case of parallel joint axes) we choose the one which minimizes the relative distance between the origin of the $(i-1)$ and i-th coordinate systems. The $\mathbf{x}_i$ basis unit vector is on the mutual perpendicular to the axis $\mathbf{z}_i$ and $\mathbf{z}_{i+1}$ directed from the former to the latter. The $\mathbf{y}_i$ basis unit vector of the i-th frame coordinate system is chosen as the unique perpendicular to both $\mathbf{z}_i$, and $\mathbf{x}_i$ at the point o_i, which defines a *right-hand* oriented coordinate system.

In the D-H convention four parameters are needed to specify completely the relative configuration of two neighboring frames. These parameters for the i-th frame are defined as follows :

α_i : The angle about $\mathbf{x}_{i-1}$, between $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$.

a_i : The distance along $\mathbf{x}_{i-1}$, between $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$.

d_i : The distance along $\mathbf{z}_i$, between $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$.

θ_i : The angle about $\mathbf{z}_i$, between $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$.

and are shown in the following Figure

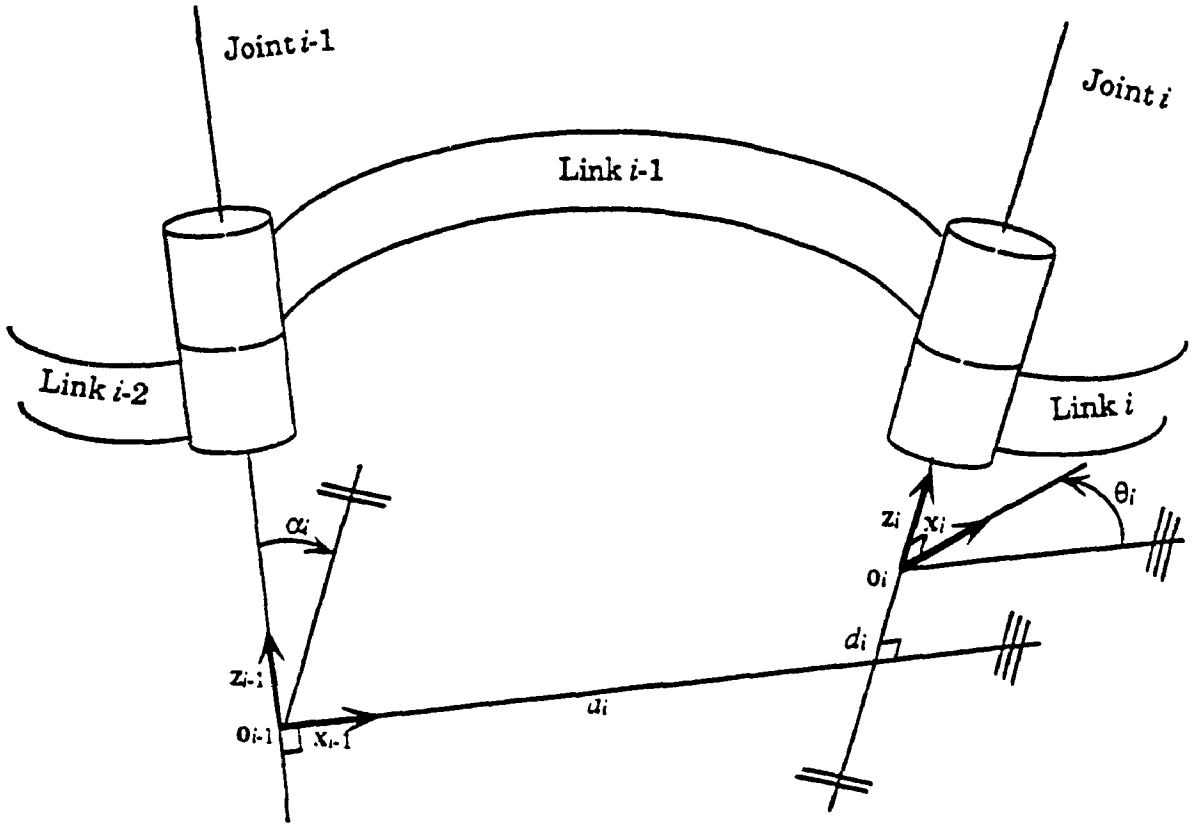

Figure 2.1 : Link Parameters and Link Coordinate Systems

Remark 2.3 : The angle α_i and the distance a_i, are associated with the $(i-1)$-th link, and can be used to define the twist angle and the length, respectively, of this link. Since these two quantities are associated with the $(i-1)$-th link, some authors [13] use the notation α_{i-1} and a_{i-1} instead of α_i and a_i. We prefer to use α_i and a_i, because, as we shall see later, it leads to a uniform notation for the matrix representation of the configuration transformation between two neighboring links.

Each of the parameters defined above, can be used to define a pure translation or a pure rotation displacement. We shall call these displacements *elementary displacements*, since they occur along or about a coordinate axis, and we denote them by *Trans* (axis , var) and *Rot* (axis , var) respectively, where "axis" is the coordinate axis of the displacement and "var" denotes the variable of the displacement.

Now, using elementary displacement operations, we can describe [13] the displacement of the i-th link relative to the $(i-1)$-th link, which we denote by $\mathbf{A}_i \equiv {}^{i-1}\mathbf{A}_i$, as follows

$$\mathbf{A}_i = Rot(\mathbf{x}_{i-1},\alpha_i)\,Trans(\mathbf{x}_{i-1},a_i)\,Rot(\mathbf{z}_i,\theta_i)\,Trans(\mathbf{z}_i,d_i) \tag{2.3.1}$$

Since the displacement $\mathbf{A}_i$ also defines the configuration of the i-th frame relative to the $(i-1)$-th frame, we refer to it as the i-th *homogeneous (configuration) coordinate transformation.* Using equations (2.2.7) and (2.2.8), it is easy to see that the matrix representation for the homogeneous transformation defined by (2.3.1), in terms of direction cosines, is given by

$$A_i = \left[\begin{array}{ccc|c} \cos\theta_i & -\sin\theta_i & 0 & a_i \\ \cos\alpha_i\sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i & -d_i\sin\alpha_i \\ \sin\alpha_i\sin\theta_i & \sin\alpha_i\cos\theta_i & \cos\alpha_i & d_i\cos\alpha_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.3.2}$$

and we refer to A_i as the i-th *homogeneous coordinate transformation matrix.*

It is clear that, since the i-th joint has one degree of freedom, three of the four parameters defined above will be constant and only one will be variable. The variable parameter of the i-th joint is known as the i-th *joint coordinate* and is usually denoted by q_i. From the definition of the four parameters, it is obvious, that $q_i \triangleq \theta_i$, when the i-th joint is revolute and $q_i \triangleq d_i$, when the i-th joint is prismatic. Therefore, with the appropriate definition for the joint variable q_i, the transformation $\mathbf{A}_i \triangleq \mathbf{A}_i(q_i)$, defined by (2.3.1), describes completely the one degree-of-freedom displacement (motion) of the i-th link relative to the $(i-1)$-th link.

As we mentioned in Section 2.2, a general displacement can be expressed as a pure rotation and a pure translation. Thus, by comparing equations (2.2.9) and (2.3.2), we can see that the rotation $\mathbf{A}_i$ of the i-th link coordinate system relative to the $(i-1)$-th, has a (3×3) matrix representation, in terms of direction cosines, which is given by

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \cos\alpha_i\sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i \\ \sin\alpha_i\sin\theta_i & \sin\alpha_i\cos\theta_i & \cos\alpha_i \end{bmatrix} \tag{2.3.3}$$

Similarly, the origin displacement or position vector of the i-th coordinate system relative to the $(i-1)$-th coordinate system, has a (3x1) matrix representation which is given by

$$s_{i-1,i}^{i-1} = \begin{bmatrix} a_i \\ -d_i\sin\alpha_i \\ d_i\cos\alpha_i \end{bmatrix} \tag{2.3.4}$$

Therefore, the homogeneous transformation $\mathbf{A}_i$, or the rotation $\mathbf{A}_i$ together with the translation $\mathbf{T}_{s_{i-1,i}}$, completely describes the configuration of the i-th link relative to the $(i-1)$-th link.

Now, we can use the relative configuration between two neighboring links to define the absolute configuration of any link of a robot manipulator. We do that here in terms of homogeneous transformations since this leads to a compact description.

It is well known [12] that, in general, if we post-multiply a homogeneous transformation representing a frame by a second homogeneous transformation describing a rotation and/or a translation, we make that rotation and/or t anslation with respect to the frame which is described by the first transformation. Thus, the homogeneous transformation

$$^{i-1}\mathbf{W}_{i+1} = \mathbf{A}_i\mathbf{A}_{i+1} \tag{2.3.5}$$

describes the configuration of the $(i+1)$-th frame (link) relative to the $(i-1)$-th frame. Therefore, the absolute configuration of the i-th link is described by the transformation,

$$
\begin{aligned}
{}^{0}\mathbf{W}_i \equiv \mathbf{W}_i &= \mathbf{A}_1\mathbf{A}_2 \cdots \mathbf{A}_i \\
&= \mathbf{W}_{i-1}\mathbf{A}_i \qquad\qquad i = 1, 2, \ldots, n
\end{aligned} \tag{2.3.6}
$$

and obviously the absolute configuration of the manipulator (i.e., the configuration of the n-th link) is given by $\mathbf{W}_n$.

As can be seen from equation (2.3.6), the homogeneous transformation $\mathbf{W}_n$ is a function of the joint coordinates q_i, $i = 1, 2, \ldots, n$. The joint coordinates q_i are linearly independent, and this allows us to view them as the components of an n dimensional vector $\mathbf{q}$, relative to some curvilinear coordinate system which describes the *joint space* of the robot manipulator. The vector $\mathbf{q}$ is referred to as the *joint space vector* and since $\mathbf{W}_n \triangleq \mathbf{W}_n(\mathbf{q})$, the $\mathbf{W}_n$ provides a *joint space description* for the configuration of a robot manipulator.

The joint space description for the configuration of a robot manipulator provided by $\mathbf{W}_n$ can be viewed as an "internal" description, since implicitly it contains the configuration of all the individual links of the manipulator. Now, from section 2.2, we recall that the configuration of a rigid body (and therefore that of the last link or the end-effector of a robot manipulator) can be described in Cartesian space in terms of a "Cartesian vector" χ†. The Cartesian space description for the configuration of a robot manipulator can be viewed as an "external" description, since it does not take into account the configuration of the individual links in the manipulator chain.

Both, the joint and Cartesian space descriptions for the configuration of a robot manipulator are fundamental in robotics. The Cartesian space description is useful, mainly to human operators, since it allows for an easy description of the motion of the end-effector between two different locations in space. However, the motion of a robot manipulator is realized in joint space and therefore a joint space description is necessary for the robot controller. Hence, some of the basic kinematic problems in robotics deal

† Note that sometimes in robotics, more general spaces (e.g. the *operational space* [27]) are used to describe the configuration of a robot manipulator.

with transformations between joint space and Cartesian space descriptions for the configuration of a robot manipulator.

Besides configuration kinematic analysis, motion kinematic analysis is also needed for the dynamic analysis of any mechanical system. However, before we deal with motion kinematics of robot manipulators we need first to develop a methodology which will allow us to study motion kinematics in a simple and efficient manner. Since this methodology will be based on Cartesian tensors, the next chapter is devoted to Cartesian tensor analysis.

2.4 REFERENCES

[1] H Goldstein, *Classical Mechanics , 2nd* ed. Reading, MA:, Addison Wesley, 1981.

[2] O. Bottema and B. Roth, *Theoretical Kinematics*, North-Holland Publishing Co., Amsterdam, 1978.

[3] J. L. Synge, "Classical Dynamics" *Encyclopedia of Physics*, S. Flugge Edit., Vol. III, Springer-Verlag, Berlin-Gottingen-Heidelberg, 1960.

[4] J. Angeles *Spatial Kinematics Chains : Analysis, Synthesis, Optimization*, Springel-Verlag, Berlin-Heidelberg-New York, 1982.

[5] J. Stuelpnagel "On the Parametrization of the Three-Dimensional Rotation Group" , *SIAM REVIEW*, pp. 422-430, Vol. 6, No. 4, October 1964.

[6] J. Rooney "A Survey of Representations of Spatial Rotations About a Fixed Point", *Environment and Planning B*, pp. 185-210, Vol. 4, 1977.

[7] D. Hestenes, "Vectors, Spinors, and Complex Numbers in Classical and Quantum Physics", *J. Math. Phys.* Vol. 39, pp 1013-1027, 1971.

[8] R. A. Wehage, "Quaternions and Euler Parameters - A Brief Exposition" *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, Edited by E. J. Haug, Springer-Verlag, Berlin-Heidelberg, 1984.

[9] J. Angeles *Rational Kinematics*, Springer-Verlag, Berlin-Heidelberg-New York, 1989.

[10] D. Lovelock and H. Rund, *Tensors, Differential Forms, and Variational Principles*, John Wiley & Sons, New York, 1965.

[11] L. Brand *Vector and Tensor Analysis*, John Wiley & Sons, New York, 1947.

[12] R. P. Paul, *Robot Manipulator : Mathematics, Programming and Control*, MIT Press, Cambridge, MA, 1981.

[13] J.J. Craig, *Introduction to Robotics : Mechanics & Control*, Reading, MA: Addison-Wesley, 1986.

[14] W. A. Wolovich *Robotics : Basic Analysis and Design*, Holt, Rinehart and Winston, New York, 1987.

[15] J.Y.S. Luh and Y.F. Zineng, "Computation of Input Generalized Forces for Robots with Closed Kinematic Chain Mechanisms", *IEEE J. Robotics and Automation*, RA-1, No 2, pp 95-103 , 1985.

[16] T.R. Kane and H. Faessler, "Dynamics of Robots and Manipulators Involving Closed Loops" , *Theory and Practice of Robots and Manipulators* , A. Morecki, G. Bianchi and K. Kedzior, *Eds.*, MIT press, Cambridge, MA, 1985.

[17] Y. Nakamura and M. Ghodoussi, "A Computational Scheme of Closed Link Robot Dynamics Derived by D'Alembert's Principle", *Proc. IEEE Int. conf. on Robotics and Automation*, pp. 1353-1360, Philadelphia PA, April 24-29, 1988.

[18] W.J. Book, "Recursive Lagrangian Dynamics of Flexible Manipulators", *Int J. Robotics Research*, vol. 3, no. 3, Fall 1984.

[19] M. Geradin, G. Robert and Bernardin, "Dynamic Modeling of Manipulators with Flexible Members" , *Advanced Software in Robotics* , A. Danthine and M. Geradin, *Eds.*, Elsevier Science Pub. Co., 1984.

[20] H. Kanoh, S. Tzafestas, H.G. Lee and J. Kalat, "Modeling and Control of Flexible Robot Arms", *Proc. of 25-th CDC*, Athens, Greece, Dec. 1986.

[21] G. B. Yang and M. Donath, "Dynamic Model of a One-Link Robot Manipulator with Both Structural and Joint Flexibility", *Proc. IEEE Int. conf. on Robotics and Automation*, pp. 476-481, Philadelphia PA, April 24-29, 1988.

[22] A. Maciejewski and C. A. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *Int. J. of Robotics Research*, Vol. 4, No. 3, pp 109-117, 1985.

[23] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *ASME J. of Appl. Mechanics*, Vol. 23, pp. 215-221, 1955.

[24] J. J. Uicker, *On the Dynamic Analysis of Spatial Linkages Using* 4×4 *Matrices*, Ph.D. Dissertation, Northwestern Univ. Aug. 1965.

[25] M. E. Kahn, "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains", Stanford Artificial Intelligence Project, *memo. AIM-106*, Dec. 1969.

[26] W. Khalil and J. F. Kleinfinger, "A New Geometric Notation for Open and Closed-Loop Robots", *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, pp. 1174-1179, 1986.

[27] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators : The Operational Space Formulation", *IEEE J. Robotics and Automation*, Vol. RA-3, No. 1, pp. 43-53, 1987.

CHAPTER III

CARTESIAN TENSOR ANALYSIS

3.1 INTRODUCTION

As we mentioned in Chapter I, the representations of various physical quantities, which are involved in the formulation of the equations of motion of a dynamic system, effects the computational efficiency of these equations. We also pointed out there that a Cartesian tensor representation for the angular velocity indicates that Cartesian tensors can be used to describe the dynamic equations of rigid body motion. Cartesian tensor analysis, being more general than vector analysis, is powerful and, if properly used, can result in a tensor formulation for the equations of general motion of a dynamic system, which may lead us in computationally efficient algorithms. That this is indeed the case for the dynamic equations of motion of rigid-link open-chain robot manipulators, will be demonstrated in Chapter V. In this chapter we provide an introduction to the theory of *Cartesian tensors*. This theory is extended here by proving a number of propositions which allow for easy algebraic manipulations of the equations of motion of a complex dynamic system such as those of a robotic system.

Historically, the ideas and symbolism of tensor calculus originated in differential geometry, and was invented by the Italian mathematicians Ricci and Levi-Civita [1]. Gradual introduction and assimilation of these ideas and symbols was greatly accelerated by their use by Einstein in his general theory of relativity; and today tensor analysis forms a well established field which provides the only appropriate language for studying differential geometry [2,4] and related topics such as the theory of general relativity.

But if tensor calculus is a necessity for studying differential geometry, for applications in classical mechanics [5-10] it is a great convenience, because it enables one to

express geometrical or physical relationships of tensor entities in a concise manner which does not depend on the introduction of a coordinate system. Moreover, even in cases where we have to introduce coordinate systems, because measurements are required or for other reasons, tensor equations are *formally* the same in *all* admissible coordinate systems. This fact will help us later when tensor equations have to be written in the various coordinate systems used to derive dynamic models for robot manipulators.

In the general theory of tensor analysis, the space or environment where a tensor is defined is a *manifold* [4] and it is characterized in terms of curvilinear coordinate systems. But, since the environment or physical space for physical systems is a Euclidean space of three dimensions we are restricting our attention to the study of tensors in Euclidean spaces. In a Euclidean space, orthogonal Cartesian coordinate systems are sufficient for tensor analysis. Hence, we shall exclusively use *orthogonal Cartesian coordinate systems* in our analysis. Tensors analyzed in an orthogonal Cartesian coordinate system are referred to as *orthogonal Cartesian tensors* [14-18]. Therefore, unless indicated otherwise, when we write Cartesian tensors, or simply tensors, we shall mean orthogonal Cartesian tensors The use of orthogonal Cartesian systems will simplify our analysis, since the distinction between *covariant* and *contravariant* components, which is necessary in *curvilinear* coordinate systems, disappears in orthogonal Cartesian coordinates. Also terms arising from *curvature* are zero in the theory of Cartesian tensors since the Euclidean space is a *flat space*. Finally, another important characteristic of Cartesian tensors is that when they are smooth functions of time, their time derivatives also form again a tensor of the same order.

The outline of this chapter is as follows : In Section 3.2 second order tensors are defined and some basic algebraic tensor operations are introduced. Section 3.3 outlines the structural symmetries of second order tensors based on *Cartesian* and *Spectral decompositions*. Also, *dual* operators between vectors and skew-symmetric tensors are defined, and a geometric characterization for skew-symmetric tensors is given. Finally,

Section 3.4 contains important propositions which establish basic tensor identities.

3.2 SECOND ORDER CARTESIAN TENSORS

In this section, we define orthogonal Cartesian tensors of order 2. We also introduce basic tensor algebraic operations which will be used later in algebraic manipulations.

3.2.1 On The Definition of the Second Order Cartesian Tensors.

As we mentioned in Section 3.1, we restrict our attention to tensor analysis in Euclidean space. We consider here three dimensional (3-D) Euclidean spaces, but the extension of this theory to Euclidean spaces of higher dimensions is straightforward.

In general, tensor analysis is concerned with mathematical or physical entities known as tensors which although of different nature have common characteristics and properties. These common properties, allow us to classify them into common classes and refer to them with such names as *scalars, vectors* or in general *tensors* of certain order.

There are two basic approaches which one can take to define a tensor - or the same, to define the class where a particular physical entity belongs.

In the first approach we analyze a physical quantity from a "*quantitative*" point of view. We define a physical quantity to be a tensor of a certain order if it has the *necessary* and *sufficient* properties which all the members of that class must have in common. We can term these necessary and sufficient properties as *intrinsic* properties, since they are independent of each other as well as of the environment, i.e., they characterize a physical quantity independently of any observer. Definitions in terms of necessary and sufficient properties are in general difficult to state, since one needs to know a great deal about any entity before one is in a position to define it using only its intrinsic properties. The difficulties associated with definitions based on this approach are obvious from the following discussion.

Physical quantities termed "vectors" in textbooks on classical mechanics are defined [13] in the 3-D linear space as follows.

Definition 3.1 : We use the term *vectors* for those quantities which for their complete specification require the following necessary and sufficient intrinsic properties : *magnitude, orientation* (or *sense*) and *direction.*

This definition which has been accepted and used universally for years, fails to distinguish between the two kind of vector quantities, known as *polar* and *axial*† vectors. We are able to distinguish (and eventually use) polar and axial vectors, but we do that only in relation with the environment, i.e., only when we test them under space inversion, which obviously are not part of the given definition. It seems that, although we classify the direction as an intrinsic property of a vector, we do not fully "understand" it. We need the environment to characterize the direction and thus, we do not treat it any more as an intrinsic property. As a consequence the definition of a vector fails.

Recently, Hestenes [25] came up with a proper understanding of the direction and therefore with a proper algebraic definition of a vector. Hestenes gives a new interpretation and meaning to the word direction. According to him, direction contains the geometrical notion of *dimension* or *grade.* Hestenes defines as a *1-vector* or a vector with *direction of grade 1* what was commonly understood as a polar vector. To define an axial vector he introduced the idea of *direction of grade 2,* as an intrinsic property of areas (since axial vectors are used to represent area). An axial vector is then defined as a *2-vector* or a *bivector,* i.e., a vector with direction of grade 2. Therefore, with this extended meaning of the word direction we are able to define vectors based only on their intrinsic properties.

In the second and almost universal approach we analyze a physical quantity from

† Polar vectors are those vectors which under space inversion change their orientation (sense), and axial vectors are those which keep their orientation under space inversion. Usually, axial vectors are assumed to result from a vector cross product operation

an *"operational"* point of view and define the tensor character of physical quantities based on their *external* or *observed* properties. These properties are determined by considering how a physical quantity is related to some environment and how this relationship changes under controlled changes of this environment. This approach is, in general, easier and overcomes difficulties associated with the first one. There are two ways to proceed in this approach.

We can introduce first a frame of reference, i.e., a coordinate system, which describes the environment and expresses (by considering a set of ordered numbers or functions, known as components) the quantity at hand relative to that coordinate system. Then we make a change (by considering a map) in the coordinate system and analyze how the components of this quantity, relative to the old and new coordinate system, are related. If the changes in the components follow a certain law we define the quantity under consideration to be a tensor. This is a *passive* approach, since the tensor quantities are actually unaware of the coordinate systems used to describe the environment and represent them. Under a change of coordinate systems it is their components that change, not they themselves. Based on what we have said, we can give the following definition.

Definition 3.2 : Suppose that the abstract object $\mathbf{T}$ is represented in an orthogonal Cartesian coordinate system $\{ \mathbf{e} \}$, of a 3-D Euclidean space, by the set of components t_{ij}, where the subscripts i, j are ordered and can take the values 1,2,3. Let t'_{ij} be the corresponding set of components representing $\mathbf{T}$ when it is referred to another orthogonal Cartesian coordinate system $\{ \mathbf{e}' \}$, which is related to $\{ \mathbf{e} \}$ by an orthogonal transformation $\mathbf{A}$, i.e., $\{ \mathbf{e}' \}$ satisfies the equation

$$[\mathbf{e}'_l]_j = a_{ji} [\mathbf{e}_l]_i \qquad l = 1,2,3 \qquad (3.2.1)$$

where a_{ji} are the entries of the coordinate matrix A which represents the transformation $\mathbf{A}$ relative to the two coordinate systems $\{ \mathbf{e}' \}$ and $\{ \mathbf{e} \}$. Then if the equation

$$t'_{rs} = a_{ri}\, a_{sj}\, t_{ij} \tag{3.2.2}$$

is valid, we say that $\mathbf{T}$ is a second order Cartesian tensor.

Remark 3.1 : In Cartesian tensor analysis the nomenclatures "order" or "rank" are used [17] to denote the exponent (r) to which the dimension (n) of the space, on which the tensor is defined, must be raised to give the number of components (coefficients) of a tensor. To put it in another way, the order of a tensor denotes how many copies of the original Euclidean space we need to consider for producing the environment (space) where the tensor is defined. In the case of the second order tensor $\mathbf{T}$ defined on a 3-dimensional Euclidean space, we must have $3^2 = 9$ components and this is the case as (3.2.2) indicates. We shall continue to use the word *order* with this meaning. Therefore, according to what we have said, it is obvious, that order is not an intrinsic property of a tensor. It depends on the space where the tensor is defined. However, with the word *rank*, we shall associate an intrinsic property of a tensor, which we shall define later.

Finally, another way to proceed and define the tensor character of a physical quantity is to consider the *action* of this quantity on its environment, i.e., we treat, in this approach, the physical quantity as an *operator*. If the operator has certain properties, then we are able to say that the quantity under consideration is a tensor. From this point of view we can give the following definition [11] :

Definition 3.3 : We say that the *linear vector transformation* $\mathbf{T}$ is a second order tensor if we can compute the action of $\mathbf{T}$ on any vector $\mathbf{r}$ and denote that action of $\mathbf{T}$ on $\mathbf{r}$ by writing $\mathbf{T}(\mathbf{r})$ or $\mathbf{T} \cdot \mathbf{r}$ or simple $\mathbf{T}\,\mathbf{r}$.

Definition 3.3 is not complete as stated, since we have not mentioned explicitly the domain and range of the linear vector transformation. The domain and range of a second order tensor are obvious from the context. For example, in the equation $\mathbf{L}_C = \mathbf{I}_C \omega$ the domain of the inertia tensor ($\mathbf{I}_C$) is the space of angular velocities (ω) and its range is the space of angular momentum ($\mathbf{L}_C$).

Remark 3.2 : Although the terms "linear transformation" and "tensor" in Definition 3.3 refer to mathematical functions of the same kind they are not completely synonymous, because they have different connotations in applications. The term "tensor" is always used when describing certain physical quantities. Thus, we often say a rotation tensor or a rotation transformation, but we never call the *inertia tensor*, $\mathbf{I}_c$, "inertia linear transformation", although it defines the linear transformation $\mathbf{L}_c = \mathbf{I}_c\omega$.

As an example, of how we can use Definition 3.3 to define a tensor, let us consider the following vector equation,

$$(\mathbf{u}\otimes\mathbf{v})\mathbf{r} = \mathbf{u}(\mathbf{v}\cdot\mathbf{r}). \tag{3.2.3}$$

It is clear that $(\mathbf{u}\otimes\mathbf{v})\mathbf{r}$ is linear in $\mathbf{r}$ since

$$\mathbf{u}[\mathbf{v}\cdot(\lambda_1\cdot + \lambda_2\mathbf{r}_2)] = \lambda_1\mathbf{u}(\mathbf{v}\cdot\mathbf{r}_1) + \lambda_2\mathbf{u}(\mathbf{v}\cdot\mathbf{r}_2) \tag{3.2.4}$$

Therefore, the quantity $\mathbf{T} = \mathbf{u}\otimes\mathbf{v}$ is said to be a second order tensor. Since the tensor $\mathbf{u}\otimes\mathbf{v}$ is defined by using the two vectors $\mathbf{u}$ and $\mathbf{v}$ it is called the *tensor product* of $\mathbf{u}$ and $\mathbf{v}$.

Remark 3.3 : Many authors prefer to write equation (3.2.3) as

$$(\mathbf{u}\,\mathbf{v})\cdot\mathbf{r} = \mathbf{u}(\mathbf{v}\cdot\mathbf{r}). \tag{3.2.5}$$

In this case the tensor $\mathbf{u}\,\mathbf{v}$ is called the *dyad* or *outer product* of $\mathbf{u}$ and $\mathbf{v}$. The first vector in a dyad is called the *antecedent* and the second vector the *consequent*. According to Gibbs [19], who developed the theory of dyadics (a dyadic is a sum of dyads), a dyad or indeterminate product is a purely symbolic quantity which requires a determinate physical meaning only when used as a linear operator. Therefore, by definition (3.3), dyads are second order tensors. Therefore, to simplify the notation we shall identify the tensor product of two vectors $\mathbf{u}$ and $\mathbf{v}$ with their dyad and we shall write

$$\mathbf{u}\otimes\mathbf{v} \equiv \mathbf{u}\,\mathbf{v} \tag{3.2.6}$$

As is well known [19], in a 3-D Euclidean space, any linear vector transformation can be written as the sum of dyads of which either the antecedents or the consequents, but not both, may be arbitrarily chosen provided they are linearly independent. This implies that to each tensor we can associate k ($k \leq 3$) *linearly independent directions.* Or, to use the approach of Hestenes' [25], to each second order tensor in a 3-D Euclidean space we can associate a direction of k dimensions ($k \leq 3$). Based on this, we can give the following definition for the term "rank".

Definition 3.4 : We define the *rank* of a tensor to be the number of linearly independent directions which a tensor possesses.

The definition of the rank as given here, coincides with what we call rank in linear algebra. In particular, the rank of a tensor (or a linear transformation) $\mathbf{T}$ coincides with the rank of the matrix T, which represents $\mathbf{T}$ in some coordinate system. This uniformity in the meaning of the term "rank" is not possible if we identify rank with order as is often done in textbooks on tensor analysis.

Remark 3.4 : Although in the general theory of tensor analysis scalars and vectors are treated as tensors of order *zero* and *one* respectively, we continue to refer to them as scalars and vectors and reserve the word "tensor" for tensors of second and higher order.

3.2.2 The Linear Space Structure for the Second Order Cartesian Tensors

In order to use tensors in an efficient manner we have to define algebraic structures on them by introducing basic algebraic operations. Thus, based on Definition 3.3, we can see that the following tensors and algebraic operations are well defined.

The *zero* and *unity (identity)* tensors are denoted and defined, respectively, by

$$0\mathbf{v} = \mathbf{0} \quad , \quad \forall\, \mathbf{v} \tag{3.2.7}$$

$$1\mathbf{v} = \mathbf{v} \quad , \quad \forall\, \mathbf{v} \tag{3.2.8}$$

The algebraic operations of *addition* and *scalar multiplication* can be defined as usual.

Thus the addition of two second order tensors $\mathbf{T}$ and $\mathbf{S}$ is defined by

$$(\mathbf{T}+\mathbf{S})\mathbf{v} = \mathbf{T}\mathbf{v} + \mathbf{S}\mathbf{v} \quad , \ \forall\, \mathbf{v} \tag{3.2.9}$$

and the multiplication of $\mathbf{T}$ by a scalar λ is defined by

$$(\lambda\mathbf{T})\mathbf{v} = \lambda(\mathbf{T}\mathbf{v}) \quad , \ \forall\, \lambda \tag{3.2.10}$$

Also, we say that two second order tensors $\mathbf{T}$ and $\mathbf{S}$ are equal if

$$\mathbf{T} = \mathbf{S} \iff \mathbf{T}\mathbf{v} = \mathbf{S}\mathbf{v} \qquad \forall\ \mathbf{v} \tag{3.2.11}$$

or equivalently

$$\mathbf{T} = \mathbf{S} \iff \mathbf{v}\cdot\mathbf{T}\mathbf{u} = \mathbf{v}\cdot\mathbf{S}\mathbf{u} \qquad \forall\ \mathbf{u}, \mathbf{v} \tag{3.2.12}$$

It is now easy to see that the set of all tensors of the second order, together with the two algebraic operations of addition and scalar multiplication, constitutes a vector space over a scalar field, which is assumed here to be the field of real numbers. To find a basis of the vector space for the second order tensors, defined on a 3-D Euclidean space and the components of a tensor $\mathbf{T}$ relative to it, let us consider an orthogonal basis $\{\mathbf{e}\} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of a 3-D Euclidean space. Then for any vector $\mathbf{r}$ we can write

$$\begin{aligned}\mathbf{T}\mathbf{r} &= \mathbf{T}(r_1\mathbf{e}_1 + r_2\mathbf{e}_2 + r_3\mathbf{e}_3)\\ &= r_1\mathbf{T}\mathbf{e}_1 + r_2\mathbf{T}\mathbf{e}_2 + r_3\mathbf{T}\mathbf{e}_3, \quad \text{[by the linearity of } \mathbf{T}\text{]}.\end{aligned} \tag{3.2.13}$$

But $\mathbf{T}\mathbf{e}_1, \mathbf{T}\mathbf{e}_2$ and $\mathbf{T}\mathbf{e}_3$ are vectors and therefore may be expressed in terms of the Cartesian components as follows :

$$\mathbf{T}\mathbf{e}_1 = T_{11}\mathbf{e}_1 + T_{21}\mathbf{e}_2 + T_{31}\mathbf{e}_3 \tag{3.2.14}$$

$$\mathbf{T}\mathbf{e}_2 = T_{12}\mathbf{e}_1 + T_{22}\mathbf{e}_2 + T_{32}\mathbf{e}_3 \tag{3.2.15}$$

$$\mathbf{T}\mathbf{e}_3 = T_{13}\mathbf{e}_1 + T_{23}\mathbf{e}_2 + T_{33}\mathbf{e}_3 \tag{3.2.16}$$

where, the coefficients $T_{11}, T_{21}, \cdots, T_{33}$ can be computed by

$$\begin{aligned} T_{11} &= \mathbf{e}_1\cdot\mathbf{T}\mathbf{e}_1\\ T_{21} &= \mathbf{e}_2\cdot\mathbf{T}\mathbf{e}_1\\ &\cdot \quad = \quad \cdot\\ &\cdot \quad = \quad \cdot\\ &\cdot \quad = \quad \cdot\\ T_{33} &= \mathbf{e}_3\cdot\mathbf{T}\mathbf{e}_3 \end{aligned} \tag{3.2.17}$$

Now, using (3.2.14)-(3.2.16) in (3.2.13) and the equations

$$\begin{aligned} r_i \mathbf{e}_i &= \mathbf{e}_i(\mathbf{e}_i \cdot \mathbf{r}) \\ &= \mathbf{e}_i \mathbf{e}_i(\mathbf{r}), \end{aligned}$$

for $i = 1, 2, 3$, we get after a few manipulations

$$\mathbf{T}\,\mathbf{r} = (T_{11}\mathbf{e}_1\mathbf{e}_1 + T_{12}\mathbf{e}_1\mathbf{e}_2 + \cdots + T_{33}\mathbf{e}_3\mathbf{e}_3)\mathbf{r}\,. \tag{3.2.18}$$

Therefore, since the vector $\mathbf{r}$ is arbitrary, we have from (3.2.18) that

$$\begin{aligned} \mathbf{T} = {} & T_{11}\mathbf{e}_1\mathbf{e}_1 + T_{12}\mathbf{e}_1\mathbf{e}_2 + T_{13}\mathbf{e}_1\mathbf{e}_3 \\ & + T_{21}\mathbf{e}_2\mathbf{e}_1 + T_{22}\mathbf{e}_2\mathbf{e}_2 + T_{23}\mathbf{e}_2\mathbf{e}_3 \\ & + T_{31}\mathbf{e}_3\mathbf{e}_1 + T_{32}\mathbf{e}_3\mathbf{e}_2 + T_{33}\mathbf{e}_3\mathbf{e}_3. \end{aligned} \tag{3.2.19}$$

or, if we use Einstein's† notation for the summation, we can write

$$\mathbf{T} = T_{ij}\mathbf{e}_i\mathbf{e}_j\,, \qquad i, j = 1, 2, 3. \tag{3.2.20}$$

Now, it can be shown [11], that this representation of $\mathbf{T}$ is unique and that the set of second order tensors $\{\mathbf{e}_1\mathbf{e}_1, \mathbf{e}_1\mathbf{e}_2, \cdots, \mathbf{e}_3\mathbf{e}_3\}$ form a basis for the aforementioned tensor vector space. Relative to this basis the components of $\mathbf{T}$ are the coefficients $T_{11}, \cdots T_{33}$ defined by (3.2.17). These coefficients can be put in a matrix form as

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \tag{3.2.21}$$

and we call T the coordinate matrix of $\mathbf{T}$ relative to the basis defined by the set $\{\mathbf{e}_1\mathbf{e}_1, \mathbf{e}_1\mathbf{e}_2, \cdots, \mathbf{e}_3\mathbf{e}_3\}$. Now, it is easy to see that there is a one-to-one correspondence between the orthogonal basis $\{\mathbf{e}\} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and the basis of the tensor linear space which is defined by the set $\{\mathbf{e}_1\mathbf{e}_1, \mathbf{e}_1\mathbf{e}_2, \cdots, \mathbf{e}_3\mathbf{e}_3\}$. Therefore, by slightly abusing the notation, we shall refer to (3.2.21) as the coordinate matrix of the tensor $\mathbf{T}$ relative to the Cartesian basis $\{\mathbf{e}\} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. With this in mind, we shall say that a second

† In Einstein's notation any expression in which two or more indices $i, j \cdots$ are each repeated is to be interpreted as the sum of all the values which it can take, as $i, j, \cdots$ take the values 1, 2, 3.

order tensor is defined in a 3-D Euclidean space instead of the 9-D linear space spanned by the basis $\{\mathbf{e}_1\mathbf{e}_1, \mathbf{e}_1\mathbf{e}_2, \cdots, \mathbf{e}_3\mathbf{e}_3\}$.

As an example of the coordinate matrix of a tensor, let us find the coordinate matrix of the dyad $\mathbf{u}\,\mathbf{v}$. By using equation (3.2.17), we get for $i, j = 1, 2, 3$

$$\begin{aligned}[\mathbf{u}\,\mathbf{v}]_{ij} &= \mathbf{e}_i \cdot (\mathbf{u}\,\mathbf{v} \cdot \mathbf{e}_j) \\ &= \mathbf{e}_i \cdot (\mathbf{u}\, v_j) \\ &= u_i\, v_j .\end{aligned}$$

Therefore,

$$[\mathbf{u}\,\mathbf{v}] = uv^T \tag{3.2.22}$$

i.e., the coordinate matrix $[\mathbf{u}\,\mathbf{v}]$ of the dyad $\mathbf{u}\,\mathbf{v}$ is given by the usual outer product of the coordinate matrices of the two vectors $\mathbf{u}$ and $\mathbf{v}$.

3.2.3 More Algebraic Operations.

Besides the addition (or subtraction) and scalar multiplication defined above, there are two other algebraic operations which one can define on tensors, namely, the tensor product and contraction. These operations are defined [5,15] as follows :

Tensor product : Let $\mathbf{T}$ and $\mathbf{S}$ be two second order tensors whose components, referred to a coordinate system $\{\mathbf{e}\}$, are t_{ij} and s_{kl}. Then it can be shown [15] that the 3^4 scalars

$$u_{ijkl} = t_{ij}\, s_{kl} \tag{3.2.23}$$

form the components of a tensor $\mathbf{U}$, say, of order 4. We call $\mathbf{U}$ the tensor product of $\mathbf{T}$ and $\mathbf{S}$ and we write $\mathbf{U} = \mathbf{T} \otimes \mathbf{S}$.

Contraction : Given a tensor of order $r \geq 2$, we may select a pair of indices and replace them by two identical indices. This action by virtue of Einstein's convention implies summation over the possible values of the identical indices. This process is known as contraction and the quantities obtained by contraction constitute the components of a tensor of order $r - 2$.

Note : In the special case where $r = 2$, the contraction operator is synonymous with the familiar *trace* operator since, as we can see, the contraction of the two indices produces a tensor of order zero, i.e., a scalar. Note, also, that the algebraic operations of tensor product and contraction can be performed on any tensor not necessarily Cartesian. Now, some other important algebraic operations, applicable to second order Cartesian tensors, are defined [16-17] as follows.

Transpose : Let T be the coordinate matrix of a second order tensor $\mathbf{T}$ relative to an orthogonal Cartesian coordinate system. Then the familiar matrix transpose T^T defines a new second order tensor, which we denote by $\mathbf{T}^T$ and call the *transpose* of $\mathbf{T}$.

The left and right dot product : Let $\mathbf{T}$ be a second order tensor and $\mathbf{v}$ be a vector, with coordinate matrices T and v, respectively, relative to the same coordinate system. Then the equation

$$u_i = T_{ij}\, v_j \tag{3.2.24}$$

which is computed by considering first the tensor product of $\mathbf{T}$ and $\mathbf{v}$ and then contracting the second index for the components of $\mathbf{T}$ and the index for the components of $\mathbf{v}$ defines the *right dot product* or *post-multiplication* of $\mathbf{T}$ and $\mathbf{v}$. Similarly using the equation :

$$w_i = v_j\, T_{ji} \tag{3.2.25}$$

we can define the vector $\mathbf{w}$. Equation (3.2.25) defines the *left dot product* or *pre-multiplication* of $\mathbf{v}$ and $\mathbf{T}$. In tensor form, equations (3.2.24) and (3.2.25) are written as

$$\mathbf{u} = \mathbf{T} \cdot \mathbf{v} \equiv \mathbf{T}\,\mathbf{v} \tag{3.2.26}$$

$$\mathbf{w} = \mathbf{v} \cdot \mathbf{T} \equiv \mathbf{v}\,\mathbf{T} \tag{3.2.27}$$

respectively. We can use the transpose operation to reorder the factors in the dot products defined above. Thus, for example, we can write

$$\mathbf{w} = \mathbf{v} \cdot \mathbf{T} = \mathbf{T}^T \cdot \mathbf{v} \tag{3.2.28}$$

In terms of their coordinate matrices, equations (3.2.26) and (3.2.27) are written as

$$u = Tv \tag{3.2.29}$$

and

$$w = v^T T \tag{3.2.30}$$

and they define the familiar post- and pre- multiplications in matrix theory.

Finally, we can define two other products between two second order tensors $\mathbf{T}$ and $\mathbf{S}$ as follows :

The dot product : Let T and S be the coordinate matrices, relative to the same coordinate system, of second order tensors $\mathbf{T}$ and $\mathbf{S}$, respectively. Then the equation

$$U_{ij} = T_{il} S_{lj} \tag{3.2.31}$$

defines the components of the second order tensor $\mathbf{U}$ which we call the *dot product* or *multiplication* of $\mathbf{T}$ and $\mathbf{S}$, and we write

$$\mathbf{U} = \mathbf{T} \cdot \mathbf{S} \equiv \mathbf{T}\,\mathbf{S} \tag{3.2.32}$$

In terms of their coordinate matrices equation (3.2.32) is written as

$$U = TS \tag{3.2.33}$$

and agrees with the usual matrix multiplication.

The double dot or inner product : The double dot product of two tensors $\mathbf{T}$ and $\mathbf{S}$ is given by

$$\mathbf{T} : \mathbf{S} = tr\,(\mathbf{T} \cdot \mathbf{S}) \tag{3.2.34}$$

where *tr* denotes the trace operator and produces a scalar.

Remark 3.5 : Strictly speaking some operations introduced above are defined by using coordinate matrix representations of second order tensors and vectors. However, once these operations have been established, the actual coordinate matrices used to represent the tensors or vectors are of no consequence and we can speak of these operations as

being defined on the tensors themselves without any ambiguity.

The double dot product is a generalization of the familiar vector dot product, and this allows us to define a norm for a tensor and the "angle" between two tensors. In particular, we define the *Frobenius norm* of a tensor $\mathbf{T}$ as following

$$\| \mathbf{T} \|_F \triangleq \sqrt{\mathbf{T} : \mathbf{T}^T}. \tag{3.2.35}$$

This called the Frobenius norm since, when we use a coordinate matrix T of a tensor $\mathbf{T}$ to evaluate equation (3.2.35), we end up with the familiar Frobenius matrix norm [31]

$$\| T \|_F \triangleq \sqrt{tr(TT^T)}. \tag{3.2.36}$$

Equation (3.2.36) follows from equation (3.2.35), by using equations (3.2.34) and (3.2.33). Now, using the double dot product and the Frobenius norm, we can define the cosine of the "angle" θ between two non-zero tensors, $\mathbf{T}$ and $\mathbf{S}$ as follows

$$\cos(\theta) = \frac{\mathbf{T} : \mathbf{S}^T}{\| \mathbf{T} \|_F \| \mathbf{S} \|_F}. \tag{3.2.37}$$

Equation (3.2.37) is a generalization of the following familiar definition for the cosine of the angle θ between two non-zero vectors $\mathbf{t}$ and $\mathbf{s}$

$$\cos(\theta) = \frac{\mathbf{t} \cdot \mathbf{s}}{\| \mathbf{t} \| \| \mathbf{s} \|}. \tag{3.2.38}$$

Actually, as we shall see in Section 3.3.3, equation (3.2.37) is essentially identical to equation (3.2.38) when it is applied to skew-symmetric tensors. Furthermore, based on this definition for the angle between two tensors, we shall say that two tensors $\mathbf{T}$ and $\mathbf{S}$ are *orthogonal* if and only if their double dot product is zero.

As we have mentioned above, the double dot product (and not the dot product) between two Cartesian tensors is the generalization of the familiar dot product between two vectors. The dot product between two second order tensors is not commutative as is the case with the familiar dot product between vectors, but it is associative and distributive over addition. This allows us to view the vector space of second order Cartesian

tensors, when supplied with this dot product, as a *linear algebra.* (This also follows from Definition 3.3 which identifies second order tensors with linear transformations). This algebra is isomorphic to the matrix algebra. Thus we can use the well-established matrix algebra to carry out calculations for various operations with tensors. Since the matrix elements are scalars, matrix algebra has the advantage of reducing all such calculations to addition and multiplication of real numbers. However, it has the disadvantage of requiring that a basis be introduced (which defines an isomorphism between tensors and matrices) and which may be quite irrelevant to the problem at hand and this often obscures the physical or geometrical meaning of the tensor involved. Moreover, this matrix representation may lead us to perform irrelevant and unnecessary calculations.

In the next sections, we analyze some basic properties of second order Cartesian tensors which will allow us to perform algebraic manipulations with tensors without resorting to their matrix representations.

3.3 Properties of the Second Order Cartesian Tensors

In this section we analyze the structural symmetries of second order Cartesian tensors by considering their *Cartesian* and *Spectral decompositions.* Also, we define their *scalar* and *vector invariants.* Moreover, for second order *skew symmetric* or *pseudo* tensors, defined on a 3-D Euclidean space, very important dual correspondences between them and vectors of the same 3-D Euclidean space are defined and a geometric characterization is given.

3.3.1 Isotropic Cartesian Tensors.

As we mentioned in section 3.2.1, tensors themselves are independent of coordinate systems; but the numerical values for the components of a tensor, in general, depend on these coordinate systems. Therefore, if we use an orthogonal transformation $\mathbf{A}$ to change the basis from $\{\mathbf{e}\}$ to $\{\mathbf{e}'\}$, i.e., if

$$\mathbf{e}'_j = \mathbf{A}\,\mathbf{e}_i \tag{3.3.1}$$

then the coordinate matrices T and T' of a tensor $\mathbf{T}$, relative to the old and new Cartesian bases respectively, are different. By Definition 3.2, they are related and this relationship is given by

$$T' = ATA^T \tag{3.3.2}$$

A tensor which has the *same* coordinate matrix in *all* Cartesian coordinate systems, or that is invariant under rotations is called an *isotropic* tensor. An example of an isotropic tensor is the Kronecker or *delta* tensor δ, with components

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{3.3.3}$$

relative to any orthogonal Cartesian coordinate system. Actually it can be shown [15] that the Kronecker tensor is the only (apart from a scalar multiple) isotropic tensor of order 2. We shall use the symbol $\mathbf{1}$ for the Kronecker tensor δ, since this tensor in our analysis stands for the *unit* tensor.

Although we are concerned with second order tensors, we mention here a 3rd-order tensor which is also isotropic and which we shall use on some occasions. This is the *Levi-Civita* or *alternating* tensor ϵ. The components of ϵ in any basis are defined by

$$\epsilon_{ijk} = \begin{cases} +1 & \text{if } (i,j,k) \text{ is an even permutation of } (1,2,3) \\ -1 & \text{if } (i,j,k) \text{ is an odd permutation of } (1,2,3) \\ 0 & \text{if } (i,j,k) \text{ has any other set of values} \end{cases}. \tag{3.3.4}$$

With this definition the non-zero components of ϵ are the following,

$$\epsilon_{123} = \epsilon_{231} = \epsilon_{312} = 1$$
$$\epsilon_{321} = \epsilon_{213} = \epsilon_{132} = -1$$

Moreover, as we can see from the above discussion, the Levi-Civita tensor is skew-symmetric (or antisymmetric) with respect to any two of its indices.

Now, a very important relation between the Levi-Civita tensor ϵ and the Kronecker tensor δ is the following

$$\epsilon_{ijk}\epsilon_{rsk} = \delta_{ir}\delta_{js} - \delta_{is}\delta_{jr} \tag{3.3.5}$$

The truth of (3.3.5) may be established as follows.

If $i = j$ or $r = s$, the right-hand side of (3.3.5) is zero and the left-hand side also vanishes by the definition of the Levi-Civita tensor. Consider the case when $i \neq j$ and $r \neq s$. Without loss of generality we may choose $i = 1$ and $j = 2$. Using the definition of the Levi-Civita tensor, the left-hand side of (3.3.5) then becomes

$$\epsilon_{121}\epsilon_{rs1} + \epsilon_{122}\epsilon_{rs2} + \epsilon_{123}\epsilon_{rs3} = \epsilon_{rs3}$$

The right-hand side of (3.3.5) becomes

$$\delta_{1r}\delta_{2s} - \delta_{1s}\delta_{2r} = \Delta.$$

where, Δ is a scalar. As $r \neq s$, there are just the following possibilities to consider :

$r = 3$ in which case $\Delta = 0$ for all s ;
$s = 3$ in which case $\Delta = 0$ for all r ;
$r = 3$, $s = 2$, giving $\Delta = 1$;
$r = 2$, $s = 1$, giving $\Delta = -1$;

Hence $\Delta = \epsilon_{rs3}$, and (3.3.5) is proved.

Furthermore, using equation (3.3.5) we can also show that

$$\epsilon_{ijk}\epsilon_{rjk} = 2\delta_{ir} \tag{3.3.6}$$

3.3.2 Cartesian and Spectral Decomposition of the Second Order Tensors.

In this section, we analyze second order Cartesian tensors in terms of their structural symmetries. Most of the results mentioned here are well known from the theory of linear transformations or from matrix theory. We state them here for later use in terms of second order Cartesian tensors.

First, let us state the following important definitions. A second order tensor $\mathbf{T}$ is said to be

(a) *Symmetric* if $\mathbf{T} = \mathbf{T}^T$

(b) *Skew-symmetric* if $\mathbf{T} = -\mathbf{T}^T$

(c) *Singular* if there exists a $\mathbf{v} \neq 0$ such that $\mathbf{T}\,\mathbf{v} = 0$

Now, an arbitrary Cartesian tensor $\mathbf{T}$ defined on an n-dimensional Euclidean space may always be decomposed into the sum of a symmetric and a skew-symmetric tensor, as follows

$$\mathbf{T} = \mathbf{T}_s + \mathbf{T}_{ss} \tag{3.3.7}$$

where, $\mathbf{T}_s$ is symmetric and $\mathbf{T}_{ss}$ is skew-symmetric. Both are given by

$$\mathbf{T}_s = \frac{1}{2}\left(\mathbf{T} + \mathbf{T}^T\right) \quad , \quad \mathbf{T}_{ss} = \frac{1}{2}\left(\mathbf{T} - \mathbf{T}^T\right). \tag{3.3.8}$$

We shall refer to (3.3.7) as the *Cartesian decomposition* of a tensor. Now, some important observations about the number of independent components for the tensors $\mathbf{T}_s$ and $\mathbf{T}_{ss}$ are in order. Each of $\mathbf{T}_s$ and $\mathbf{T}_{ss}$ has $n^2 - n$ off-diagonal ($i \neq j$) components, only half of which are independent numbers. The skew-symmetric tensor $\mathbf{T}_{ss}$ has only zeros on the diagonal and hence has just $\frac{1}{2}n(n-1)$ independent components. The symmetric tensor $\mathbf{T}_s$ contains the remaining $\frac{1}{2}n(n+1)$ pieces of information given by the n^2 components which form the original coordinate matrix T of the tensor $\mathbf{T}$.

We saw earlier that second order tensors constitute a vector space. It is easy to see now that the set of symmetric tensors form a subspace of this space. The same is also true for the set of skew-symmetric tensors. Obviously, these two subspaces are distinct. This allow us to view, by using the Cartesian decomposition, the tensor vector space as a *direct sum* of these two subspaces. Moreover, Proposition 3.13 (see Section 3.4) reveals that these two subspaces are also orthogonal, relative to the double dot product, and so one is the *orthogonal complement* of the other.

As we know, the tensor product of two vectors is a second order tensor. The sim-

plest, nontrivial example of a second order tensor which we can construct from the tensor product of two vectors is the "projection tensor" [11]. A projection tensor is symmetric and can be defined as follows.

Given two vectors $\mathbf{e}$ and $\mathbf{v}$, where $\mathbf{e}$ is a unit direction vector, the projection of $\mathbf{v}$ on $\mathbf{e}$ is denoted and defined by

$$\mathbf{P}_e \mathbf{v} = \mathbf{e}\,\mathbf{e} \cdot \mathbf{v} = \mathbf{e}(\mathbf{e} \cdot \mathbf{v}). \tag{3.3 9}$$

The tensor

$$\mathbf{P}_e = \mathbf{e}\,\mathbf{e} \tag{3.3.10}$$

which is defined as the tensor product (or dyad) of the unit vector $\mathbf{e}$, is called the *projection tensor* along the direction of $\mathbf{e}$. From the theory of projections, or by direct verification, we can see that the tensor

$$\begin{aligned} \mathbf{E}_e &= 1 - \mathbf{P}_e \\ &= 1(\mathbf{e} \cdot \mathbf{e}) - \mathbf{e}\,\mathbf{e} \end{aligned} \tag{3.3.11}$$

is also a projection tensor which projects any vector $\mathbf{v}$ onto a plane perpendicular to $\mathbf{e}$. From the foregoing it is obvious that a vector $\mathbf{v}$ can be decomposed into the following orthogonal components with respect to $\mathbf{e}$

$$\mathbf{v}_{\|_e} = \mathbf{P}_e \cdot \mathbf{v} \tag{3.3.12}$$

$$\mathbf{v}_{\perp_e} = \mathbf{E}_e \cdot \mathbf{v} \tag{3.3.13}$$

Now, given an orthogonal Cartesian coordinate system with basis vectors $\mathbf{e}_i$, we can define the projection tensors $\mathbf{P}_i \equiv \mathbf{P}_{e_i}$, $i = 1, 2, 3$. It is easy to see that these projection tensors are symmetric and have the following properties [15].

(a) *orthogonality*

$$\mathbf{P}_i : \mathbf{P}_j = 0 \quad \text{if } i \neq j \tag{3.3.14}$$

(b) *idempotence*

$$\mathbf{P}_i^2 = \mathbf{P}_i \tag{3.3.15}$$

(c) *completeness*

$$\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3 = 1. \tag{3.3.16}$$

Another important decomposition, for symmetric tensors, can be defined based on projection tensors. As is well known [18], any symmetric tensor $\mathbf{T}$ has real eigenvalues (λ_i) with a complete set of orthogonal eigenvectors. Therefore, if $\mathbf{P}_k$ is the projection tensor along the unit direction of the k-th eigenvector, we can write [15] for $\mathbf{T}$ the following *canonical form* or *spectral decomposition*

$$\mathbf{T} = \lambda_i \mathbf{P}_i = \lambda_1 \mathbf{P}_1 + \lambda_2 \mathbf{P}_2 + \lambda_3 \mathbf{P}_3 \tag{3.3 17}$$

The spectral decomposition of a symmetric tensor, will allow us to give simple proofs for some basic propositions in Section 3.4.

3.3.3 Tensor Invariants

a) Scalar Invariants

As we saw in section 3.3.1, except for isotropic tensors, the individual components of a tensor are not invariant. They depend on the coordinate systems. There are, however, a number of scalar invariants associated with every second order tensor, i.e., scalars which depend on the tensor itself, and not on the matrix representing it or its individual components. These numbers are known as *scalar invariants.*

Functional expressions for the scalar invariant of a tensor can be written in different forms. Thus, for example, for a general second order tensor $\mathbf{T}$, which has matrix representation T relative to some basis, we can define [16] four scalar invariants as follows.

i) $$I_1 = t_{ii} \tag{3.3.18}$$

i.e., I_1 is the trace of the tensor $\mathbf{T}$.

ii) $$I_2 = t_{22}t_{33} - t_{23}t_{32} + t_{33}t_{11} - t_{13}t_{31} + t_{11}t_{22} - t_{12}t_{21} \tag{3.3.19}$$

i.e., I_2 is the sum of principal minors of the tensor.

$$iii) \quad I_3 = t_{11}t_{22}t_{33} + t_{12}t_{23}t_{31} + t_{13}t_{21}t_{32} - t_{11}t_{23}t_{32} - t_{22}t_{13}t_{31} - t_{33}t_{12}t_{21} \tag{3.3.20}$$

i.e., I_3 is the determinant of $\mathbf{T}$.

$$iv) \quad I_4 = t_{ik}\, t_{ik} \tag{3.3.21}$$

In the case of symmetric tensors, I_4 is not independent of I_1, I_2, I_3. The three independent scalar invariants of a symmetric tensor are also known as *principal invariants* and in terms of the eigenvalues (λ_i) of the symmetric tensor are given by

$$I_1 = \lambda_1 + \lambda_2 + \lambda_3 = tr\,(\mathbf{T}) \tag{3.3.22}$$

$$I_2 = \lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1 \tag{3.3.23}$$

$$I_3 = \lambda_1\lambda_2\lambda_3 \tag{3.3.24}$$

Remark 3.6 : For a symmetric tensor $\mathbf{T}$, we can define the scalar invariants without resolving to the components of a matrix representation of $\mathbf{T}$. It can be shown [17], that the scalars $F_1 = \text{tr}\,(\mathbf{T})$, $F_2 = \text{tr}\,(\mathbf{T}^2)$, $F_3 = \text{tr}\,(\mathbf{T}^3)$ are invariant. Obviously, these scalar invariants are not independent from the principal invariants. It can be verified that

$$I_1 = F_1, \quad I_2 = \frac{1}{2}(F_1^2 - F_2) \quad \text{and} \quad I_3 = \frac{1}{6}(2F_3 + F_1^3 - 3F_1F_2)$$

b) Vector Invariants and Relatively Oriented Skew-Symmetric Tensors

Besides scalar invariants, we can associate to any second order tensor, defined on a 3-D Euclidean space, a vector which belongs in the 3-D Euclidean space and is defined as follows.

Let t_{kj} be the components of a tensor $\mathbf{T}$ relative to a Cartesian orthogonal basis $\{\mathbf{e}\}$. By considering the tensor product of ϵ_{ijk} and t_{kj} and contracting twice over the two common indices we have

$$t_i = \epsilon_{ijk}\, t_{kj}\,, \tag{3.3.25}$$

which, by using equation (3.3.4) for $i = 1, 2, 3$ gives

$$t_1 = t_{32} - t_{23} \tag{3.3.26a}$$

$$t_2 = t_{13} - t_{31} \tag{3.3.26b}$$

$$t_3 = t_{21} - t_{12}. \tag{3.3.26c}$$

Equation (3.3.26) implies that $\mathbf{t} = \mathbf{0}$, if $\mathbf{T}$ is symmetric, and $\mathbf{t}$ has components which are numerically twice those of $\mathbf{T}$, if $\mathbf{T}$ is skew-symmetric. Therefore, the vector

$$t_i = \frac{1}{2}\epsilon_{ijk}\, t_{kj} \tag{3.3.27}$$

is uniquely defined when the tensor $\mathbf{T}$ is given. We denote this vector by writing

$$\mathbf{t} = vect(\mathbf{T}) = vect(\mathbf{T}_{ss}) \tag{3.3.28}$$

where, $vect(\cdot)$ denotes the *tensor valued vector operator* which is defined by equation (3.3.27). The vector $\mathbf{t}$ is referred [23,24] to as the *vector (geometric) invariant* of $\mathbf{T}$.

Now, as we can see from equation (3.3.26), the *kernel* (or *null space*) of the vect operator is the subspace of the symmetric tensors, and therefore it is a non-empty set. Hence, the vect operator in not a 1–1 operator. Therefore, if we are seeking a 1–1 correspondence between tensors and their vector invariants we have to consider the restriction of the vect operator onto the subspace of the skew-symmetric tensors. To do this, we shall restrict our attention to the subspace of skew-symmetric tensors.

As we saw above to each skew-symmetric tensor there corresponds a vector, its vector invariant. Conversely, as we shall show, to any vector $\mathbf{t}$ in an 3-D Euclidean space there corresponds a skew-symmetric tensor, which we shall denote by $\tilde{\mathbf{t}}$. Moreover, the skew-symmetric tensor $\tilde{\mathbf{t}}$ has the important property that its vector invariant is the vector $\mathbf{t}$, from which the tensor $\tilde{\mathbf{t}}$ has been generated. To see this, given a vector $\mathbf{t}$, we define the tensor $\tilde{\mathbf{t}}$ by considering the tensor product of the Levi-Civita tensor ϵ with the vector $\mathbf{t}$ and contract the first index of ϵ with that of $\mathbf{t}$, i.e., we consider the equation

$$t_{kj} = \epsilon_{ijk}\, t_i \tag{3.3.29}$$

The tensor $\tilde{\mathbf{t}}$ is skew-symmetric, since the Levi-Civita tensor is antisymmetric with respect to the indices j and k. Moreover, if we multiply equation (3.3.29) by ϵ_{rjk} and use equation (3.3.6) we get

$$\epsilon_{rjk}\, t_{kj} = 2\delta_{ri}\, t_i ,$$

which, by using the definition of the Kroneker tensor δ_{ri}, can also be written as

$$t_r = \frac{1}{2}\epsilon_{rjk}\, t_{kj} .$$

This equation is equivalent to equation (3.3.27) and therefore, $\mathbf{t}$ is indeed the vector invariant of the skew-symmetric tensor $\tilde{\mathbf{t}}$.

From the foregoing, we can see that by considering the restriction of the vect operator onto the subspace of the skew-symmetric tensors, equation (3.3.27) (together with equation (3.3.29)) defines a 1–1 correspondence between skew-symmetric tensors and vectors. To express this 1–1 correspondence between skew-symmetric tensors and vectors, we introduce the following tensor-valued tensor operator,

$$dual(\cdot) \triangleq vect(\cdot)\Big|_{\left\{\text{second order skew-symmetric tensors}\right\}} \tag{3.3.30}$$

which is a 1–1 operator, as can be seen from equation (3.3.26).

The dual operator can also be defined in a component-wise manner, as follows :

Definition 3.4 : The *dual operator* is a 1–1 *tensor-valued tensor operator* which has the following property : when this operator is evaluated at a tensor of order one (i.e., a vector) we get a skew-symmetric tensor of order two, and when it is evaluated at a second order skew-symmetric tensor we get a tensor of order one. We define the action of the dual operator on a vector or a skew-symmetric tensor, component-wise, using the following 1–1 correspondence

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{3.3.31}$$

Symbolically, we denote the action of the dual operator on a vector $\mathbf{u}$ by writing

$$\tilde{\mathbf{u}} \triangleq dual(\mathbf{u}), \tag{3.3.32}$$

and, similarly

$$\mathbf{u} \triangleq dual(\tilde{\mathbf{u}}) \tag{3.3.33}$$

denotes the action of the dual operator on a skew-symmetric tensor $\tilde{\mathbf{u}}$.

It is important to note here that, for simplicity, we write $dual(\cdot)$ to denote both, the vector-valued and the tensor-valued dual operators. We shall rely on the argument to distinguish between the two cases (i.e., the "direct" or the "inverse" operator), and we shall referred to the tensor $\tilde{\mathbf{u}}$, defined by (3.3.32), as the *dual tensor* of the vector $\mathbf{u}$. Similarly, we shall refer to the vector $\mathbf{u}$, defined by equation (3.3.33), as the *dual vector* of the tensor $\tilde{\mathbf{u}}$.

It can be verified that the dual vector $\mathbf{u}$ of the skew-symmetric tensor $\tilde{\mathbf{u}}$ is simply its vector invariant as defined by equation (3.3.27). Therefore, Definition 3.4 and equation (3.3.30), both introduce the same tensor-valued tensor operator.

Obviously, the 1–1 correspondence, which has been established above, is not the only possible 1–1 correspondence between vectors and skew-symmetric tensors in a 3-D Euclidean space. For example, if instead of equation (3.3.31), we consider the following correspondence

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \Longleftrightarrow \begin{bmatrix} 0 & u_3 & -u_2 \\ -u_3 & 0 & u_1 \\ u_2 & -u_1 & 0 \end{bmatrix}, \tag{3.3.34}$$

we can define, as in Definition 3.4, another operator which also establishes a 1–1 correspondence between vectors and skew-symmetric tensors in the 3-D Euclidean space.

The tensor-valued tensor operator which describes, mathematically, this new 1–1 correspondence between vectors and skew-symmetric tensors will again be referred to as a dual operator. But, to express it symbolically, we shall use a different notation from that used for the dual operator given by Definition 3.4. In particular, to denote the

action of the dual operator, which is introduced here by the correspondence (3.3.34), at a vector or a skew-symmetric tensor, we shall write

$$\bar{\mathbf{u}} \triangleq (\mathbf{u})^{dual}, \tag{3.3.35}$$

and

$$\mathbf{u} \triangleq (\bar{\mathbf{u}})^{dual}, \tag{3.3.36}$$

respectively. In the following, we shall rely on the notation or the context to make clear which correspondence, i.e., (3.3.31) or (3.3.34), has to be used for the dual operator.

Now, as we can see from the correspondence (3.3.31), the coordinate matrix of the skew-symmetric tensor $\tilde{\mathbf{u}}$ has the following structure,

$$\tilde{u} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}, \tag{3.3.37}$$

i.e., the signs of its components follow a special pattern. Similarly, from the correspondence (3.3.34), we have that the coordinate matrix of the dual tensor $\bar{\mathbf{u}}$ is given by the following equation

$$\bar{u} = \begin{bmatrix} 0 & u_3 & -u_2 \\ -u_3 & 0 & u_1 \\ u_2 & -u_1 & 0 \end{bmatrix}. \tag{3.3.38}$$

To emphasize the fact that the coordinate matrices of the tensors $\tilde{\mathbf{u}}$ and $\bar{\mathbf{u}}$ have special sign patterns, we shall say that the tensors $\tilde{\mathbf{u}}$ and $\bar{\mathbf{u}}$ are *oriented* relative to their dual vector $\mathbf{u}$, or simply, that they are *relatively oriented.* In particular, we shall say that the dual tensor $\tilde{\mathbf{u}}$ has a *positive* or a *right-handed* orientation relative to $\mathbf{u}$. Similarly, we shall say that the tensor $\bar{\mathbf{u}}$ has a *negative* or *left-handed* orientation relative to the vector $\mathbf{u}$. This terminology and characterization for the relative orientation will become clear shortly when we assign a geometric meaning to the skew-symmetric tensors $\tilde{\mathbf{u}}$ and $\bar{\mathbf{u}}$.

Based on the relative orientation of the two tensors $\tilde{\mathbf{u}}$ and $\bar{\mathbf{u}}$, it would be more informative if we call *right dual* and *left dual* the two dual operators which are defined

by the correspondence (3.3.31), and (3.3.34) respectively. We shall do this if we need to emphasize which of the two dual operators is involved in the correspondence between vectors and skew-symmetric tensors.

Remark 3.7 : A *dual* operator is a peculiar characteristic of 3-D Euclidean space. This follows from the fact that a skew-symmetric tensor of order two defined in n-D Euclidean space has $\frac{1}{2}n(n-1)$ independent scalar components. Obviously then, if we view these scalars as the components of a vector, only when $n = 3$ does this vector belong to n-D Euclidean space.

3.3.4 A Geometric Characterization for the Second Order Skew-Symmetric Cartesian Tensors

It is well known, from spatial intuition, that in three dimensional space an unambiguous distinction can be made between two *orientations* of a plane relative to a normal vector. These are the *positive* or *right-handed* (or *counter-clockwise*) orientation and the *negative* or *left-handed* (or *clockwise*) orientation.

Geometrically, we can describe these two oriented planes, relative to a normal vector, as follows.

(*a*) (*b*)

Figure 3.1 : Relatively Oriented Planes and Skew-Symmetric Tensors

and we shall refer to them as *relatively oriented planes.*

Now, as is the case most of the time, geometric ideas are more useful in applications if we describe them algebraically. Therefore, to describe algebraically what Figure 3.1 shows geometrically, or to express in mathematical symbols the words "relatively oriented plane", we first note the following important points.

a) Any non-zero second order skew-symmetric tensor, defined in 3-D Euclidean space, has rank 2 [18]. This implies (see Definition 3.4) that to any non-zero skew-symmetric tensor we can associate two linearly independent directions, or equivalently, a plane in the 3-D physical space.

b) As we saw before, given a vector $\mathbf{u}$, there correspond to it two relatively oriented skew-symmetric tensors, namely, the tensors $\tilde{\mathbf{u}}$ and $\overline{\mathbf{u}}$.

Therefore, just as we consider, vectors to represent the geometric notion of directed line segments in vector analysis, we can consider in 3-D Cartesian tensor analysis, skew-symmetric tensors to represent the geometric notion of *directed plane segments* or *directed plane areas.* Moreover, we can use skew-symmetric dual tensors to represent relatively oriented planes. In particular, given a vector $\mathbf{u}$, we can use its right-handed dual tensor $\tilde{\mathbf{u}}$ to represent the positively oriented (relative to $\mathbf{u}$) plane area shown in Figure 3.1(a); similarly we can use its left-handed dual tensor $\overline{\mathbf{u}}$ to represent the negatively oriented (relative to $\mathbf{u}$) plane area which is shown in Figure 3.1(b). Thus, according to this convention, what Figure 3.1(a) describes geometrically, the correspondence (3.3.31) describes algebraically. Similarly, the correspondence (3.3.34) is the algebraic equivalence of Figure 3.1(b). Therefore, we shall denote the positively oriented plane in Figure 3.1(a) by $\tilde{\mathbf{u}}$, and similarly the negatively oriented plane in Figure 3.1(b) by $\overline{\mathbf{u}}$.

The identification of dual skew-symmetric tensors with relatively oriented plane areas reveals that the algebraic relationship which exists, between the vector $\mathbf{u}$ and the skew-symmetric dual tensors $\tilde{\mathbf{u}}$ and $\overline{\mathbf{u}}$, is actually equivalent to the geometric notion of

an orthogonal relationship. This identification implies that the vector $\mathbf{u}$ is normal to both dual tensors $\tilde{\mathbf{u}}$ and $\overline{\mathbf{u}}$. Obviously then, the vector $\mathbf{u}$ remains normal, and thus invariant, when the plane $\tilde{\mathbf{u}}$ (or $\overline{\mathbf{u}}$) rotates in space. Therefore, the terminology "vector invariant", which has also been assigned to the dual vector $\mathbf{u}$, becomes clear.

Also, it is obvious from Figure 3.1, that a vector $\mathbf{u}$, together with one of its relatively oriented tensors can be used to define an orientation for the 3-D physical space or the same, an oriented coordinate system for 3-D Euclidean space. In particular, the vector $\mathbf{u}$ and the dual tensor $\tilde{\mathbf{u}}$ define the right-handed orientation, as opposed to the vector $\mathbf{u}$ and the dual tensor $\overline{\mathbf{u}}$ which define the left-handed orientation. Moreover, as we can see, the difference between the right and left handed coordinate system orientations results from the relative orientation of the dual tensors $\tilde{\mathbf{u}}$ and $\overline{\mathbf{u}}$. From the foregoing, expressions from the nature language such as "right-handed" or "left-handed" coordinate systems, which are usually used to augment the analytic algebra (which is a mathematical language) of 3-D Euclidean space, are not necessary when a vector and the *dual* operators are used to define a coordinate system for the space. Now, since

$$\tilde{\mathbf{u}} = \overline{\mathbf{u}}^{\,T}, \tag{3.3.39}$$

it is obvious that by taking the transpose of a relatively oriented tensor, we actually change its orientation relative to its dual vector. Therefore, geometrically, the algebraic operation of taking the transpose of a relatively oriented skew-symmetric tensor means that we are changing the orientation of a reference coordinate system in the 3-D physical space. Now, as a consequence of this, we have the following. As is well known [23], a rotation tensor in 3-D Euclidean space can be written in the form

$$\mathbf{R} = \mathbf{u}\otimes\mathbf{u} + \cos(\theta)\left(\mathbf{1} - \mathbf{u}\otimes\mathbf{u}\right) + \tilde{\mathbf{u}}\sin(\theta) \tag{3.3.40}$$

where $\mathbf{u}$ is the axis of rotation and θ is the angle of rotation. Then, since we can write (see equation (3.4.14))

$$\mathbf{u}\otimes\mathbf{u} \equiv \mathbf{u}\,\mathbf{u} = \mathbf{1} + \tilde{\mathbf{u}}\,\tilde{\mathbf{u}},$$

equation (3.3.40) can be written as

$$\mathbf{R} = \mathbf{1} + \big(1 - cos(\theta)\big)\,\tilde{\mathbf{u}}\,\tilde{\mathbf{u}} + \tilde{\mathbf{u}}\sin(\theta). \tag{3.3.41}$$

Moreover, since the tensor $\tilde{\mathbf{u}}\,\tilde{\mathbf{u}}$ is symmetric, we have

$$\begin{aligned}\mathbf{R}^{T} &= \mathbf{1} + \big(1 - cos(\theta)\big)\,\tilde{\mathbf{u}}\,\tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{T}\sin(\theta) \\ &= \mathbf{1} + \big(1 - cos(\theta)\big)\,\tilde{\mathbf{u}}\,\tilde{\mathbf{u}} + \overline{\mathbf{u}}\sin(\theta).\end{aligned} \tag{3.3.42}$$

Now, using the assigned orientation to the skew-symmetric tensors $\tilde{\mathbf{u}}$ and $\overline{\mathbf{u}}$, we can see that equation (3.3.40) defines a positive or right-handed rotation about the axis $\mathbf{u}$, and equation (3.3.42) defines a negative or left-handed rotation of the same magnitude (angle) about the axis $\mathbf{u}$. This result provides, obviously, a geometric explanation why the transpose $\mathbf{R}^{T}$ of a rotation tensor $\mathbf{R}$ is equal to $\mathbf{R}^{-1}$, i.e., it is equal to its inverse.

As is well known, the usefulness of vector analysis in practical applications lies in the fact that the introduction of a coordinate system gives to the abstract mathematical notion of vectors a definite geometrical or physical substance; it makes a vector isomorphic to a directed line segment. By direct analogy, this has to be true for the skew-symmetric tensor analysis. Therefore, let us see how we can introduce a practical coordinate system in 3-D linear space of skew-symmetric tensors which will allow us to establish an isomorphism between skew-symmetric tensors and plane areas. Based on this isomorphism, one can give geometric representations to important algebraic tensor equations.

Let $\{\,\mathbf{e}\,\} \equiv \{\,\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\,\}$ be a right-handed Cartesian orthogonal basis for the 3-D Euclidean vector space. Then, using the right dual operator we define the three right-handed dual tensors $\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2$ and $\tilde{\mathbf{e}}_3$ which define an orthogonal (see equation (3.2.34)) basis for the linear space of the skew-symmetric tensors of order two. We denote this right handed basis of the 3-D linear space of the skew-symmetric tensors by writing $\{\,\tilde{\mathbf{e}}\,\} \equiv \{\,\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2, \tilde{\mathbf{e}}_3\,\}$. Now, by identifying dual skew-symmetric tensors with relatively oriented plane areas, we can use the three linearly independent, mutually

orthogonal, and positively oriented planes $\tilde{\mathbf{e}}_1$, $\tilde{\mathbf{e}}_2$ and $\tilde{\mathbf{e}}_3$, to define a "right-handed" orthogonal basis for the 3-D linear space of plane areas. It is true now that relative to the basis $\{\ \tilde{\mathbf{e}}\ \}$, skew-symmetric tensors and plane areas are isomorphic entities.

The following diagram describes, in more technical terms, the algebraic and geometric equivalence, as well as, the two isomorphisms which coordinate systems introduce between vectors and oriented line segments, and between skew-symmetric tensors and oriented plane areas.

$$\begin{array}{ccc} & \{\ \mathbf{e}\ \} & \\ \{\ \text{vectors}\ \} & \longleftrightarrow & \{\ \text{oriented lines segments}\ \} \\ \Big\updownarrow\ dual\,(\cdot) & & \text{Fig. 3.1}\ \Big\updownarrow \\ & \{\ \tilde{\mathbf{e}}\ \} & \\ \{\ \text{dual skew-symmetric tensors}\ \} & \longleftrightarrow & \{\ \text{relatively oriented planes}\ \} \end{array} \tag{3.3.43}$$

Now, let us consider a vector $\mathbf{u}$ in the 3-D physical space. This vector relative to the basis $\{\ \mathbf{e}\ \}$ can be written as

$$\mathbf{u} = u_1\mathbf{e}_1 + u_2\mathbf{e}_2 + u_3\mathbf{e}_3. \tag{3.3.44}$$

and, as is well known, in vector analysis we use the coordinate matrix

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{3.3.45}$$

to represent the vector $\mathbf{u}$ relative to the basis $\{\ \mathbf{e}\ \}$. However, by applying the right dual operator in equation (3.3.44), we get the following equation

$$\tilde{\mathbf{u}} = u_1\tilde{\mathbf{e}}_1 + u_2\tilde{\mathbf{e}}_2 + u_3\tilde{\mathbf{e}}_3. \tag{3.3.46}$$

Equation (3.3.46) implies that the same coordinate matrix u can also be used to represent a plane area in the 3-D linear space of planes. Hence, the coordinate matrix u can be used to represent the vector $\mathbf{u}$, relative to the basis $\{\ \mathbf{e}\ \}$, or the plane area $\tilde{\mathbf{u}}$, relative to the basis $\{\ \tilde{\mathbf{e}}\ \}$. Obviously, this has to be expected, since the two bases $\{\ \mathbf{e}\ \}$

and $\{ \tilde{\mathbf{e}} \}$ have the same dimension and therefore these two linear spaces are *isomorphic*. It is the particular physical or geometrical properties of their basis vectors that allow one to distinguish the linear space of vectors from the linear space of planes. As we can see, a basis vector from the basis $\{ \mathbf{e} \}$ has rank 1 as opposed to a basis "vector" from the basis $\{ \tilde{\mathbf{e}} \}$ which has rank 2.

Surveying the applied literature, one can hardly† find any reference to the linear space of plane areas. Usually, plane areas are "seen" as a result of a definite relationship between vectors, and not as independent quantities which can stand on their own and form a linear space. For example, in applications of classical mechanics, one finds two kinds of vectors, namely, the *polar* and the *axial* or *pseudo* vectors. The polar vectors are used to represent oriented line segments, and the axial vectors are used to represent oriented plane areas [12]. But, although, polar and axial vectors represent different physical quantities, we consider them as belonging to the same linear space. This implies that in the vector analysis of classical mechanics, we "see" plane areas indirectly. We consider the dual vector $\mathbf{u}$, and not the skew-symmetric tensor $\tilde{\mathbf{u}}$, to represent a relatively oriented area. However, as the following example shows, we can study relationships between plane areas directly in the linear space of planes, as we study relationships between directed lines in the linear space of vectors.

In applications, where a measure for the angle between two planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ is needed, we often use the cosine of the angle. To calculate the cosine of the angle between the planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$, we usually consider the following equation

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\| \mathbf{u} \| \, \| \mathbf{v} \|}. \tag{3.3.47}$$

where, $\mathbf{u}$ and $\mathbf{v}$ are the normal vectors to the planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$, respectively, and θ is the angle between them. Equation (3.3.47), makes no reference to the linear space of planes. It is defined by using the theory of the linear space of vectors. Alternatively, by

† One of the few exceptions is the work by Hestenes [25].

considering that planes form a linear space too, we can define the same measure for the angle between two planes directly without mentioning the linear space of vectors at all. As we saw before, equation (3.2.37) defines the cosine of the "angle" between two tensors. Therefore, for the angle between the two planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ we can write

$$\cos(\theta) = \frac{\tilde{\mathbf{u}} : \tilde{\mathbf{v}}^{T}}{\| \tilde{\mathbf{u}} \|_{F} \| \tilde{\mathbf{v}} \|_{F}}. \tag{3.3.48}$$

Obviously, equation (3.3.48) has to be equivalent to equation (3.3.47). To show that this is indeed the case we need the following equation

$$\tilde{\mathbf{u}} : \tilde{\mathbf{v}}^{T} = 2\mathbf{u} \cdot \mathbf{v} \tag{3.3.49}$$

which is proven in Section 3.4 (equation 3.4.21). Now, equation (3.3.49) and the definitions of the Euclidean vector norm ($\| \mathbf{u} \| = \sqrt{\mathbf{u} \cdot \mathbf{u}}$) and the Frobenius tensor norm (equation (3.2.35)), imply that

$$\| \tilde{\mathbf{u}} \|_{F} = \sqrt{2} \| \mathbf{u} \| . \tag{3.3.50}$$

Therefore, equation (3.3.48) follows on substituting equations (3.3.49) and (3.3.50) into equation (3.3.47).

Our objective here was not to provide a complete analysis for the linear space of planes. We simply wanted to point out a particular application which reveals that the outlined geometric interpretation for skew-symmetric tensors and their vector invariants allow one to give a concrete physical interpretation to tensor algebraic equations. This is important, since many practical problems are usually described, mathematically, in terms of tensor equations. Thus, by providing a new physical insight into these equations, we may find ways to manipulate them in order to obtain more efficient solutions for practical problems.

3.4 CARTESIAN TENSOR ALGEBRAIC IDENTITIES

In this section we shall prove a number of propositions which define important tensor equations. These equations will allow us to manipulate second order tensors very

efficiently as abstract objects, without the need to resort to coordinate bases [27,29]. Moreover, if needed, the transition from these tensor equations to the corresponding coordinate matrix equations is effortless. This is so, because the basic tensor algebraic operations, as defined in section 3.2.3, are formally the same as the basic algebraic operations in matrix theory.

To prove most of the propositions, which are presented in this section, we shall use the following well known equation [14,19,20]

$$\mathbf{u} \times \mathbf{v} = \tilde{\mathbf{u}} \cdot \mathbf{v} \tag{3.4.1}$$

which expresses the vector cross product of two vectors $\mathbf{u}$ and $\mathbf{v}$ as a right dot product between the dual tensor $\tilde{\mathbf{u}}$ and the vector $\mathbf{v}$.

In the following, unless mentioned otherwise, by a dual operator, we mean the right dual operator. We shall state a number of propositions in terms of this dual operator. Obviously, similar propositions can also be stated in terms of the left dual operator.

Proposition 3.1 : The dual operator is linear, i.e., it satisfies the following :

$$dual(k\,\mathbf{u}) = k\,\tilde{\mathbf{u}} \tag{3.4.2}$$

$$dual(\mathbf{u} + \mathbf{v}) = \tilde{\mathbf{u}} + \tilde{\mathbf{v}} \tag{3.4.3}$$

where, k is a scalar and $\mathbf{u}$, $\mathbf{v}$ are vectors.

Proof : The result follows from the correspondence (3.3.31). □

Proposition 3.2 : The right dot product of a dual tensor with a vector satisfies the following equation,

$$\tilde{\mathbf{u}} \cdot \mathbf{v} = -\tilde{\mathbf{v}} \cdot \mathbf{u} \tag{3.4.4}$$

Proof : The result follows from the anticommutativity of the vector cross product and equation (3.4.1). □

Proposition 3.3 : The left and right dot products of a vector and a dual tensor are anticommutative, i.e., they satisfy the equation

$$\mathbf{v} \cdot \tilde{\mathbf{u}} = -\tilde{\mathbf{u}} \cdot \mathbf{v} \tag{3.4.5}$$

Proof : The result follows from equation (3.2.28) and the fact that a dual tensor is skew-symmetric. □

We can combine Propositions (3.2) and (3.3) to write get another tensor identity :

$$\mathbf{v} \cdot \tilde{\mathbf{u}} = \tilde{\mathbf{v}} \cdot \mathbf{u} \tag{3.4.6}$$

Note : Equations (3.4.3)-(3.4.6) allow us to reorder the factors of a left or right dot product between dual tensors and vectors. This is often desirable when algebraic manipulations are needed for simplifying other complex tensor equations.

Now, by viewing a skew-symmetric tensor as a linear operator, we can state the following results.

Proposition 3.4 : The vector $\mathbf{u}$ provides a basis for the null space of its dual tensor $\tilde{\mathbf{u}}$, i.e.,

$$\tilde{\mathbf{u}} \cdot \mathbf{u} = \mathbf{u} \cdot \tilde{\mathbf{u}} = 0 \tag{3.4.7}$$

Proof : The result follows from the fact that $\mathbf{u} \times \mathbf{u} = 0$. □

Remark 3.8 : As we saw in Section 3.3.3, geometrically, the dual tensor $\tilde{\mathbf{u}}$ represents a relatively oriented plane which is normal to the vector $\mathbf{u}$. Now, we see that this orthogonality relationship between the tensor $\tilde{\mathbf{u}}$ and the vector $\mathbf{u}$ is expressed algebraically, by equation (3.4.7). Moreover, it is easy to see, that for a non-zero tensor $\tilde{\mathbf{u}}$ and a non-zero vector $\mathbf{v}$, the equation

$$\tilde{\mathbf{u}} \cdot \mathbf{v} = 0 \tag{3.4.8}$$

implies that the vector $\mathbf{v}$ is parallel to $\mathbf{u}$ and thus perpendicular to plane $\tilde{\mathbf{u}}$. Therefore, equation (3.4.8) can be used as the algebraic definition of the orthogonality condition between a plane and a vector.

Proposition 3.5 : Given the vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{u}$, $\mathbf{v}$ and the dual tensors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ we can write the following identity :

$$(\mathbf{a} \times \mathbf{u}) \cdot (\mathbf{b} \times \mathbf{v}) = -\mathbf{a} \cdot \tilde{\mathbf{u}}\, \tilde{\mathbf{v}} \cdot \mathbf{b} \tag{3.4.9}$$

Proof : It is well known [12] that the vector identity

$$\mathbf{a} \times \mathbf{u} \cdot \mathbf{x} = \mathbf{a} \cdot \mathbf{u} \times \mathbf{x}$$

is valid for any vector $\mathbf{x}$. This implies, that the identity is true and for $\mathbf{x} = \mathbf{b} \times \mathbf{v}$. Therefore, we can write

$$\begin{aligned} (\mathbf{a} \times \mathbf{u}) \cdot (\mathbf{b} \times \mathbf{v}) &= \mathbf{a} \cdot (\mathbf{u} \times (\mathbf{b} \times \mathbf{v})) \\ &= \mathbf{a} \cdot (\tilde{\mathbf{u}}\, (-\tilde{\mathbf{v}}\, \mathbf{b})) \\ &= -\mathbf{a} \cdot \tilde{\mathbf{u}}\, \tilde{\mathbf{v}} \cdot \mathbf{b} \end{aligned}$$ □

Proposition 3.6 : The dot product between two dual tensors can be written in terms of their vector invariants (dual vectors), as follows :

$$\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} = \mathbf{v}\,\mathbf{u} - \mathbf{u} \cdot \mathbf{v}\, \mathbf{1}. \tag{3.4.10}$$

Proof : Using equation (3.4.1) we can write the double vector cross product $\mathbf{u} \times (\mathbf{v} \times \mathbf{r})$ as

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{r}) = \tilde{\mathbf{u}} \cdot (\tilde{\mathbf{v}} \cdot \mathbf{r}) = \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} \cdot \mathbf{r} \tag{3.4.11}$$

Also, the same double vector cross product can be written [12] as

$$\begin{aligned} \mathbf{u} \times (\mathbf{v} \times \mathbf{r}) &= \mathbf{v}\,(\mathbf{u} \cdot \mathbf{r}) - (\mathbf{u} \cdot \mathbf{v})\mathbf{r} \\ &= (\mathbf{v}\,\mathbf{u} - \mathbf{u} \cdot \mathbf{v}\, \mathbf{1}) \cdot \mathbf{r} \end{aligned} \tag{3.4.12}$$

Therefore, equating (3.4.11) with (3.4.12) we have

$$\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} \cdot \mathbf{r} = (\mathbf{v}\,\mathbf{u} - \mathbf{u} \cdot \mathbf{v}\, \mathbf{1}) \cdot \mathbf{r} \tag{3.4.13}$$

from where the identity (3.4.10) follows, since (3.4.13) is true for every vector $\mathbf{r}$. □

Another useful form of equation (3.4.10) is the following

$$\mathbf{v}\,\mathbf{u} = \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} + (\mathbf{v} \cdot \mathbf{u})\mathbf{1} \tag{3.4.14}$$

which expresses the tensor product or dyad of two vectors in terms of their dual tensors.

Proposition 3.7 : The dual tensor $dual(\tilde{u} \cdot v)$ can be written as the difference of two dyads or the difference of two tensor dot product as follows.

$$dual(\tilde{u} \cdot v) = v\,u - u\,v = \tilde{u} \cdot \tilde{v} - \tilde{v} \cdot \tilde{u} \tag{3.4.15}$$

Proof : To prove (3.4.15), we shall use the double vector cross product $(u \times v) \times r$. From vector analysis it is known [12] that

$$\begin{aligned}(u \times v) \times r &= v\,u \cdot r - u\,v \cdot r \\ &= (v\,u - u\,v) \cdot r\end{aligned} \tag{3.4.16}$$

Moreover, we can use the dual operator to write $(u \times v) \times r$ as

$$(u \times v) \times r = dual(\tilde{u} \cdot v) \cdot r \tag{3.4.17}$$

Now, since (3.4.16) and (3.4.17) are true for every vector r we can state that

$$dual(\tilde{u} \cdot v) = v\,u - u\,v$$

Further, since

$$v\,u - u\,v = (v\,u - u \cdot v\,1) - (u\,v - u \cdot v\,1),$$

using equation (3.4.10) we get,

$$v\,u - u\,v = \tilde{u} \cdot \tilde{v} - \tilde{v} \cdot \tilde{u} \qquad \square$$

Remark 3.9 : As is well known [26], the set of skew-symmetric tensors constitutes a *Lie algebra* over the real field (or the complex field) with the operation of multiplication defined as

$$[\tilde{u}, \tilde{v}] = \tilde{u} \cdot \tilde{v} - \tilde{v} \cdot \tilde{u} \tag{3.4.18}$$

The multiplication, defined by (3.4.18), is referred to as the *Lie bracket* or *commutator*, and satisfies the following conditions :

(a) $[\tilde{u}, \tilde{v}]$ is linear in $\tilde{u}$ and $\tilde{v}$

(b) $[\tilde{u}, \tilde{u}] = 0$ for all $\tilde{u}$

(c) $[\tilde{\mathbf{u}}, [\tilde{\mathbf{v}}, \tilde{\mathbf{r}}]] + [\tilde{\mathbf{r}}, [\tilde{\mathbf{u}}, \tilde{\mathbf{v}}]] + [\tilde{\mathbf{v}}, [\tilde{\mathbf{r}}, \tilde{\mathbf{u}}]] = 0$

Condition (c) is usually referred to as the *Jacobi identity.* As we can see from (3.4.15) and (3.4.18) the commutator can be defined as the dual tensor of a vector

$$dual(\tilde{\mathbf{u}} \cdot \mathbf{v}) = [\tilde{\mathbf{u}}, \tilde{\mathbf{v}}]$$

i.e., we can use the dual operation to introduce the same Lie algebra.

Proposition 3.8 : The double dot product of two tensors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ is related to the dot product of their dual vectors $\mathbf{u}$ and $\mathbf{v}$ by the equation

$$\tilde{\mathbf{u}} : \tilde{\mathbf{v}} = -2\mathbf{u} \cdot \mathbf{v} \tag{3.4.19}$$

Proof : By the definition of the double dot product (equation (3.2.34)) we have

$$\tilde{\mathbf{u}} : \tilde{\mathbf{v}} = tr\,[\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}].$$

Now, using equation (3.4.10) we can write,

$$\begin{aligned} tr\,[\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}] &= tr\,[\mathbf{v}\,\mathbf{u} - \mathbf{u} \cdot \mathbf{v}\,\mathbf{1}] \\ &= tr\,[\mathbf{v}\,\mathbf{u}] - \mathbf{u} \cdot \mathbf{v}\; tr\,[\mathbf{1}] \\ &= \mathbf{u} \cdot \mathbf{v} - 3\mathbf{u} \cdot \mathbf{v} \\ &= -2\mathbf{u} \cdot \mathbf{v}, \end{aligned} \tag{3.4.20}$$

and this proves equation (3.4.19). □

Equation (3.4.19) can also be written as

$$\tilde{\mathbf{u}} : \tilde{\mathbf{v}}^{\,T} = 2\mathbf{u} \cdot \mathbf{v}. \tag{3.4.21}$$

The following proposition enable us to reorder the terms in dot products between dual tensors.

Proposition 3.9 :

$$\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} \cdot \tilde{\mathbf{u}} = \tilde{\mathbf{v}} \cdot \tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} + \tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} - \frac{1}{2} tr\,[\tilde{\mathbf{u}} \cdot \tilde{\mathbf{u}}]\tilde{\mathbf{v}} \tag{3.4.22}$$

Proof : The first two terms on the right-hand side of (3.4.22) can be written as

$$\begin{aligned}
\tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}} + \tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}} &= \tilde{\mathbf{v}}\cdot(\mathbf{u}\,\mathbf{u} - \mathbf{u}\cdot\mathbf{u}\,\mathbf{1}) + \tilde{\mathbf{u}}\cdot(\mathbf{v}\,\mathbf{u} - \mathbf{v}\cdot\mathbf{u}\,\mathbf{1}) \qquad \text{[by (3.4.10)]}\\
&= (\tilde{\mathbf{v}}\cdot\mathbf{u})\mathbf{u} - (\mathbf{u}\cdot\mathbf{u})\tilde{\mathbf{v}} + (\tilde{\mathbf{u}}\cdot\mathbf{v})\mathbf{u} - (\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}}\\
&= -(\tilde{\mathbf{u}}\cdot\mathbf{v})\mathbf{u} - (\mathbf{u}\cdot\mathbf{u})\tilde{\mathbf{v}} + (\tilde{\mathbf{u}}\cdot\mathbf{v})\mathbf{u} - (\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}} \qquad \text{[by (3.4.4)]}\\
&= -(\mathbf{u}\cdot\mathbf{u})\tilde{\mathbf{v}} - (\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}}\\
&= \frac{1}{2}tr\,[\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}]\tilde{\mathbf{v}} - (\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}} \qquad \text{[by (3.4.20)]}
\end{aligned}$$

Therefore, the right hand side of (3.4.22) becomes

$$\tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}} + \tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}} - \frac{1}{2}tr\,[\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}]\tilde{\mathbf{v}} = -(\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}} \tag{3.4.23}$$

The left hand side of (3.4.22) can be written as

$$\begin{aligned}
\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}} &= \tilde{\mathbf{u}}\cdot(\mathbf{u}\,\mathbf{v} - \mathbf{v}\cdot\mathbf{u}\,\mathbf{1}) \qquad \text{[by (3.4.10)]}\\
&= (\tilde{\mathbf{u}}\cdot\mathbf{u})\mathbf{v} - (\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}}\\
&= -(\mathbf{v}\cdot\mathbf{u})\tilde{\mathbf{u}} \quad , \quad \text{[since } \tilde{\mathbf{u}}\cdot\mathbf{u} = 0\text{]}
\end{aligned} \tag{3.4.24}$$

completing the proof. □

Proposition 3.10 : The dual tensor $dual\,(\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\mathbf{v})$ can be written in terms of the dual tensors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ as

$$dual\,(\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\mathbf{v}) = -[\tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}} + \tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}] + tr\,[\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}]\tilde{\mathbf{v}} \tag{3.4.25}$$

Proof : The left hand side of (3.4.25) can be expressed as

$$dual\,(\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\mathbf{v}) = dual\,(\tilde{\mathbf{u}}\cdot(\tilde{\mathbf{u}}\cdot\mathbf{v}))$$

Using equation (3.4.15), the above equation becomes

$$\begin{aligned}
dual\,(\tilde{\mathbf{u}}\cdot(\tilde{\mathbf{u}}\cdot\mathbf{v})) &= \tilde{\mathbf{u}}\cdot dual\,(\tilde{\mathbf{u}}\cdot\mathbf{v}) - dual\,(\tilde{\mathbf{u}}\cdot\mathbf{v})\cdot\tilde{\mathbf{u}}\\
&= \tilde{\mathbf{u}}\cdot(\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}} - \tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}}) - (\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}} - \tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}})\cdot\tilde{\mathbf{u}}\\
&= -2\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}} + \tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}} + \tilde{\mathbf{v}}\cdot\tilde{\mathbf{u}}\cdot\tilde{\mathbf{u}}\,.
\end{aligned}$$

Finally, using equation (3.4.22), we can simplify the above equation to get (3.4.25). □

The following proposition is fundamental in the sense that it enables us to give another formulation for the famous theorem of Euler on rotational rigid body motion. This will be demonstrated in the next chapter.

Proposition 3.11 : Let $\mathbf{I}$ be a symmetric tensor. Then the dual tensor $dual(\mathbf{I}\cdot\mathbf{v})$, where $\mathbf{v}$ is any vector, satisfies the following equation

$$dual(\mathbf{I}\cdot\mathbf{v}) = -[\mathbf{I}\cdot\tilde{\mathbf{v}} + \tilde{\mathbf{v}}\cdot\mathbf{I}] + tr[\mathbf{I}]\tilde{\mathbf{v}} \tag{3.4.26}$$

Proof : Since $\mathbf{I}$ is a symmetric tensor, it has a complete set of eigenvectors [18]. Let us denote the unit directions of those eigenvectors by $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Then, using equations (3.3.10) and (3.3.17), we can write $\mathbf{I}$ in its spectral decomposition form as

$$\mathbf{I} = \lambda_1\mathbf{x}\,\mathbf{x} + \lambda_2\mathbf{y}\,\mathbf{y} + \lambda_3\mathbf{z}\,\mathbf{z} \tag{3.4.27}$$

where, λ_i, $i = 1, 2, 3$ are the eigenvalues of $\mathbf{I}$. Moreover, using equations (3.4.14) and (3.3.22), we can write the spectral decomposition of $\mathbf{I}$ in terms of the dual tensor of its eigenvectors, i.e., we can write

$$\mathbf{I} = \lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}} + tr[\mathbf{I}]\mathbf{1} \tag{3.4.28}$$

Then, the right dot product $\mathbf{I}\cdot\mathbf{v}$ becomes

$$\mathbf{I}\cdot\mathbf{v} = \lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}}\cdot\mathbf{v} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}}\cdot\mathbf{v} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}}\cdot\mathbf{v} + tr[\mathbf{I}]\mathbf{v} \tag{3.4.29}$$

Now, using Proposition 3.1 and 3.10, we can write

$$dual(\mathbf{I}\cdot\mathbf{v}) = -\left\{\tilde{\mathbf{v}}\cdot[\lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}}] + [\lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}}]\cdot\tilde{\mathbf{v}}\right\}$$
$$+ \left(\lambda_1 tr[\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}}] + \lambda_2 tr[\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}}] + \lambda_3 tr[\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}}]\right)\tilde{\mathbf{v}} + tr[\mathbf{I}]\tilde{\mathbf{v}} \tag{3.4 30}$$

Then, from equation (3.4.20), for unit vectors we get

$$tr[\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}}] = tr[\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}}] = tr[\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}}] = -2$$

Therefore, we can write

$$dual(\mathbf{I}\cdot\mathbf{v}) = -\left\{\tilde{\mathbf{v}}\cdot\left[\lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}} + tr[\mathbf{I}]\mathbf{1}\right]\right.$$
$$\left. + \left[\lambda_1\tilde{\mathbf{x}}\cdot\tilde{\mathbf{x}} + \lambda_2\tilde{\mathbf{y}}\cdot\tilde{\mathbf{y}} + \lambda_3\tilde{\mathbf{z}}\cdot\tilde{\mathbf{z}} + tr[\mathbf{I}]\mathbf{1}\right]\cdot\tilde{\mathbf{v}}\right\} + tr[\mathbf{I}]\tilde{\mathbf{v}}$$

from where, using (3.4.28), we get (3.4.26). □

Proposition 3.12 : Let $\mathbf{I}$ be a symmetric tensor and $\tilde{\mathbf{u}}$ a dual skew symmetric tensor. Then, the tensor

$$\mathbf{I}' = \tilde{\mathbf{u}} \cdot \mathbf{I} \cdot \tilde{\mathbf{u}} \tag{3.4.31}$$

is a symmetric tensor.

Proof : Since $\mathbf{I}$ is a symmetric tensor, we can write $\mathbf{I}$ in its spectral decomposition as follows

$$\mathbf{I} = \lambda_1 \mathbf{x}\,\mathbf{x} + \lambda_2 \mathbf{y}\,\mathbf{y} + \lambda_3 \mathbf{z}\,\mathbf{z}$$

Then, for the tensor $\mathbf{I}' = \tilde{\mathbf{u}} \cdot \mathbf{I} \cdot \tilde{\mathbf{u}}$ we have

$$\mathbf{I}' = \lambda_1 \tilde{\mathbf{u}} \cdot \mathbf{x}\,\mathbf{x} \cdot \tilde{\mathbf{u}} + \lambda_2 \tilde{\mathbf{u}} \cdot \mathbf{y}\,\mathbf{y} \cdot \tilde{\mathbf{u}} + \lambda_3 \tilde{\mathbf{u}} \cdot \mathbf{z}\,\mathbf{z} \cdot \tilde{\mathbf{u}} \tag{3.4.32}$$

Now, let us consider the term $\tilde{\mathbf{u}} \cdot \mathbf{x}\,\mathbf{x} \cdot \tilde{\mathbf{u}}$. As we can see

$$\begin{aligned} \tilde{\mathbf{u}} \cdot \mathbf{x}\,\mathbf{x} \cdot \tilde{\mathbf{u}} &= \tilde{\mathbf{u}} \cdot (\mathbf{x} \otimes \mathbf{x}) \cdot \tilde{\mathbf{u}} \\ &= (\tilde{\mathbf{u}} \cdot \mathbf{x}) \otimes (\mathbf{x} \cdot \tilde{\mathbf{u}}) \\ &= -(\tilde{\mathbf{u}} \cdot \mathbf{x}) \otimes (\tilde{\mathbf{u}} \cdot \mathbf{x}) \qquad \text{[by (3.4.4)]} \end{aligned}$$

and since the tensor product of a vector with itself is a symmetric tensor we have that $\tilde{\mathbf{u}} \cdot \mathbf{x}\,\mathbf{x} \cdot \tilde{\mathbf{u}}$ is symmetric. Therefore, $\mathbf{I}'$ as a sum of symmetric tensors is symmetric. □

Proposition 3.13 : The double dot product of a symmetric tensor $\mathbf{I}$ and a skew-symmetric tensor $\mathbf{S}$ is always zero, i.e.,

$$\mathbf{I} : \mathbf{S} = tr\,[\mathbf{I} \cdot \mathbf{S}] = 0 \tag{3.4.33}$$

In other words, symmetric and skew-symmetric tensors, relative to the double dot product, are always orthogonal.

Proof : Without any loss of generality we can assume that the skew-symmetric tensor is the dual tensor $\tilde{\mathbf{s}}$ of a vector $\mathbf{s}$. Since $\mathbf{I}$ is symmetric, let

$$\mathbf{I} = \lambda_1 \mathbf{x}\,\mathbf{x} + \lambda_2 \mathbf{y}\,\mathbf{y} + \lambda_3 \mathbf{z}\,\mathbf{z}$$

be its spectral decomposition. Then

$$\mathbf{I}\cdot\tilde{\mathbf{s}} = \lambda_1 \mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}} + \lambda_2 \mathbf{y}\,\mathbf{y}\cdot\tilde{\mathbf{s}} + \lambda_3 \mathbf{z}\,\mathbf{z}\cdot\tilde{\mathbf{s}}$$

Now, let us consider any term from that sum, say the term $\lambda_1 \mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}}$. Then, using equation (3.4.6) we can write the term $\mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}}$ as follows

$$\begin{aligned}\mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}} &= \mathbf{x}\,(\tilde{\mathbf{x}}\cdot\mathbf{s}) \\ &= dual\,(\tilde{\mathbf{x}}\cdot\mathbf{s})\cdot\tilde{\mathbf{x}} + \mathbf{x}\cdot(\tilde{\mathbf{x}}\cdot\mathbf{s})1 \qquad \text{[by (3.4.14)]} \\ &= dual\,(\tilde{\mathbf{x}}\cdot\mathbf{s})\cdot\tilde{\mathbf{x}}\end{aligned}$$

since, $\mathbf{x}\cdot(\tilde{\mathbf{x}}\cdot\mathbf{s}) = (\mathbf{x}\cdot\tilde{\mathbf{x}})\cdot\mathbf{s} = 0$ by (3.4.7). This implies that

$$\begin{aligned}tr\,[\mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}}] &= tr\,[dual\,(\tilde{\mathbf{x}}\cdot\mathbf{s})\cdot\tilde{\mathbf{x}}] \\ &= -2\mathbf{x}\cdot(\tilde{\mathbf{x}}\cdot\mathbf{s}) \\ &= 0\end{aligned}$$

Therefore,

$$\begin{aligned}tr\,[\mathbf{I}\cdot\tilde{\mathbf{s}}] &= \lambda_1 tr\,[\mathbf{x}\,\mathbf{x}\cdot\tilde{\mathbf{s}}] + \lambda_2 tr\,[\mathbf{y}\,\mathbf{y}\cdot\tilde{\mathbf{s}}] + \lambda_3 tr\,[\mathbf{z}\,\mathbf{z}\cdot\tilde{\mathbf{s}}] \\ &= 0 \qquad\qquad \square\end{aligned}$$

Proposition 3.14 : Let $\mathbf{I}$ be a symmetric tensor, and $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ dual skew-symmetric tensors which correspond to the vectors $\mathbf{u}$ and $\mathbf{v}$ respectively. Then, the following equation

$$\mathbf{u}\cdot\mathbf{I}\cdot\mathbf{v} = -\,tr\,[\tilde{\mathbf{u}}\cdot\mathbf{J}\cdot\tilde{\mathbf{v}}] \qquad (3.4.34)$$

where $\mathbf{J} = -\mathbf{I} + \frac{1}{2}tr\,[\mathbf{I}]1$, is valid.

Proof : Using equation (3.4.20), we can write the left hand side of equation (3.4.34) as follows

$$\begin{aligned}\mathbf{u}\cdot\mathbf{I}\cdot\mathbf{v} &= -\frac{1}{2}tr\left[\tilde{\mathbf{u}}\cdot dual\,(\mathbf{I}\cdot\tilde{\mathbf{v}})\right] \\ &= -\frac{1}{2}tr\left[\tilde{\mathbf{u}}\cdot\left(-\mathbf{I}\cdot\tilde{\mathbf{v}} - \tilde{\mathbf{v}}\cdot\mathbf{I} + tr\,[\mathbf{I}]\tilde{\mathbf{v}}\right)\right] \qquad \text{[by (3.4.26)]} \\ &= -\frac{1}{2}tr\left[-\tilde{\mathbf{u}}\cdot\mathbf{I}\cdot\tilde{\mathbf{v}} - \tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\cdot\mathbf{I} + tr\,[\mathbf{I}]\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\right] \\ &= -\frac{1}{2}tr\left[-\tilde{\mathbf{u}}\cdot\mathbf{I}\cdot\tilde{\mathbf{v}} + \frac{1}{2}tr\,[\mathbf{I}]\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\right] - \frac{1}{2}tr\left[-\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\cdot\mathbf{I} + \frac{1}{2}tr\,[\mathbf{I}]\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\right].\end{aligned} \qquad (3.4.35)$$

Moreover, since $tr\,[\mathbf{S}\cdot\mathbf{T}] = tr\,[\mathbf{T}\cdot\mathbf{S}]$, for any tensor $\mathbf{S}$ and $\mathbf{T}$ we have

$$\begin{aligned} tr\,[\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\cdot\mathbf{I}] &= tr\,[\tilde{\mathbf{v}}\cdot\mathbf{I}\cdot\tilde{\mathbf{u}}] \\ &= tr\,[(\tilde{\mathbf{v}}\cdot\mathbf{I}\cdot\tilde{\mathbf{u}})^T] \\ &= tr\,[\tilde{\mathbf{u}}\cdot\mathbf{I}\cdot\tilde{\mathbf{v}}]. \end{aligned} \tag{3.4.36}$$

Therefore, using equation (3.4.36) in equation (3.4.35) we have

$$\begin{aligned} \mathbf{u}\cdot\mathbf{I}\cdot\mathbf{v} &= -tr\left[-\tilde{\mathbf{u}}\cdot\mathbf{I}\cdot\tilde{\mathbf{v}} + \frac{1}{2}tr\,[\mathbf{I}]\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}\right] \\ &= -tr\left[\tilde{\mathbf{u}}\cdot\left(-\mathbf{I} + \frac{1}{2}tr\,[\mathbf{I}]\mathbf{1}\right)\cdot\tilde{\mathbf{v}}\right] \\ &= -tr\,[\tilde{\mathbf{u}}\cdot\mathbf{J}\cdot\tilde{\mathbf{v}}] \end{aligned}$$

where $\mathbf{J} = -\mathbf{I} + \frac{1}{2}tr\,[\mathbf{I}]\mathbf{1}$ and this completes the proof. □

With this analysis for Cartesian tensors at our disposal, we proceed, in the next chapter, to describe in a tensor formulation the kinematic and dynamic equations of rigid body motion. In Chapter V, this tensor description for rigid body motion will form the base for the propcsed dynamic analysis of rigid-link open-chain robot manipulators.

3. 5 REFERENCES

[1] G. Ricci and T. Levi-Civita, *Methodes de calcul differentiel absolu et leurs applications*, (Paris, 1923). (Reprinted from *Mathematische Annalen*, tome 54, 1900).

[2] I. S. Sokolnikoff, *Tensor Analysis : Theory and Applications to Geometry and Mechanics of Continua*, John Wiley & Sons, New York, 1965.

[3] D. Lovelock and H. Rund, *Tensors, Differential Forms, and Variational Principles*, John Wiley & Sons, New York, 1965.

[4] R. L. Bishop and S. I. Goldberg *Tensor Analysis on Manifolds*, The Macmillan Company, New York, 1968.

[5] J. L. Synge and A. Schild, *Tensor Calculus*, University of Toronto Press, Toronto, 1949.

[6] S. F. Borg, *Matrix-Tensor Methods in Continuum Mechanics*, D. Van Nostrand Company, Princeton, New Jersey, 1963.

[7] M. S. Smith, *Principles & Applications of Tensor Analysis*, Howard W. Sams & Co., New York, 1963.

[8] A. J. McConnell, *Applications of Tensors Analysis*, Dover Publications, New York, 1947.

[9] A. I. Borisenko and I. E. Tarapov, *Vector and Tensor Analysis With Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1968.

[10] L. Brillouin, *Tensors in Mechanics and Elasticity*, Academic Press, New York, 1964.

[11] J. G. Simmonds, *A Brief on Tensor Analysis*, Springer-Verlag, New York, 1982.

[12] L. Brand, *Vector and Tensor Analysis*, John Wiley & Sons, New York, 1947.

[13] W. G. Bickley and R. E. Gibson, *Via Vector to Tensor*, The English Universities Press, London, 1962.

[14] H. Jeffreys, *Cartesian Tensors*, Cambridge University Press, Cambridge, 1961.

[15] G. Temple, *Cartesian Tensors : An Introduction*, John Wiley & Sons, New York, 1960.

[16] A. M. Goodbody, *Cartesian Tensors: With Applications to Mechanics, Fluid Mechanics and Elasticity*, Ellis Horwood, England, 1982.

[17] D. E. Bourne and P. C. Kendall, *Vector Analysis and Cartesian Tensors*, Thomas Nelson & Sons, England, 1977.

[18] J. Heading, *Matrix Theory for Physicists*, Longmans, London, 1958.

[19] W. Gibbs *Vector Analysis*, Dover Publications, New York, 1960.

[20] O. Bottema and B. Roth, *Theoretical Kinematics*, North-Holland Publishing Co., Amsterdam, 1978.

[21] E. J. Konopinski, *Classical Descriptions of Motion*, W. H. Freeman and Company, San Francisco, 1969.

[22] J. Angeles, *Spatial Kinematic Chains : Analysis, Synthesis, Optimization*, Springer-Verlag, New York, 1982.

[23] J. Angeles, *Rational Kinematics*, Springer-Verlag, New York, 1989.

[24] J. Angeles, "On the Numerical Solution of the Inverse Kinematics Problem", *The International Journal of Robotics Research*, pp. 21-37, Vol, 4, 1985.

[25] D. Hestenes, *New Foundations of Classical Mechanics*, D. Reidel Publishing Company, Dordrecht, Holland, 1986.

[26] R. Gilmore, *Lie Groups, Lie Algebras, and Some of Their Applications*, John Wiley & Sons, New York, 1974.

[27] C. A. Balafoutis, R. V. Patel and P. Misra, "Efficient Modeling and Computation of Manipulator Dynamics Using Orthogonal Cartesian Tensors", *IEEE J. Robotics and Automation*, Vol. 4, No. 6, pp. 665-676, 1988.

[28] C. A. Balafoutis, R. V. Patel and P. Misra, "A Cartesian Tensor Approach for Fast Computation of Manipulator Dynamics", *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1348-1353, Philadelphia, April 24-29, 1988.

[29] I. J. Wittenburg, *Dynamics of Systems of Rigid Bodies*, B. G. Teubner, Stuttgart, 1977.

[30] T. Crouch, *Matrix Methods Applied to Engineering Rigid Body Mechanics*, Pergamon Press, Oxford, 1981.

[31] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

CHAPTER IV

CARTESIAN TENSORS AND RIGID BODY MOTION

4.1 INTRODUCTION

As we have mentioned in Chapter I, the representation of the physical quantities which are involved in the formulation of the equations of motion of a rigid bodies system, determines the kind of mathematical analysis that will be used in deriving these equations. In the classical Newtonian formulation of rigid body dynamics, vectors are normally used to represent basic physical quantities and, therefore, vector analysis is used for deriving the equations of rigid body motion. In particular, vector analysis is imposed on classical Newtonian dynamics from the consideration that *angular rates* (i.e., linearly independent rates of change of a rigid body orientation) constitute the components of a vector quantity, the *angular velocity vector*. This consideration also assigns a vector character to other physical quantities which are defined in terms of the angular velocity vector such as *angular acceleration, angular momentum, external torque*, etc. However, as is well known [1-7, 14-19], angular velocity can also be described by a second order skew-symmetric Cartesian tensor, the *angular velocity tensor*. Obviously then, the tensor representation of angular velocity indicates that besides vector analysis, Cartesian tensor analysis can also be applied in rigid body dynamics.

Application of Cartesian tensor analysis (within the framework of the Newtonian approach) in rigid body dynamics requires that all the other physical quantities which are defined in terms of the angular velocity be treated as Cartesian tensors instead of vectors. Besides the angular velocity, tensor representations for the angular momentum and the external or resultant torque have appeared in the literature since 1931 in the work of Jeffreys [1]. Jeffreys introduced the angular momentum and torque tensors in his Cartesian tensor description for the Newtonian formulation of rigid body motion. But

his formulation for the equations of motion felt short of producing any significant computational advantages over the classical vector formulation. Therefore, in the Newtonian approach to classical dynamics, Jeffreys work has been largely ignored, and almost exclusively, the equations of rigid body motion are formulated following the vector approach [9-13].

Jeffreys approach did not succeed in improving the computational efficiency of the equations of motion, mainly because he follows a component-wise approach in his analysis and he did not use a tensor representation for the angular acceleration. This resulted in rather complex expressions for the computation of linear acceleration and torque vectors. In this chapter, following Jeffreys approach and using the angular velocity and angular acceleration tensors to simplify the differentiation, we shall provide a Cartesian tensor description for rotational rigid body motion. To this end, a tensor formulation for the Euler equation, which describes purely rotational rigid body motion, is derived. This new formulation of Euler's equation has the same simplicity as the classical vector formulation but it can be implemented far more efficiently.

The outline of this chapter is as follows : Section 4.2 deals with kinematic aspects of rigid body motion. In particular, we show that by using the angular velocity and angular acceleration tensors, the velocity and acceleration of any point on the rigid body can by computed very easily. Section 4.3 deals with dynamic aspects of rigid body motion. In particular, an analysis of the rigid body inertia tensor is given and the vector formulation of rigid body motion is reviewed. From this, the angular momentum and torque tensors surface naturally and they lead to a new formulation for Euler's equation.

4.2 On the Kinematic Analysis of Rigid Body Motion

The kinematic analysis of rigid body motion deals with motion without regard to forces or moments that cause that motion. In particular, the kinematic analysis of rigid body motion is concerned with *configuration* and *motion* kinematic analysis.

Configuration kinematic analysis deals with possible descriptions of the rigid body spatial configuration as a function of time, and motion kinematic analysis deals with the first and second time derivatives of these configuration functions. We dealt with configuration kinematic analysis in Chapter II. In this section we shall be concerned mainly with motion kinematic analysis of rigid body rotational motion.

As we have outlined in section 2.2, the spatial configuration of a rigid body, relative to a Cartesian orthogonal coordinate system, is defined by considering the description of a position vector and the description of a rotation tensor where the latter defines the orientation of the rigid body. Moreover, as is well known from Chasles Theorem (Theorem 2.2), a general rigid body motion can be considered as the superposition of a purely translational and a purely rotational rigid body motion. Therefore, obviously the decomposition of the rigid body spatial configuration, as outlined above, implies that the position vector and its time derivatives describe purely translational rigid body motion and, similarly, the rotation tensor and its time derivatives describe purely rotational rigid body motion. The latter follows also from the fact that a rotation tensor can be used to describe a finite displacement about a fixed point (see Chapter II), which in the 3-D physical space is equivalent to finite rigid body displacement about a fixed axis. Hence, a rotation tensor in the 3-D physical space, when it is considered to be a continuous function of time, and its time derivatives are sufficient to study pure rotational rigid body motion.

Therefore, for the kinematic analysis of a rigid body pure rotational motion we need only to consider a rotation tensor $\mathbf{R}$, which defines the rigid body orientation, and its first and second time derivatives. In the following, we shall denote the time dependent rotation tensor by $\mathbf{R}(t)$. Also, for the absolute time derivatives (i.e., the time derivatives relative to an inertia frame) of $\mathbf{R} \equiv \mathbf{R}(t)$, we shall use the classical Newtonian notation, i.e., we write $\frac{d\mathbf{R}}{dt} \equiv \dot{\mathbf{R}}$ and $\frac{d^2\mathbf{R}}{dt^2} \equiv \ddot{\mathbf{R}}$.

4.2.1 The Angular Velocity Tensor

It is well known [2-9] that the angular rates (or linear independent rates of change of the rigid body orientation) constitute the components of a tensor quantity, the *angular velocity tensor*. But this is not reflected in the applied literature. Even in the cases where the angular velocity tensor is mentioned, its vector invariant or dual vector, i.e., the familiar *angular velocity vector*, is used in applications instead of the tensor itself. This does not mean that the angular velocity vector is more important than the angular velocity tensor. On the contrary, as we shall see in this chapter, the angular velocity tensor is more suitable for the kinematic and dynamic analysis of rigid body rotational motion.

Perhaps the easiest way to introduce the angular velocity tensor is by using a corollary of the following theorem [2].

Theorem 4.1 : Any differentiable orthogonal tensor $\mathbf{Q} \equiv \mathbf{Q}(t)$ satisfies the following first order differential equation

$$\dot{\mathbf{Q}} = \Phi \mathbf{Q} \tag{4.2.1}$$

where, Φ is a second order skew-symmetric tensor.

Proof : Since for any orthogonal tensor we have $\mathbf{Q}^T \mathbf{Q} = 1$, we can write

$$\dot{\mathbf{Q}} = \dot{\mathbf{Q}} \mathbf{Q}^T \mathbf{Q} . \tag{4.2.2}$$

Now, let

$$\Phi = \dot{\mathbf{Q}} \mathbf{Q}^T . \tag{4.2.3}$$

Then, (4.2.2) can be written as (4.2.1). Therefore, $\mathbf{Q}$ satisfies a first order differential equation. Now, we shall show that Φ, as defined by (4.2.3), is skew-symmetric. It follows from the orthogonality of $\mathbf{Q}$ that

$$\dot{\mathbf{Q}} \mathbf{Q}^T + \mathbf{Q} \dot{\mathbf{Q}}^T = 0$$

or

$$\Phi \mathbf{Q}\,\mathbf{Q}^{T} + \mathbf{Q}\,[\Phi \mathbf{Q}\,]^{T} = 0$$

or

$$\Phi + \Phi^{T} = 0$$

i.e.,

$$\Phi = -\,\Phi^{T}$$

and this completes the proof. []

Now, since a rotation tensor is an orthogonal tensor, it satisfies Theorem 4.1 and we can state the following corollary.

Corollary 4.1 : A rotation tensor $\mathbf{R}$ satisfies a first order differential equation given by

$$\dot{\mathbf{R}} = \Phi \mathbf{R} \tag{4.2.4}$$

where, Φ is a second order skew-symmetric tensor.

From this Corollary we can derive the definition of the angular velocity tensor as follows : When the rotation tensor $\mathbf{R}$ is defined in a 3-D Euclidean space, it describes the orientation of a rigid body. In this case, we denote the skew-symmetric tensor Φ by $\tilde{\omega}$, i.e., we write equation (4.2.4) as

$$\dot{\mathbf{R}} = \tilde{\omega} \mathbf{R}\,. \tag{4.2.5}$$

and refer to $\tilde{\omega}$ as the *angular velocity tensor* [9]. Equation (4.2.5), which is sometimes referred to as *Poisson's equation*, can also be written in the form

$$\tilde{\omega} = \dot{\mathbf{R}}\,\mathbf{R}^{T} \tag{4.2.6}$$

and can be used as the definition of the angular velocity tensor. Moreover, since the skew-symmetric tensor $\tilde{\omega}$ is defined in a 3-D Euclidean space, it has a unique dual vector or vector invariant. The dual vector ω, of $\tilde{\omega}$, is the familiar *angular velocity vector*. Therefore, equation (4.2.6) and the dual tensor operator provide a simple definition for the angular velocity vector.

A tensor representation for angular velocity has many advantages over the classical vector representation. First of all, from a practical point of view, a tensor treatment for angular rates allows us to relate the angular velocity to the total derivative of another

tensor physical quantity, namely, the total derivative of the orientation tensor $\mathbf{R}$ as expressed by equation (4.2.5). This is not possible when we describe the angular rates with the angular velocity vector ω, since it is well known [10] that there is no vector physical quantity whose total derivative is related to the angular velocity vector. Also from a theoretical point of view, the tensor representation for angular rates is not restricted to a 3-D Euclidean space. As we can see, by generalizing equation (4.2.5) from a 3-D to an n-D Euclidean space, the tensor $\tilde{\omega}$ becomes Φ. Therefore, by analogy, we can refer to the skew-symmetric tensor Φ as the angular velocity tensor in an n-D Euclidean space. The angular velocity tensor Φ is well defined in any n-D Euclidean space. This unfortunately is not true for the angular velocity vector which exists only in a 3-D Euclidean space. This reveals that the angular velocity tensor $\tilde{\omega}$ is *the* "primitive" physical quantity which describes the angular velocity. Therefore, the proper mathematical representation for the angular velocity is a tensor representation and this implies that the proper mathematical description for rigid body rotational motion is provided by tensor analysis instead of vector analysis.

4.2.2 The Angular Acceleration Tensor

Let us consider now the second time derivative $\ddot{\mathbf{R}}$, relative to an inertial frame, of a rotation tensor $\mathbf{R}$. The functional relationship between $\ddot{\mathbf{R}}$ and $\mathbf{R}$ follows from the following theorem.

Theorem 4.2 : Any differentiable orthogonal tensor $\mathbf{Q} \equiv \mathbf{Q}(t)$ satisfies the following second order differential equation

$$\ddot{\mathbf{Q}} = \Psi \mathbf{Q} \tag{4.2.7}$$

where, Ψ is a second order tensor defined by

$$\Psi = \dot{\Phi} + \Phi^2 , \tag{4.2.8}$$

Φ is the angular velocity tensor defined by (4.2.3) and $\Phi^2 = \Phi \cdot \Phi \equiv \Phi\Phi$.

Proof : Differentiation of (4.2.1) gives

$$\ddot{\mathbf{Q}} = \dot{\mathbf{\Phi}}\mathbf{Q} + \mathbf{\Phi}\dot{\mathbf{Q}}$$

and substituting for $\dot{\mathbf{Q}}$ we get

$$\begin{aligned}\ddot{\mathbf{Q}} &= (\dot{\mathbf{\Phi}} + \mathbf{\Phi}\mathbf{\Phi})\mathbf{Q} \\ &= \mathbf{\Psi}\mathbf{Q}\end{aligned}$$

where $\mathbf{\Psi} = \dot{\mathbf{\Phi}} + \mathbf{\Phi}^2$. This complete the proof. □

Corollary 4.2 : A second order rotation tensor $\mathbf{R}$ satisfies the following second order differential equation

$$\ddot{\mathbf{R}} = \mathbf{\Psi}\mathbf{R} \tag{4.2.9}$$

where, $\mathbf{\Psi}$ is a second order tensor defined by

$$\mathbf{\Psi} = \dot{\mathbf{\Phi}} + \mathbf{\Phi}^2, \tag{4.2.10}$$

$\mathbf{\Phi}$ is the angular velocity tensor which corresponds to $\mathbf{R}$ and $\mathbf{\Phi}^2 = \mathbf{\Phi}\mathbf{\Phi}$.

Theorem 4.2 and its Corollary are obviously valid for orthogonal tensors defined in an n-D Euclidean space, but here we shall be concerned only with the particular case where $n = 3$. In this case, i.e., when $n = 3$, we introduce the notation $\mathbf{\Omega}$ to denote the tensor $\mathbf{\Psi}$ and write equation (4.2.9) as

$$\ddot{\mathbf{R}} = \mathbf{\Omega}\mathbf{R}\,. \tag{4.2.11}$$

Also, using the corresponding notation for the tensor $\mathbf{\Phi}$, we write equation (4.2.10) as

$$\mathbf{\Omega} = \dot{\bar{\omega}} + \bar{\omega}\bar{\omega} \tag{4.2.12}$$

and we refer to the tensor $\mathbf{\Omega}$ as the *angular acceleration tensor*. Obviously, using equation (4.2.9), we can also define the angular acceleration tensor $\mathbf{\Omega}$ by the equation

$$\mathbf{\Omega} = \ddot{\mathbf{R}}\,\mathbf{R}^T \tag{4.2.13}$$

where $\mathbf{R}$ is a rotation tensor in a 3-D Euclidean space.

The tensor $\mathbf{\Omega}$ is neither symmetric nor skew-symmetric. Actually, since $\dot{\bar{\omega}}$ is skew-symmetric and $\bar{\omega}\bar{\omega}$ is symmetric, equation (4.2.12) represents the *Cartesian decomposition* of $\mathbf{\Omega}$. Moreover, since the tensor $\bar{\omega}\bar{\omega}$ is symmetric we have from equations (3.3.28) and (3.3.30) that

$$vect\,(\mathbf{\Omega}) = vect\,(\dot{\bar{\omega}}) \equiv dual\,(\dot{\bar{\omega}}) = \dot{\omega} \tag{4.2.14}$$

i.e., the vector invariant of the angular acceleration tensor Ω is the familiar *angular acceleration vector*. This obviously justifies the name given to the tensor Ω.

The angular acceleration tensor has only recently [5,14,15,17] been reported in the literature. This is probably due to the fact that a vector approach in analyzing rigid body motion fails to establish a clear relationship between vector invariants and tensors other than skew-symmetric tensors. In the case of the angular velocity vector and angular velocity tensor, a 1–1 relationship between them is obvious, since the angular velocity tensor is skew-symmetric. Thus, in this case, one representation is the dual of the other. A similar dual relationship between the angular acceleration vector and angular acceleration tensor does not exist because the angular acceleration tensor is not skew-symmetric. The dual tensor of the angular acceleration vector defines only the skew-symmetric part of the angular acceleration tensor and not the whole tensor. As we can see from equation (4.2.12), to define the angular acceleration tensor we need to use not only the angular acceleration vector but also the angular velocity vector. Therefore, the transition from vector analysis to tensor analysis is not straightforward.

We shall conclude this section with applications of the angular velocity and angular acceleration tensors in the computation of the linear velocity and acceleration of points on a moving rigid body.

4.2.3 Linear Velocity and Acceleration in Rigid Body Motion

The concepts of angular velocity and angular acceleration tensors provides a powerful tool for describing the motion of a rigid body since they enable us to derive equations with very simple structure. To see this, let us consider some arbitrary vector involved in a mechanical problem, such as the position vector $\mathbf{r}$ of a point on the rigid body. Usually such a vector will vary in time as the body moves. Therefore, it is important that its linear velocity and acceleration relative to an inertial coordinate system can be determined in a computational efficient manner. To solve problems of this type, we proceed

as follows.

As is usually the practice in rigid body motion, we consider two coordinate systems. An inertial and a body coordinate system, which we denote by $\{ \mathbf{e} \}$ and $\{ \mathbf{e}' \}$ respectively. The body coordinate system is rigidly attached to the rigid body and so moves with it. Suppose, now, that a point P on the body has a position vector $\mathbf{r}_1$ relative to the origin o of the inertial coordinate system and a position vector $\mathbf{r}_2$ relative to the origin o' of the moving coordinate system. Let also $\mathbf{s}$ be the vector from o to o'. Then, as we can see in the following figure

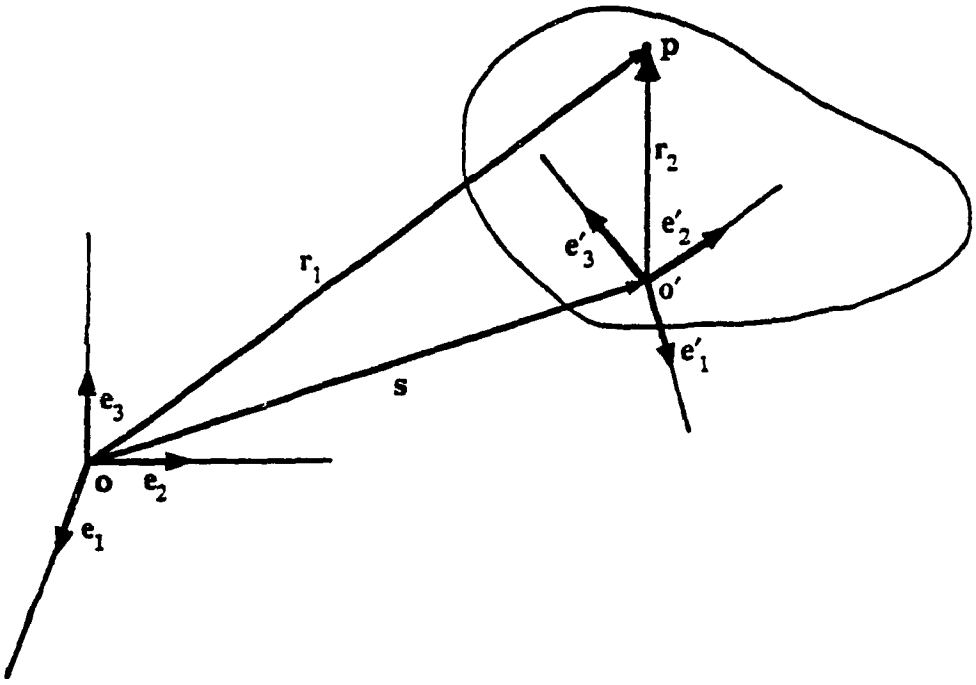

Figure 4.1 : Position Vectors and Coordinate Systems in Rigid Body Motion

the three vectors are related by the equation

$$\mathbf{r}_1 = \mathbf{s} + \mathbf{r}_2. \tag{4.2.15}$$

Moreover, let $\mathbf{R}$ be the rotation tensor which specifies the orientation of the moving coordinate system relative to the inertial one. Then, if $\mathbf{r}_2'$ denotes the position vector of the point P relative to o' when it is expressed in the moving coordinate system, premultiplication by the rotation tensor $\mathbf{R}$ expresses it relative to the inertial coordinate system, i.e., we can write

$$\mathbf{r}_2 = \mathbf{R}\,\mathbf{r}_2' \tag{4.2.16}$$

Therefore, equation (4.2.15) can also be written as

$$\mathbf{r}_1 = \mathbf{s} + \mathbf{R}\,\mathbf{r}_2'\,. \tag{4.2.17}$$

Now, as is well known, the vector of the absolute linear velocity of the point P is defined to be the first time derivative of its position vector relative to the inertial coordinate system. In other words the absolute linear velocity of the point P is given by the vector $\dot{\mathbf{r}}_1$. To compute this derivative we shall use equation (4.2.17). Thus, we have

$$\dot{\mathbf{r}}_1 = \dot{\mathbf{s}} + \dot{\mathbf{R}}\,\mathbf{r}_2'\,, \tag{4.2.18}$$

since the vector $\mathbf{r}_2'$ is time independent. Now, substituting for $\dot{\mathbf{R}}$ from equation (4.2.5) and using equation (4.2.16), we finally have for the absolute velocity of the point P

$$\dot{\mathbf{r}}_1 = \dot{\mathbf{s}} + \tilde{\omega}\mathbf{r}_2 \tag{4.2.19}$$

where $\tilde{\omega}$ is the angular velocity tensor of the moving body. Also, the second time derivative of equation (4.2.17) defines the vector of the absolute linear acceleration of the point P relative to the inertial coordinate system. Therefore, for the absolute linear acceleration of the point P we have

$$\ddot{\mathbf{r}}_1 = \ddot{\mathbf{s}} + \ddot{\mathbf{R}}\,\mathbf{r}_2'$$

which can be simplified to

$$\ddot{\mathbf{r}}_1 = \ddot{\mathbf{s}} + \Omega\mathbf{r}_2 \tag{4.2.20}$$

where Ω is the angular acceleration tensor of the moving body. Moreover, as we can see from equation (4.2.16), the absolute time derivatives for vectors which are constant rela-

tive to the body coordinate system are computed by using the simple equations

$$\dot{\mathbf{r}}_2 = \tilde{\omega}\mathbf{r}_2 \tag{4.2.21}$$

and

$$\ddot{\mathbf{r}}_2 = \Omega\mathbf{r}_2 \tag{4.2.22}$$

Let us recall at this point, that in the classical vector description of rigid body dynamics, the vectors of the absolute linear velocity and acceleration of the same point P are computed from the following vector equations

$$\dot{\mathbf{r}}_1 = \dot{\mathbf{s}} + \omega\times\mathbf{r}_2 \tag{4.2.23}$$

and

$$\ddot{\mathbf{r}}_1 = \ddot{\mathbf{s}} + \dot{\omega}\times\mathbf{r}_2 + \omega\times(\omega\times\mathbf{r}_2) \tag{4.2.24}$$

respectively. Obviously, equation (4.2.19) is equivalent to equation (4.2.23) and the same is true for equations (4.2.20) and (4.2.24). However, as we can see, the introduction of the angular velocity and angular acceleration tensors enables us to derive a simple and compact representation for the absolute linear velocity and acceleration of points on a moving rigid body. In particular, equation (4.2.20) enables us to manipulate very effectively equations involving the linear acceleration of various position vectors on the same rigid body. From the foregoing, we see that the introduction of the angular velocity and angular acceleration tensors provides more efficient means for the analysis of motion kinematics. Moreover, as we shall see in the following section, the angular velocity and angular acceleration tensors can be used to simplify motion dynamics as well.

4.3 On the Dynamic Analysis of Rigid Body Motion

In the dynamic analysis of motion, we deal with relationships between the motion of a body and the forces and/or torques which cause or result from that motion. As is well known from classical dynamics [9-13], for the dynamic analysis of rigid body motion a number of schemes have been developed over the years such as those based on the

equations of *d'Alembert, Newton-Euler, Euler-Lagrange, Hamilton* and others. The chief value of all of these schemes is that they offer different ways of tackling the problem of describing motion. Thus a great variety of viewpoints is available and this can help us to clarify what is essential for an efficient description of the equations of motion. In this section we shall be concerned with the Newtonian formulation of the equations of rigid body motion.

As we have mentioned before, a general motion of a rigid body can be considered as resulting from the superposition of two independent motions; namely a pure translational motion of a point (usually its center of mass) and a pure rotational motion about that point. The Newton-Euler procedure uses exactly this decomposition. In particular, in the Newtonian formulation of the equations of rigid body motion, the translational motion is described by Newton's equation (or Newton's second law) which symbolically is stated as follows

$$\mathbf{F}_C = m\,\ddot{\mathbf{r}}_C \tag{4.3.1}$$

where $\mathbf{F}_C$ is the total external (or resultant) force acting on the rigid body, m is the mass of the rigid body and $\ddot{\mathbf{r}}_C$ is the absolute linear acceleration of its center of mass. The rotational motion is described by Euler's equation which is symbolically stated as

$$\mathbf{M}_C = \mathbf{I}_C \cdot \dot{\omega} + \omega \times \mathbf{I}_C \cdot \omega \tag{4.3.2}$$

where $\mathbf{M}_C$ is the total external (or resultant) torque about the center of mass, $\mathbf{I}_C$ is the inertia tensor of the body about its center of mass and $\omega\,(\dot{\omega})$ is the angular velocity (acceleration) of the body.

Equations (4.3.1) and (4.3.2) are the fundamental equations which describe the rigid body motion in the classical Newtonian formulation. As we can see, these two vector equations provide six differential scalar equations which, when the external force and torque (with appropriate initial conditions) are given, can be solved to determine the six degrees of freedom of a rigid body in the three dimensional physical space, i.e., the posi-

tion of its center of mass and its orientation.

It is obvious from equations (4.3.1) and (4.3.2) that the dynamic analysis of rigid body motion in the classical Newtonian formulation, for both translational and rotational rigid body motion, is based on vector analysis. In this section, as an alternative to vector analysis, we shall use Cartesian tensors to analyze the rotational rigid body motion. Since in rigid body rotational motion, the inertia tensor of the body plays an important role, we first review some relevant facts about the rigid body inertia tensor.

4.3.1 The Rigid Body Inertia Tensor

As is well known [9,10], the *inertia tensor* of a rigid body characterizes the mass distribution of the body relative to a point, and is usually defined by the equation

$$\mathbf{I}_o = \int_m (\mathbf{r} \cdot \mathbf{r}\,\mathbf{1} - \mathbf{r}\,\mathbf{r})\,dm \tag{4.3.3}$$

where o denotes a point of the body and $\mathbf{r}$ denotes the position vector of a point mass relative to the point o.

The rigid body inertia tensor, as defined by equation (4.3.3), is used extensively in the dynamic analysis of rigid body motion. Actually, it is the only definition which most books provide for the inertia tensor, especially when a vector treatment of the Newtonian dynamic analysis of rigid body motion is followed. For a treatment of the dynamic analysis of rigid body motion based on Cartesian tensors, this definition for the inertia tensor needs to be modified. As we shall see later, the proper definition for a Cartesian tensor formulation of rigid body rotational dynamics is provided by the equation

$$\mathbf{J}_o = \int_m \mathbf{r}\,\mathbf{r}\,dm \tag{4.3.4}$$

where o is a point on the rigid body, $\mathbf{r}$ is the position vector of a point mass relative to point o and $\mathbf{r}\,\mathbf{r}$ denotes the dyad or tensor product of $\mathbf{r}$ with itself. We shall refer to the inertia tensor $\mathbf{J}_o$ as the *pseudo-inertia* tensor of a rigid body.

Obviously, the two inertia tensors $\mathbf{I}_o$ and $\mathbf{J}_o$ describe the same physical property of a rigid body, and thus they have to be equivalent. To see this, we proceed as follows [19]. First we note that equation (4.3.3) can be written in the form

$$\mathbf{I}_o = -\int_m \tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\, dm \tag{4.3.5}$$

if one uses the tensor equation (3.4.10). Then, starting from equation (4.3.4) and using the tensor equation (3.4.14) we can write :

$$\begin{aligned}
\mathbf{J}_o &= \int_m (\tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}} + \mathbf{r} \cdot \mathbf{r}\, 1)dm \\
&= \int_m \tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\, dm + 1\int_m -\frac{1}{2} tr\,[\tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\,]dm \qquad \text{[by (3.4.20)]} \\
&= \int_m \tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\, dm + \frac{1}{2} tr\,[-\int_m \tilde{\mathbf{r}} \cdot \tilde{\mathbf{r}}\, dm\,]1 \\
&= -\mathbf{I}_o + \frac{1}{2} tr\,[\mathbf{I}_o]1 \qquad \text{[by (4.3.5)]}
\end{aligned}$$

i.e., we have

$$\mathbf{J}_o = \frac{1}{2} tr\,[\mathbf{I}_o]1 - \mathbf{I}_o \tag{4.3.6}$$

Therefore, equation (4.3.6) provides the equivalence relationship between the two tensors, $\mathbf{J}_o$ and $\mathbf{I}_o$. Similarly, it can be shown that the equation

$$\mathbf{I}_o = tr\,[\mathbf{J}_o]1 - \mathbf{J}_o, \tag{4.3.7}$$

is also valid.

Now, as is often the case with most mathematical definitions, in practical applications we cannot use these definitions for the computation of the inertia tensor. In practice, the inertia tensor of a rigid body is computed experimentally. Moreover, even in experimental measurements, the direct computation of the rigid body inertia tensor about any point o other than the center of mass, is in general very difficult. Therefore, the body center of mass c is used when the inertia tensor of a body is evaluated. Then, in applications where the inertia tensor relative to points other than the center of mass is required, and the inertia tensor about the center of mass is known, the *parallel axis*

theorem is used. The parallel axis theorem for the inertia tensor which is defined by equation (4.3.3) is usually stated in the following form

$$\mathbf{I}_o = \mathbf{I}_c + m\left(\mathbf{r}_c \cdot \mathbf{r}_c \mathbf{1} - \mathbf{r}_c \mathbf{r}_c\right) \tag{4.3.8}$$

where $\mathbf{I}_c$ is the rigid body inertia tensor about its center of mass, $\mathbf{r}_c$ is the position vector of the center of mass relative to point o, and m is the mass of the body. Equation (4.3.8) can also be written in a compact tensor form as follows

$$\mathbf{I}_o = \mathbf{I}_c - m\,\tilde{\mathbf{r}}_c \cdot \tilde{\mathbf{r}}_c \tag{4.3.9}$$

where equation (4.3.9) is derived from (4.3.8) using equation (3.4.10).

The parallel axis theorem for the inertia tensor $\mathbf{I}_o$ is a basic theorem in rigid body dynamics and its proof can be found in any book on classical dynamics (e.g. [9,10]). Obviously, the parallel axis theorem is also valid for the pseudo-inertia tensor $\mathbf{J}_o$, which is defined by (4.3.5). Since the application of the parallel axis theorem for the tensor $\mathbf{J}_o$ is not well known, we provide here a formulation and a proof for it.

Theorem 4.3 (*parallel axis theorem*) : When the pseudo-inertia tensor $\mathbf{J}_c$ of a rigid body about its center of mass is known, then the pseudo-inertia tensor, $\mathbf{J}_o$, about any other point o, is given by

$$\mathbf{J}_o = \mathbf{J}_c + m\,\mathbf{r}_c \mathbf{r}_c \tag{4.3.10}$$

where $\mathbf{r}_c$ is the position vector of the center of mass relative to point o and m is the total mass of the body.

Proof : From the formulation of the parallel axis theorem in terms of the inertia tensor $\mathbf{I}_c$, i.e., from equation (4.3.8) we have that

$$tr\,[\mathbf{I}_o] = tr\,[\mathbf{I}_c] + (3\mathbf{r}_c \cdot \mathbf{r}_c - \mathbf{r}_c \cdot \mathbf{r}_c)m$$

or

$$tr\,[\mathbf{I}_o] = tr\,[\mathbf{I}_c] + 2\mathbf{r}_c \cdot \mathbf{r}_c\, m. \tag{4.3.11}$$

Now, using (4.3.8) and (4.3.11) we can rewrite (4.3.6) as follows

$$\mathbf{J}_o = \frac{1}{2}\left\{ tr\,[\mathbf{I}_c] + 2\mathbf{r}_c \cdot \mathbf{r}_c m \right\} \mathbf{1} - \mathbf{I}_c - (\mathbf{r}_c \cdot \mathbf{r}_c \mathbf{1} - \mathbf{r}_c \mathbf{r}_c) m$$

$$= \frac{1}{2} tr\,[\mathbf{I}_c]\mathbf{1} - \mathbf{I}_c + m\,\mathbf{r}_c \mathbf{r}_c .$$

Further, since equation (4.3.6) is valid for any point o, it is valid for the center of mass, i.e., we have

$$\mathbf{J}_c = \frac{1}{2} tr\,[\mathbf{I}_c]\mathbf{1} - \mathbf{I}_c \tag{4.3.12}$$

Therefore, by substituting equation (4.3.12) to the last expression for $\mathbf{J}_o$, we get equation (4.3.10). □

The inertia tensor, like any other tensor, is described relative to a coordinate system by a set of components which are known as the *moments of inertia* and *products of inertia*. These components define the coordinate matrix I_o for the tensor $\mathbf{I}_o$. If J_o is the coordinate matrix of the pseudo-inertia tensor $\mathbf{J}_o$ relative to the same coordinate system $\{\mathbf{e}\}$, then the equivalence which is established above by equation (4.3.6) or equation (4.3.7) between $\mathbf{J}_o$ and $\mathbf{I}_o$ leads us to a component-wise relationship between the coordinate matrices J_o and I_o. This component-wise relationship is expressed as follows :

$$J_o = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{12} & J_{22} & J_{23} \\ J_{13} & J_{23} & J_{33} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{-I_{11} + I_{22} + I_{33}}{2} & -I_{12} & -I_{13} \\ -I_{12} & \dfrac{I_{11} - I_{22} + I_{33}}{2} & -I_{23} \\ -I_{13} & -I_{23} & \dfrac{I_{11} + I_{22} - I_{33}}{2} \end{bmatrix} \tag{4.3.13}$$

or

$$I_o = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{12} & I_{22} & I_{23} \\ I_{13} & I_{23} & I_{33} \end{bmatrix}$$

$$= \begin{bmatrix} J_{22} + J_{33} & -J_{12} & -J_{13} \\ -J_{12} & J_{11} + J_{33} & -J_{23} \\ -J_{13} & -J_{23} & J_{11} + J_{22} \end{bmatrix} \tag{4.3.14}$$

Now, depending on the coordinate system, the components of the inertia tensor can be time-dependent or time-independent. In particular, relative to a coordinate system which is rigidly attached to the rigid body, the components of the inertia tensor are always time independent. However, when the rigid body is moving in space, the components of the inertia tensor relative to an inertial coordinate system will be time-dependent, and in this case it may be required to calculate their time derivatives. For example, in practical applications it is usually required to know the first time derivative of the inertia tensor $\mathbf{I}_c$ (or $\mathbf{J}_c$) relative to an inertial coordinate system. Therefore, in the following, we provide a simple formulation for this derivative.

Let us consider the coordinate system $\{ \mathbf{e} \}$ to be an inertial coordinate system. Also, let us consider a body coordinate system $\{ \mathbf{e}' \}$ whose orientation relative to the inertial coordinate system is described by the rotation tensor $\mathbf{R}$ which is assumed to be a continuous differentiable function of time. Moreover, let us denote by $\mathbf{I}_c$ and $\mathbf{I}_c'$ the rigid body inertia tensors relative to the inertial and body coordinate systems, respectively. From the foregoing, it is obvious that the inertia tensor $\mathbf{I}_c'$ is time independent as opposed to the inertia tensor $\mathbf{I}_c$ which is time dependent. We express the time dependence of $\mathbf{I}_c$ by writing

$$\mathbf{I}_c = \mathbf{R}\,\mathbf{I}_c'\,\mathbf{R}^T. \tag{4.3.15}$$

Equation (4.3.15) allows us to derive the absolute time derivative (i.e., the time derivative in an inertial reference frame) of the inertia tensor $\mathbf{I}_c$ of a rigid body in a simple and concise manner as follows

$$\begin{aligned} \dot{\mathbf{I}}_c &= \dot{\mathbf{R}}\,\mathbf{I}_c'\,\mathbf{R}^T + \mathbf{R}\,\mathbf{I}_c'\,\dot{\mathbf{R}}^T \\ &= \tilde{\omega}\mathbf{R}\,\mathbf{I}_c'\,\mathbf{R}^T + \mathbf{R}\,\mathbf{I}_c'\,\mathbf{R}^T\tilde{\omega}^T \qquad \text{[by (4.2.5)]} \\ &= \tilde{\omega}\mathbf{I}_c + \mathbf{I}_c\tilde{\omega}^T \end{aligned}$$

or, since $\tilde{\omega}$ is skew-symmetric, we can write

$$\dot{\mathbf{I}}_c = \tilde{\omega}\mathbf{I}_c - \mathbf{I}_c\tilde{\omega}. \tag{4.3.16}$$

Equation (4.3.16), is also valid if we consider the inertia tensor of the rigid body about any other point o instead of that of the center of mass c. We can see this as follows :

Using the parallel axis theorem (equation (4.3.9)), we have

$$\dot{\mathbf{I}}_o = \dot{\mathbf{I}}_c - m\left(\dot{\tilde{\mathbf{r}}}_c\tilde{\mathbf{r}}_c + \tilde{\mathbf{r}}_c\dot{\tilde{\mathbf{r}}}_c\right)$$

which, by equation (4.2.21), can be written as

$$\dot{\mathbf{I}}_o = \dot{\mathbf{I}}_c - m\left(dual\,(\tilde{\omega}\mathbf{r}_c)\tilde{\mathbf{r}}_c + \tilde{\mathbf{r}}_c\,dual\,(\tilde{\omega}\mathbf{r}_c)\right).$$

Now, using equation (3.4.15), we get after a few manipulations

$$\begin{aligned}\dot{\mathbf{I}}_o &= \dot{\mathbf{I}}_c - \tilde{\omega}\left(m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right) + \left(m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right)\tilde{\omega} \\ &= \tilde{\omega}\left(\mathbf{I}_c - m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right) - \left(\mathbf{I}_c - m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right)\tilde{\omega}\end{aligned}$$

or, finally

$$\dot{\mathbf{I}}_o = \tilde{\omega}\mathbf{I}_o - \mathbf{I}_o\tilde{\omega}. \tag{4.3.17}$$

Equations (4.3.16) and (4.3.17) are also valid if we use the pseudo-inertia tensor $\mathbf{J}_o$ instead of $\mathbf{I}_o$. To see this, we need only to notice that

$$\dot{\mathbf{I}}_o = -\dot{\mathbf{J}}_o \tag{4.3.18}$$

for any point o. Equation (4.3.18) follows from either (4.3.6) or (4.3.7), since the trace of a tensor is a scalar invariant (see section 3.3.3) and thus is time independent. Then a simple substitution in equation (4.3.17) shows that the absolute derivative of $\mathbf{J}_o$ is given by

$$\dot{\mathbf{J}}_o = \tilde{\omega}\mathbf{J}_o - \mathbf{J}_o\tilde{\omega}. \tag{4.3.19}$$

In the following, we shall use the two inertia tensors $\mathbf{I}$ and $\mathbf{J}$ to compute other basic physical quantities in rigid body motion such as the angular momentum and the external (or resultant) torque.

4.3.2 The Angular Momentum Tensor

One of the most important physical quantities in rigid body dynamics is that of the *angular momentum* or *absolute moment of momentum.* In the classical vectorial treatment of rigid body dynamics the angular momentum is represented by a vector which is defined [9] by the equation

$$\mathbf{L}_o = \mathbf{s} \times \left(\dot{\mathbf{s}} + \boldsymbol{\omega} \times \mathbf{r}_c\right) m + \mathbf{r}_c \times \dot{\mathbf{s}}\, m + \mathbf{I}_{o'} \cdot \boldsymbol{\omega} \tag{4.3.20}$$

where o is the origin of the inertial coordinate system, o' is a point fixed on the rigid body, $\mathbf{s}$ is the position vector of o' relative to o, $\mathbf{r}_c$ is the position vector of the center of mass relative to o', $\mathbf{I}_{o'}$ is the rigid body inertia tensor about the point o' and $\boldsymbol{\omega}$ is the vector of the angular velocity. The expression for $\mathbf{L}_o$ in equation (4.3.20) becomes particularly simple if either the body fixed point o' is also fixed in inertial space ($\dot{\mathbf{s}} = 0$) or the center of mass is used as the reference point o' ($\mathbf{r}_c = 0$). In both cases the term $\mathbf{r}_c \times \dot{\mathbf{s}}\, m$ vanishes. The first term then represents the angular momentum with respect to o due to translation of the center of mass and the last term represents the angular momentum caused by the rotation of the rigid body. In the following, we shall assume that there is no translational motion ($\dot{\mathbf{s}} = 0$) and so equation (4.3.20) takes the form

$$\mathbf{L}_o = \mathbf{s} \times \left(\boldsymbol{\omega} \times \mathbf{r}_c\right) m + \mathbf{I}_{o'} \cdot \boldsymbol{\omega}. \tag{4.3.21}$$

Moreover, we shall assume that the inertial coordinate system has its origin at the point o' ($\mathbf{s} = 0$) and in this case we shall write

$$\mathbf{L}_o = \mathbf{I}_o \cdot \boldsymbol{\omega} \equiv \mathbf{I}_o \boldsymbol{\omega}. \tag{4.3.22}$$

Obviously, when the center of rotation is at the center of mass, equation (4.3.22) becomes

$$\mathbf{L}_c = \mathbf{I}_c \boldsymbol{\omega}. \tag{4.3.23}$$

However, even when the center of rotation is different from the center of mass, it is use-

ful to write $\mathbf{L}_o$ in terms of $\mathbf{L}_c$. An expression of $\mathbf{L}_o$ in terms of $\mathbf{L}_c$ can be easily derived by using Cartesian tensor analysis, as follows :

Using the parallel axis theorem, equation (4.3.22) can be modified as shown

$$\begin{aligned}\mathbf{L}_o &= \left(\mathbf{I}_c - m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right)\omega \\ &= \mathbf{L}_c - m\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\omega \\ &= \mathbf{L}_c + m\tilde{\mathbf{r}}_c\tilde{\omega}\mathbf{r}_c \\ &= \mathbf{L}_c + m\tilde{\mathbf{r}}_c\dot{\mathbf{r}}_c. \end{aligned} \tag{4.3.24}$$

In a pure vector notation equation (4.3.24) takes the form

$$\mathbf{L}_o = \mathbf{L}_c + \mathbf{r}_c \times \dot{\mathbf{r}}_c m \tag{4.3.25}$$

The angular momentum can also be defined in terms of the pseudo-inertia tensor $\mathbf{J}_o$. To see this we need only to substitute $\mathbf{I}_o$ in equation (4.3.22) by $\mathbf{J}_o$. For this, we use equation (4.3.7) and get

$$\mathbf{L}_o = -\mathbf{J}_o\omega + tr\,[\mathbf{J}_o]\omega. \tag{4.3.26}$$

Besides its vector description, the angular momentum can also be described [1] by a second order skew-symmetric Cartesian tensor. To see this, we need only apply the dual operator on the angular momentum vector $\mathbf{L}_o$. This gives a dual skew-symmetric tensor $\tilde{\mathbf{L}}_o$ which we express by writing

$$\tilde{\mathbf{L}}_o = dual(\mathbf{L}_o). \tag{4.3.27}$$

We refer to the dual skew-symmetric tensor $\tilde{\mathbf{L}}_o$ as the *angular momentum tensor* about the point o.

The dual operator provides an indirect definition for the angular momentum tensor. However, as the following theorem shows, it is possible to define the angular momentum tensor $\tilde{\mathbf{L}}_o$ directly in terms of the inertia tensor $\mathbf{I}_o$ and the angular velocity tensor $\tilde{\omega}$, i.e., without the need to first compute the angular momentum vector.

Theorem 4.4 : The angular momentum tensor of a rotating rigid body about a point

o, satisfies the equation

$$\tilde{L}_o = (\tilde{\omega} I_o)^T - (\tilde{\omega} I_o) + tr[I_o]\tilde{\omega} \tag{4.3.28}$$

where I_o is the inertia tensor of the rigid body about the center of rotation o and $\tilde{\omega}$ is the angular velocity tensor.

Proof: Using equation (4.3.22) we can write equation (4.3.27) as

$$\tilde{L}_o = dual(I_o\tilde{\omega}).$$

Further, since the inertia tensor I_o is symmetric, by using Proposition (3.11) we get

$$\begin{aligned}\tilde{L}_o &= -(I_o\tilde{\omega} + \tilde{\omega} I_o) + tr[I_o]\tilde{\omega} \\ &= I_o\tilde{\omega}^T - \tilde{\omega} I_o + tr[I_o]\tilde{\omega} \\ &= (\tilde{\omega} I_o)^T - \tilde{\omega} I_o + tr[I_o]\tilde{\omega}\end{aligned}$$

and this completes the proof. □

Theorem 4.4 can also be written in terms of the pseudo-inertia tensor J_o. First we notice that from equation (4.3.7) we have

$$tr[I_o] = 2tr[J_o]. \tag{4.3.29}$$

Therefore, if we substitute equation (4.3.7) and (4.3.29) in equation (4.3.28) we have

$$\begin{aligned}\tilde{L}_o &= \left[tr[J_o]\tilde{\omega} - \tilde{\omega} J_o\right]^T - \left[tr[J_o]\tilde{\omega} - \tilde{\omega} J_o\right] + 2tr[J_o]\tilde{\omega} \\ &= -tr[J_o]\tilde{\omega} + J_o\tilde{\omega} - tr[J_o]\tilde{\omega} + \tilde{\omega} J_o + 2tr[J_o]\tilde{\omega} \\ &= J_o\tilde{\omega} + \tilde{\omega} J_o\end{aligned}$$

or, finally

$$\tilde{L}_o = J_o\tilde{\omega} - \left[J_o\tilde{\omega}\right]^T. \tag{4.3.30}$$

Obviously, when the rigid body is rotating about its center of mass, the angular momentum tensor is defined by the equation

$$\tilde{L}_c = (\tilde{\omega} I_c)^T - (\tilde{\omega} I_c) + tr[I_c]\tilde{\omega} \tag{4.3.31}$$

or the equation

$$\dot{\mathbf{L}}_C = \mathbf{J}_C\tilde{\omega} - \left[\mathbf{J}_C\tilde{\omega}\right]^T. \tag{4.3.32}$$

As we can see from equations (4.3.22) and (4.3.26), the angular momentum vector has a simpler expression when it is written in terms of the inertia tensor $\mathbf{I}_O$. But this is not true for the angular momentum tensor. Equations (4.3.28) and (4.3.30) show that the angular momentum tensor has a simpler expression when it is expressed in terms of the pseudo-inertia tensor $\mathbf{J}_O$.

In the following, we shall use the angular momentum, in its vector or tensor representation, to describe the dynamic behavior of a rotating rigid body.

4.3.3 The Torque Tensor

As is well known [9], the absolute time derivative of angular momentum defines the *resultant torque* or *moment of force*. This derivative expresses the basic law governing the rotational motion of a rigid body and in vector form is written as

$$\mathbf{M}_O = \dot{\mathbf{L}}_O. \tag{4.3.33}$$

Obviously, when the point of rotation is the center of mass of the rigid body, equation (4.3.33) takes the form

$$\mathbf{M}_C = \dot{\mathbf{L}}_C. \tag{4.3.34}$$

To show that equation (4.3.34) is equivalent to equation (4.3.2), we proceed as follows

$$\begin{aligned}
\mathbf{M}_C &= \frac{d}{dt}(\mathbf{I}_C\cdot\omega) \\
&= \dot{\mathbf{I}}_C\cdot\omega + \mathbf{I}_C\cdot\dot{\omega}) \\
&= \left(\tilde{\omega}\mathbf{I}_C - \mathbf{I}_C\tilde{\omega}\right)\cdot\omega + \mathbf{I}_C\cdot\dot{\omega} \qquad \text{[by (4.3.16)]} \\
&= \mathbf{I}_C\cdot\dot{\omega} + \tilde{\omega}\mathbf{I}_C\cdot\omega \qquad \text{[by (3.4.7)]} \\
&= \mathbf{I}_C\cdot\dot{\omega} + \omega\times\mathbf{I}_C\cdot\omega
\end{aligned}$$

Sometimes, in the literature on classical dynamics, equation (4.3.2) or equation (4.3.34) is referred to as the *generalized Euler equation* for rigid body rotational motion. Here, we shall refer to equation (4.3.2) as the *vector* formulation of the generalized Euler equation.

Now, as in the case of $\mathbf{M}_c$, it can be shown that the vector $\mathbf{M}_o$ satisfies the equation

$$\mathbf{M}_o = \mathbf{I}_o \cdot \dot{\omega} \cdot \omega \mathbf{I}_o \cdot \omega \tag{4.3.35}$$

when the center of rotation, i.e., the point o, is any point other than the center of mass. Moreover, the torque vector $\mathbf{M}_o$ can be written in terms of the torque vector $\mathbf{M}_c$ as follows

$$\mathbf{M}_o = \mathbf{M}_c + \mathbf{r}_c \times \mathbf{F}_c \tag{4.3.36}$$

where $\mathbf{F}_c$ is the total force caused at the center of the mass of the rigid body due to its rotational motion. To see that equation (4.3.36) is equivalent to equation (4.3.35), we write from equation (4.3.36),

$$\begin{aligned}
\mathbf{M}_o &= \mathbf{M}_c + m\,\tilde{\mathbf{r}}_c\ddot{\mathbf{r}}_c \\
&= \mathbf{M}_c + m\,\tilde{\mathbf{r}}_c\Omega\mathbf{r}_c \\
&= \mathbf{M}_c + m\left(\tilde{\mathbf{r}}_c(\dot{\tilde{\omega}} + \tilde{\omega}\tilde{\omega})\mathbf{r}_c\right) \\
&= \mathbf{M}_c - m\left(\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\dot{\omega} + \tilde{\mathbf{r}}_c\tilde{\omega}\tilde{\mathbf{r}}_c\omega\right)
\end{aligned} \tag{4.3.37}$$

where equation (3.4.4) has been used in the last step. Now, from equation (3.4.22) we have

$$\tilde{\mathbf{r}}_c\tilde{\omega}\tilde{\mathbf{r}}_c = \tilde{\omega}\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c + \tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\tilde{\omega} - \frac{1}{2}tr\,[\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c]\tilde{\omega}$$

and since $\tilde{\omega}\omega = 0$, equation (4.3.37) becomes

$$\begin{aligned}
\mathbf{M}_o &= \mathbf{M}_c - m\left(\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\dot{\omega} + \tilde{\omega}\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\omega\right) \\
&= \left(\mathbf{I}_c - m\,\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right)\dot{\omega} + \tilde{\omega}\left(\mathbf{I}_c - m\,\tilde{\mathbf{r}}_c\tilde{\mathbf{r}}_c\right)\omega. \qquad \text{[by (4.3.2)]}
\end{aligned}$$

Finally, using the parallel axis theorem, we can see that the last equation is equivalent to equation (4.3.35).

Equation (4.3.35) has been stated in terms of the inertia tensor $\mathbf{I}_o$. If we use equation (4.3.7) to substitute for $\mathbf{I}_o$ in terms of $\mathbf{J}_o$, equation (4.3.35) becomes

$$\mathbf{M}_o = -\left(\mathbf{J}_o \cdot \dot{\omega} + \tilde{\omega}\mathbf{J}_o \cdot \omega\right) + tr\,[\mathbf{J}_o]\dot{\omega}. \tag{4.3.38}$$

Similarly, for $\mathbf{M}_c$ we can write

$$\mathbf{M}_c = -\left(\mathbf{J}_c \cdot \dot{\omega} + \tilde{\omega}\mathbf{J}_c \cdot \omega\right) + tr\,[\mathbf{J}_c]\dot{\omega}. \tag{4.3.39}$$

Now, as with angular momentum, the torque can be described by a second order skew-symmetric Cartesian tensor, which we shall refer to as the *torque tensor* . Using the dual operator we define the torque tensor as follows

$$\tilde{\mathbf{M}}_o = dual\,(\mathbf{M}_o). \tag{4.3.40}$$

This definition for the torque tensor requires the computation of the torque vector $\mathbf{M}_o$. Another definition for the torque tensor in terms of the inertia and the angular acceleration tensors is given by the following theorem.

Theorem 4.5 : The torque tensor about the center of rotation o, is defined as the absolute time derivative of the angular momentum tensor about the point o and satisfies the equation

$$\tilde{\mathbf{M}}_o = \left(\Omega\mathbf{I}_o\right)^T - \Omega\mathbf{I}_o + tr\,[\mathbf{I}_o]\dot{\tilde{\omega}} \tag{4.3.41}$$

where Ω is the angular acceleration tensor of the rotation and $\mathbf{I}_o$ is the rigid body inertia tensor about the point o.

Proof : By definition, we have

$$\tilde{\mathbf{M}}_o = \frac{d}{dt}\mathbf{L}_o.$$

Further, using equation (4.3.28), we get

$$\tilde{\mathbf{M}}_o = \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \omega\dot{\mathbf{I}}_o\right]^T - \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \tilde{\omega}\dot{\mathbf{I}}_o\right]^T + tr\,[\mathbf{I}_o]\dot{\tilde{\omega}}\ ,$$

since the time derivative of the scalar invariant $tr\,[\mathbf{I}_o]$ is zero. Now, using equation (4.3.17), we have

$$\tilde{\mathbf{M}}_o = \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \tilde{\omega}\tilde{\omega}\mathbf{I}_o - \tilde{\omega}\mathbf{I}_o\tilde{\omega}\right]^T - \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \tilde{\omega}\tilde{\omega}\mathbf{I}_o - \tilde{\omega}\mathbf{I}_o\tilde{\omega}\right] + tr\,[\mathbf{I}_o]\dot{\tilde{\omega}}$$

$$= \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \tilde{\omega}\tilde{\omega}\mathbf{I}_o\right]^T - \left[\dot{\tilde{\omega}}\,\mathbf{I}_o + \tilde{\omega}\tilde{\omega}\mathbf{I}_o\right] + tr\,[\mathbf{I}_o]\dot{\tilde{\omega}}$$

$$= \left[\Omega \mathbf{I}_o\right]^T - \Omega \mathbf{I}_o + tr\,[\mathbf{I}_o]\tilde{\omega} \qquad \square$$

The torque tensor $\tilde{\mathbf{M}}_o$ can also be defined in terms of the pseudo-inertia tensor $\mathbf{J}_o$ and the angular acceleration tensor Ω. To see this, we can consider the time derivative of equation (4.3.30) or use equation (4.3.7) to substitute for $\mathbf{I}_o$ in equation (4.3.41). In both cases after a few manipulations we arrive at the following equation

$$\tilde{\mathbf{M}}_o = \Omega \mathbf{J}_o - \left(\Omega \mathbf{J}_o\right)^T. \tag{4.3.42}$$

Obviously, when the rigid body is rotated about its center of mass equations (4.3.41) and (4.3.42) are written as

$$\tilde{\mathbf{M}}_c = \left[\Omega \mathbf{I}_c\right]^T - \Omega \mathbf{I}_c + tr\,[\mathbf{I}_c]\tilde{\omega} \tag{4.3.43}$$

and

$$\tilde{\mathbf{M}}_c = \Omega \mathbf{J}_c - \left(\Omega \mathbf{J}_c\right)^T, \tag{4.3.44}$$

respectively. We shall refer to equation (4.3.44) as the *tensor* formulation of the generalized Euler equation of a rigid body rotational motion.

As we can see from equations (4.3.2) and (4.3.39) the torque vector $\mathbf{M}_c$ has a simpler expression when it is defined in terms of the inertia tensor $\mathbf{I}_c$. But, as in the case of the angular momentum tensor, the pseudo-inertia tensor $\mathbf{J}_o$ leads to a simpler equation for the definition of the torque tensor $\tilde{\mathbf{M}}_c$. This implies that for a vector formulation of the equations of rotational rigid body motion, the proper definition for the inertia tensor is given by equation (4.3.3). But, when a tensor formulation for the equations of rotational rigid body motion is required, then the proper definition for the inertia tensor is given by equation (4.3.4).

From the foregoing, to describe rotational rigid body motion within the Newtonian formulation, we can use either vector analysis or Cartesian tensor analysis. The two approaches are equivalent in the sense that the torque vector $\mathbf{M}_c$ is the dual vector or vector invariant of the torque tensor $\tilde{\mathbf{M}}_c$. Therefore, we can use either approach for

describing the resultant torque of a rigid body motion. But as we shall see in the following, in practical applications a tensor description for the resultant torque is to be preferred since it is computationally more efficient.

4.3.4 Computational Considerations

In the following, we shall assume that the angular velocity vector, ω, the angular acceleration vector, $\dot{\omega}$, and the angular acceleration tensor Ω are available and we shall examine the computational cost of computing the vectors $\mathbf{F}_C$ and $\mathbf{M}_C$ which describe a rigid body motion.

To compute the vector $\mathbf{F}_C$ we need to evaluate equation (4.3.1). It is obvious that the computational burden of evaluating this equation results mainly from the computation of the vector $\ddot{\mathbf{r}}_C$. From a computational point of view, most of the times computing the vector $\ddot{\mathbf{r}}_C$ is similar of computing the vector $\ddot{\mathbf{r}}_1$ which is defined by either equation (4.2.20) or equation (4.2.24). Thus, to compute the vector $\ddot{\mathbf{r}}_C$ we can use equation (4.2.20) or equation (4.2.24). In the latter case we need to perform three vector cross product operations and two vector additions and this requires a total of 18 scalar multiplications and 15 scalar additions. In the former case we need to perform a matrix-vector multiplication and a vector addition and this requires a total of 9 scalar multiplications and 9 scalar additions. To compute the torque vector $\mathbf{M}_C$ we can use equation (4.3.2) or we can use equation (4.3.44) which computes the torque tensor $\mathbf{M}_C$ and then from that skew-symmetric tensor we can extract the vector $\mathbf{M}_C$ by using the correspondence (3.3.31). In the first approach, i.e., using equation (4.3.2), we need to perform two matrix-vector multiplications, a vector cross product operation and a vector addition which requires a total of 24 scalar multiplications and 18 scalar additions. In the second approach, we can evaluate the torque vector $\mathbf{M}_C$ with only 15 scalar multiplications and 15 scalar additions. This is so, because there is no need to compute the complete matrix-matrix multiplication, which is involved in equation (4.3.44), since the tensor

$\tilde{M}_c$ is skew-symmetric, and the extraction of its dual vector requires no computations.

From the foregoing, when vectors are used to describe the Newtonian formulation of rigid body motion, i.e., equations (4.3.1), (4.2.24) and (4.3.2) to compute the vectors F_c anf M_c we require a total of 45 scalar multiplications and 33 scalar additions. On the other hand, when Cartesian tensors are used to describe the Newtonian formulation of rigid body motion, i.e., equations (4.3.1), (4.2.20), (4.3.44) and (3.3.31) we require a total of only 27 scalar multiplications and 24 scalar additions for evaluating the same vectors. Therefore, the tensorial treatment of rigid body motion which is presented in this chapter is reducing the computational cost of evaluating the equations of rigid body motion considerably. This, obviously, has very important consequences for many practical problems of mechanics where rigid body dynamics plays an important role. In the following chapters, we shall use this tensorial treatment of rigid body motion to solve in a computationally efficient manner the three problems of robot dynamics, namely, the problem of inverse dynamics, the problem of forward dynamics, and the linearization of the equations of motion of rigid-links open-chain robot manipulators.

4.4 REFERENCES

[1] H. Jeffreys, *Cartesian Tensors*, Cambridge University Press, Cambridge, 1961.

[2] O. Bottema and B. Roth, *Theoretical Kinematics*, North-Holland Publishing Co., Amsterdam, 1978.

[3] E. J. Konopinski, *Classical Descriptions of Motion*, W. H. Freeman and Company, San Francisco, 1969.

[4] J. Angeles, *Spatial Kinematic Chains : Analysis, Synthesis, Optimization*, Springer-Verlag, New York, 1982.

[5] J. Angeles, *Rational Kinematics*, Springer-Verlag, New York, 1989.

[6] J. S. Beggs, *Kinematics*, Hemisphere Publishing Corporation, Washington, 1983.

[7] T. Crouch, *Matrix Methods Applied to Engineering Rigid Body Mechanics*, Pergamon Press, Oxford, 1981.

[8] S. F. Borg, *Matrix-Tensor Methods in Continuum Mechanics*, D. Van Nostrand Company, Princeton, New Jersey, 1963.

[9] I. J. Wittenburg, *Dynamics of Systems of Rigid Bodies*, B. G. Teubner, Stuttgart, 1977.

[10] H. Goldstein, *Classical Mechanics*. 2nd Ed., Addison-Wesley, Reading, MA : 1980.

[11] J. B. Marion, *Classical Dynamics of Particles and Systems*, 2nd Ed., Academic Press, New York, 1970.

[12] S. N. Rasband, *Dynamics*, John Wiley & Sons, New York, 1983.

[13] L. A. Pars, *A Treatise on Analytical Dynamics*, Heinemann, London, 1965.

[14] C. A. Balafoutis, R. V. Patel and P. Misra, "Efficient Modeling and Computation of Manipulator Dynamics Using Orthogonal Cartesian Tensors" *IEEE J. Robotics and Automation*, pp 665-676, Vol. 4, No. 6, 1988.

[15] C. A. Balafoutis, R. V. Patel and P. Misra, "A Cartesian Tensor Approach for Fast Computation of Manipulator Dynamics", *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1348-1353, Philadelphia, April 24-29, 1988.

[16] C. G. Atkeson, C. H. An and J. M. Hollerbach, "Rigid Body Load Identification for Manipulators", *24th IEEE Conf. on Decision and Control* , pp. 996-1002, December 1985.

[17] J. Casey and V. C. Lam, "A Tensor Method for the Kinematical Analysis of Systems of Rigid Bodies", *Mechanism and Machine Theory*, pp 87-97, Vol. 21, No. 1, 1986.

[18] W. M. Silver, "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators", *Int. Journal of Robotics Research*, pp. 60-70, Vol. 1, No. 2, 1982.

[19] C. A. Balafoutis, R. V. Patel and J. Angeles, "A Comparative Study of Newton-Euler, Euler-Lagrange and Kane's Formulation for Robot Manipulator Dynamics", *Robotics and Manufacturing : Recent Trends in Research, Education and Applications*, M. Jamsidi, J. Y. S. Luh, H. Seraji, and G. P. Starr, *Eds.*,, ASME Press, New York, 1988.

CHAPTER V

INVERSE DYNAMICS OF RIGID-LINK OPEN-CHAIN ROBOT MANIPULATORS

5.1 INTRODUCTION

Inverse dynamics is the calculation of the forces and/or torques required at a robot's joints in order to produce a given motion trajectory consisting of a set of joint positions, velocities and accelerations. The principal uses of inverse dynamics are in robot control and trajectory planning. In control applications it is usually incorporated as an element in the feedback or feedforward path to convert positions, velocities and accelerations computed according to some desired trajectory into the joint generalized forces which will achieve those accelerations (e.g. see [1-4]). In trajectory planning, inverse dynamics can be used to check or ensure that a proposed trajectory can be executed without exceeding the actuators' limits [5-7]. Also, using time-scaling properties of inverse dynamics we can facilitate minimum-time (or near minimum-time) trajectory planning [8]. Moreover, inverse dynamics are also taken into consideration in defining manipulability measures of robot arms. (The manipulability is usually expressed as a quantitative measure of robot arm manipulating ability in positioning and orienting its end-effector. [9]). Finally, as we shall see in the next chapter, inverse dynamics is also used as a building block for constructing forward dynamics algorithms which are useful in performing dynamic simulations of robot arms.

Mathematically, the inverse dynamics problem (IDP) can be described by an equation of the form

$$\tau = \mathbf{f}\ (\mathbf{q}\ ,\ \dot{\mathbf{q}}\ ,\ \ddot{\mathbf{q}}\ ,\ \text{manipulator parameters}\ ) \qquad (5.1.1)$$

where τ is the vector of the unknown generalized forces, $(\mathbf{q}\ ,\ \dot{\mathbf{q}}\ ,\ \ddot{\mathbf{q}}\ )$ denotes a given manipulator trajectory and the "manipulator parameters" are all those parameters

which characterize the particular geometry and dynamics of a robot manipulator. Equation (5.1.1) is referred to as the *dynamic model* or the *dynamic equation* of a robot manipulator.

To derive equation (5.1.1), we can use well established procedures from classical mechanics such as those based on the equations of *Newton* and *Euler*, *Euler* and *Lagrange*, *Kane*, and others. Intuitively, one would expect that all the different approaches of formulating this dynamic model should result in the same or equivalent equations. This, in fact, is true. However, it is understandable that the choice of a particular procedure is important because it determines the nature of the analysis and the amount of effort needed to obtain these equations. But, otherwise, it is not important which procedure we choose because, as we shall see in this Chapter, all procedures can be formulated to lead to the same algorithm for solving the IDP.

It is known, (from experience) that independent of which procedure we use to derive the dynamic model of a *simple* mechanical system, the vector function $\mathbf{f}$ is usually simple and thus it is possible to express it explicitly in terms of the system kinematic parameters (generalized position, velocity and acceleration). But, unfortunately, this is extremely difficult for mechanical systems of the complexity of a robot manipulator for which the vector function $\mathbf{f}$ is known to be highly nonlinear and coupled. Although, based on symbolic manipulations, some explicit formulations for the function $\mathbf{f}$ have been proposed [10,11], usually the function $\mathbf{f}$ is obtained via a structured algorithm, i.e., the dynamic model of a robot manipulator is usually evaluated in stages. The results of each stage are a set of values for intermediate variables which are used in subsequent stages to evaluate other variables (or expressions). Depending on what we consider as intermediate variables and how we represent them, we can derive different algorithms for evaluating the dynamic model of a robot manipulator.

The computational complexity for different algorithms varies enormously and these differences are accounted for by the amount of calculations involved in evaluating the

equations of motion via a prescribed set of intermediate variables. The key to efficient dynamics calculation is to find a set of common sub-expressions which will effectively be the intermediate variables to be calculated by the algorithm. This eliminates most of the repetition inherent in the equations of motion. Moreover, the representation of the intermediate variables is also important because, based on their representation other subsequent intermediate quantities may be formulated more efficiently.

As for the general structure of these algorithms it has been found that by formulating the motion dynamics *recursively* in terms of recurrence relations, we can produce computationally efficient algorithms for solving the IDP. In this approach the task is first broken down into a number of partially ordered steps. In each step a number of intermediate variables are evaluated. The value of each variable is determined by the application of a formula to each link in turn. Where possible and appropriate, the formula defines the quantity of interest for the link in question in terms of that quantity for one or more of the link's immediate neighbors, and in this case the formula is known as a *recurrence relation*. At the end, these steps are stated together to form a recursive algorithm which solves the problem of inverse dynamics.

The outline of this Chapter is as follows : Section 5.2 contains a review of existing methods for solving the IDP; some "classical" algorithms which have been derived from these methods are presented. Also, some observations are made about various issues concerning the computational efficiency of these algorithms. In Section 5.3, two new algorithms for solving the IDP are presented which are based on Cartesian tensors and use two different modeling schemes. The computational complexity of these algorithms is analyzed and compared with that of other algorithms in the literature. In Section 5.4 it is demonstrated that the computational efficiency of an algorithm which solves inverse dynamics, is actually independent of the particular formulation from classical mechanics that is used for its derivation. Finally, Section 5.5 concludes this chapter.

5.2 Previous Results and General Observations on Inverse Dynamics

An extensive literature exists on the subject of rigid-link manipulator dynamics in general and on inverse manipulator dynamics in particular. In this section, basic contributions on this subject will be mentioned and the "classical" computational algorithms for solving inverse dynamics will be presented.

The usual methods for computing inverse dynamics are classified with respect to the laws of mechanics on the basis of which the equations of motion are formed. Thus, one may distinguish methods based on *Euler-Lagrange, Newton-Euler, Kane's, d'Alembert's,* and other equations. Among them, methods based on Euler-Lagrange and Newton-Euler equations have gained early popularity and, as a consequence, there are many algorithms available today that have been derived using these equations to obtain inverse dynamics. The Euler-Lagrange equations are popular because they are conceptually simple and the methods based on them can lead easily to *closed-form* algorithms which are attractive from both the dynamic modeling and control points of view. On the other hand, the Newton-Euler equations became popular because they led early to computationally efficient recursive algorithms which can be used for real-time control applications and simulation. But besides these two commonly used formulations, in the past few years researchers have also successfully used Kane's dynamical equations in deriving efficient recursive algorithms for computing inverse dynamics.

In the following, we present a brief survey of the methods for solving the IDP based on the Euler-Lagrange, Newton-Euler and Kane's equations.

5.2.1 Formulations Based on Euler-Lagrange Equations

In the Lagrangian approach, the *Lagrangian* of the manipulator is expressed in terms of joint positions and velocities (which are the generalized coordinates and their derivatives). This expression is substituted into the Euler-Lagrange equation which is

expanded by symbolic differentiation to give the generalized forces (joint forces and torques) in terms of the generalized joint positions, velocities and accelerations. The first result in this class of formulations was proposed in 1965 by J. J. Uicker for the study of the dynamical behavior of joint-connected systems of arbitrary structure and with an arbitrary number of closed-loop kinematic chains [12]. This method was later modified by Kahn [13] to include open-loop mechanisms. The Uicker/Kahn method leads to a closed-form algorithm which, as estimated in [14], has $O(n^4)$ computational complexity, where n is the number of links or the number of degrees-of-freedom for a serial-type manipulator to which the algorithm is applied. In particular, for a 6 degrees-of-freedom manipulator the evaluation of the generalized forces at a trajectory point using the Uicker/Kahn algorithm requires 66271 scalar multiplications and 51548 scalar additions. This led some researchers to consider simplifications in the dynamical equations, namely, ignoring the Coriolis and centrifugal forces (Paul [15], Bejczy [16]). However, since simplifications of this nature are justifiable only for slow movements of the manipulator [17], this approach was soon abandoned. Another approach to reduce the computational complexity was consider by Albus [18] and Raibert [19]. They proposed a table-look-up method, whereby all the configuration dependent terms in the dynamical equations were computed in advance and tabulated for discrete points on the trajectory. Because of the large memory requirements involved in this approach, Horn and Raibert [20] proposed yet another method in which only the position dependent terms where tabulated. However, besides memory requirements, tabular methods have other serious limitations such as poor accuracy of the trajectory, due to interpolation between the stored discrete points. Moreover, the requirement that the trajectory has to be known in advance makes such methods unattractive because this obviously prevents their applicability to robots working in a dynamically changing environment.

It was soon realized that inverse dynamics for open-loop manipulators with simple kinematic chain structure, could be analyzed more effectively using recursive methods.

Waters [21] was the first to notice that the generalized forces, which in terms of homogeneous coordinates are defined [16] by the equation

$$\tau_i = \sum_{j=i}^{n}\left\{\sum_{k=1}^{j}\left[tr\left(\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}_{o_j}^{j}\frac{\partial \mathbf{W}_j^T}{\partial q_i}\right)\right]\ddot{q}_k + \sum_{k=1}^{j}\sum_{l=1}^{j}\left[tr\left(\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}_{o_j}^{j}\frac{\partial^2 \mathbf{W}_j^T}{\partial q_k\,\partial q_l}\right)\dot{q}_k\,\dot{q}_l\right] - m_j\,\mathbf{g}^T\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{r}_{j,j}^{j}\right\}$$

$$i = 1, 2, \cdots, n, \qquad (5.2.2)$$

could be written in the following form

$$\tau_i = \sum_{j=i}^{n}\left[tr\left(\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}_{o_j}^{j}\ddot{\mathbf{W}}_j^T\right) - m_j\,\mathbf{g}^T\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{r}_{j,j}^{j}\right], \quad i = 1, \cdots, n. \qquad (5.2.3)$$

Equation (5.2.3) allows one to take advantage of several kinematic recurrence relations for efficiently computing the homogeneous transformations and their time derivatives. Thus, Waters proposed a recursive algorithm for solving only inverse dynamics which has computational complexity $o(n^2)$. In terms of scalar multiplications and additions, for $n = 6$, Waters' algorithm requires [14] 7051 multiplications and 5652 additions. Later on, by introducing dynamic recurrence relations, Hollerbach [14] proposed modifications in Water's algorithm which led to a complete recursive algorithm for implementing equation (5.2.3). Hollerbach's recursive algorithm, formulated in the notation of this thesis, can be stated as follows :

ALGORITHM 5.1

Step 0: Initialization

$$\mathbf{W}_0 = \mathbf{I}\ , \quad \dot{\mathbf{W}}_0 = 0\ , \quad \ddot{\mathbf{W}}_0 = 0\ , \quad \mathbf{A}_{n+1} = 0$$

end

Step 1: Outward Recursion:- $i = 1, n$

$$\mathbf{W}_i = \mathbf{W}_{i-1}\mathbf{A}_i \qquad (5.2.4a)$$

$$\dot{\mathbf{W}}_i = \dot{\mathbf{W}}_{i-1}\mathbf{A}_i + \mathbf{W}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\dot{q}_i \tag{5.2.4b}$$

$$\ddot{\mathbf{W}}_i = \ddot{\mathbf{W}}_{i-1}\mathbf{A}_i + 2\,\dot{\mathbf{W}}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\dot{q}_i + \mathbf{W}_{i-1}\frac{\partial^2 \mathbf{A}_i}{\partial q_i^{\,2}}\dot{q}_i^{\,2} + \mathbf{W}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\ddot{q}_i \tag{5.2.4c}$$

end

Step 2 : Inward Recursion :- $i = n, 1$

$$\mathbf{D}_i = \mathbf{J}^i_{O_i}\,\ddot{\mathbf{W}}_i^T + \mathbf{A}_{i+1}\mathbf{D}_{i+1} \tag{5.2.5a}$$

$$\mathbf{c}_i^i = m_i\,\mathbf{r}^i_{i,i} + \mathbf{A}_{i+1}\mathbf{c}_{i+1}^{i+1} \tag{5.2.5b}$$

$$\tau_i = tr\left\{\frac{\partial \mathbf{W}_i}{\partial q_i}\mathbf{D}_i\right\} - \mathbf{g}^T\frac{\partial \mathbf{W}_i}{\partial q_i}\mathbf{c}_i^i \tag{5.2.5c}$$

end

For its implementation, Algorithm 5.1 requires $830n - 592$ scalar multiplications and $675n - 464$ scalar additions. Hence, Algorithm 5.1 has $o(n)$ computational complexity, but is still computationally inefficient for real-time applications, since for $n = 6$ it requires 4388 multiplications and 3586 additions, respectively, for its implementation. However, it was noticed by Hollerbach that the computational inefficiency of this algorithm resulted from the fact that homogeneous transformations are used to describe general rigid body displacements. Therefore, using rotation tensors and three dimensional displacement vectors to describe rigid body displacements, he wrote equation (5.2.3) as follows :

$$\tau_i = \sum_{j=i}^{n}\left\{tr\left[m_j\frac{\partial \mathbf{s}_{0,j}}{\partial q_i}\ddot{\mathbf{s}}^T_{0,j} + \frac{\partial \mathbf{s}_{0,j}}{\partial q_i}(\mathbf{n}_j^j)^T\ddot{\mathbf{W}}_j^T + \frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{n}_j^j\ddot{\mathbf{s}}^T_{0,j}\right.\right.$$

$$\left.\left. + \frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}^j_{O_j}\ddot{\mathbf{W}}_j^T\right] - m_j\,\mathbf{g}^T\frac{\partial \mathbf{W}_i}{\partial q_i}\mathbf{r}^i_{i,j}\right\}\dagger\ , \qquad i = 1, \cdots, n \tag{5.2.6}$$

Based on this equation, he proposed another algorithm which has basically the same

† This form is different from equation (15) in Hollerbach's formulation [14] and leads to a slightly modified analysis which is given in Appendix A. A consequence of this is that equation (5.2.8c) in Algorithm 5.2 is different from the corresponding equation (equation (13)) in Ref. [14].

recursive structure as Algorithm 5.1 and can be stated as follows :

ALGORITHM 5.2

Step 0: Initialization

$$\mathbf{W}_0 = \mathbf{I} \ , \quad \dot{\mathbf{W}}_i = 0 \ , \quad \ddot{\mathbf{W}}_i = 0 \ , \quad \ddot{\mathbf{s}}_{0,i} = 0$$
$$\mathbf{n}_i^i = m_i \mathbf{r}_{i,i}^i \ , \quad \mathbf{A}_{n+1} = 0 \ , \quad \mathbf{e}_{n+1} = 0$$

end

Step 1: Outward Recursion:- $i = 1, n$

$$\mathbf{W}_i = \mathbf{W}_{i-1}\mathbf{A}_i \tag{5.2.7a}$$

$$\dot{\mathbf{W}}_i = \dot{\mathbf{W}}_{i-1}\mathbf{A}_i + \mathbf{W}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\dot{q}_i \tag{5.2.7b}$$

$$\ddot{\mathbf{W}}_i = \ddot{\mathbf{W}}_{i-1}\mathbf{A}_i + 2\,\dot{\mathbf{W}}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\dot{q}_i + \mathbf{W}_{i-1}\frac{\partial^2 \mathbf{A}_i}{\partial q_i^{\,2}}\dot{q}_i^{\,2} + \mathbf{W}_{i-1}\frac{\partial \mathbf{A}_i}{\partial q_i}\ddot{q}_i \tag{5.2.7c}$$

$$\ddot{\mathbf{s}}_{0,i} = \ddot{\mathbf{s}}_{0,i-1} + \ddot{\mathbf{W}}_{i-1}\mathbf{s}_{i-1,i}^{i-1} \tag{5.2.7d}$$

end

Step 2 : Inward Recursion :- $i = n \ , \ 1$

$$\mathbf{e}_i = \mathbf{e}_{i+1} + m_i \ddot{\mathbf{s}}_{0,i}^T + \mathbf{n}_i^{i\,T}\ddot{\mathbf{W}}_i^T \tag{5.2.8a}$$

$$\mathbf{D}_i = \mathbf{A}_{i+1}\mathbf{D}_{i+1} + \mathbf{s}_{i,i+1}^i \mathbf{e}_{i+1} + \mathbf{n}_i^i \ddot{\mathbf{s}}_{0,i}^T + \mathbf{J}_{O_i}^i \ddot{\mathbf{W}}_i^T \tag{5.2.8b}$$

$$\mathbf{c}_i^i = m_i \mathbf{r}_{i,i}^i + \overline{m}_{i+1}\mathbf{s}_{i,i+1}^i + \mathbf{A}_{i+1}\mathbf{c}_{i+1}^{i+1} \tag{5.2.8c}$$

$$\tau_i = tr\left(\frac{\partial \mathbf{W}_i}{\partial q_i}\mathbf{D}_i\right) - \mathbf{g}^T\frac{\partial \mathbf{W}_i}{\partial q_i}\mathbf{c}_i^i \tag{5.2.8d}$$

end

For its implementation† Algorithm 5.2 requires $412n - 277$ scalar multiplications and $675n - 201$ scalar additions, which for $n = 6$ gives 2195 scalar multiplications and 1719 scalar additions, respectively, and this is a significant improvement over Algorithm 5.1.

Besides the representation of physical quantities, it was soon realized that the structure of the kinematic and dynamic recurrence relations have a direct effect on the

† Note that the difference in equation (5.2.8c) does not change the computational requirements of the original equation (equation (13) in Ref. [14]) when the $(i+1)$-th joint is revolute. When the $(i+1)$-th joint is prismatic the implementation of equation (5.2.8c) requires a few extra computations.

computational complexity of recursive as well as closed-form manipulator dynamics algorithms. In particular, it was realized that the structure of the kinematic and dynamic recurrence relations depends on the particular modeling scheme which is used for deriving the equations of motion and on the final formulation of these equations. As result of this, modeling schemes based on the ideas of augmented and generalized links [22] or on the idea of articulated bodies [48] have been proposed. An advantage of such modeling schemes is that they lead to efficient recurrence relations for computing the generalized inertia tensor of a robot manipulator. These modeling schemes are best utilized when the Lagrangian formulation is described by the following closed-form equation

$$\tau_i(t) = \sum_{j=1}^{n} d_{i,j}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} \dot{q}_j\, c_{j,k}(i)\dot{q}_k + g_i \qquad i = 1, 2, \cdots, n \tag{5.2.9}$$

where $d_{i,j}$ are the inertia coefficients,

$$c_{j,k}(i) = \frac{1}{2}\left[\frac{\partial d_{i,j}}{\partial q_k} + \frac{\partial d_{i,k}}{\partial q_j} - \frac{\partial d_{j,k}}{\partial q_i}\right] \tag{5.2.10}$$

are the centrifugal and Coriolis coefficients, defined using Christoffel symbols, and g_i are the gravitational coefficients. Thus, with this formulation and analytical procedures for better organization of the computations, many researchers, e.g. [23, 24, 48] have proposed efficient algorithms which are of complexity $O(n^3)$. The latter results from the fact that the algorithms are in closed-form and thus there are $\frac{1}{6}n(2n^2 + 3n + 7)$ independent coefficients to be computed [29]. Another efficient algorithm worth mentioning which uses the Lagrangian formulation and analytic procedures, but was originally devised in conjunction with the Newton-Euler formulation, is the algorithm proposed recently (1988) by Li [26]. Also, it is worth mentioning here, that a new method for deriving the Euler-Lagrange equations of motion has been devised by Angeles and Lee [27]. In this method the dynamic equations of motion are derived based on a *natural orthogonal complement* of the velocity constraints equations of motion, instead of the Lagrangian function. Application of this method in deriving efficient algorithms for

solving the IDP can be found in [28].

Finally, to reduce further the computational cost for solving the IDP, the particular kinematic and dynamic structures of the manipulator were taken into consideration. This effort resulted in a class of dynamic algorithms which are referred to in the literature as *customized* algorithms. Customized robot dynamics algorithms are based on mathematical models of individual manipulators rather than on general-purpose formulations. Hence, customized algorithms can be systematically organized [29-31] and therefore such algorithms exhibit a significant increase in computational efficiency over general-purpose algorithms. Within the Lagrangian formulation, customized inverse dynamics algorithms have been proposed by Renaud [23], Burdick [25], Neuman and Murray [29] and Ramos and Khosla [30]. Also some aspects of better manipulator design for reduced dynamic complexity have been considered [32,33]. In this approach, the kinematic structure and mass distribution of a manipulator arm are designed so that the inertia of the manipulator becomes diagonal and/or invariant for an arbitrary arm configuration. However, this approach leads to algorithms which are applicable to particular classes of manipulators with two and three degrees of freedom only.

5.2.2 Formulations Based on Newton-Euler Equations

In the Newton-Euler approach, the equations of motion (Newton's and Euler's) are applied to each link and the resulting equations are combined with constraint equations from the joints in such a way as to give the joint generalized forces in terms of the joint acceleration. This method was originally developed for multi-body satellite and spacecraft dynamics [34]. The first applications of the Newton-Euler dynamic equations to robotic systems may be found, among others, in the work of Stepanenko and Vucobratovic [35], Vucobratovic [36], Ho [37] and Hughes [38]. However, algorithms based on these methods, as in the case of the Uicker/Kahn method of the Lagrangian formulation, are computationally very inefficient. These early formulations led to closed-form

algorithms which have computational complexity $o(n^3)$ or in some cases even $o(n^4)$ [14,24].

A more efficient method was proposed by Orin et al [39] by introducing link coordinate systems. Using relationships of moving coordinate systems, they where able to achieve more efficient kinematic recurrence relations for computing velocities and accelerations and dynamic recurrence relations for computing forces and torques. Based on these relations, they derived an algorithm which has computational complexity $o(n^2)$. However, in this method, the basic equations of motion for each link are expressed in the inertial coordinate system and this involves unnecessary coordinate transformations. Thus, the number of scalar operations is not sufficiently reduced to make the method implementable in real time. Luh, Walter and Paul [40], modified this method by expressing the equations of motion in link coordinate systems instead the inertial coordinate system. They proposed a recursive algorithm which for its implementation requires $150n - 48$ scalar multiplications and $131n - 48$ scalar additions, so that for $n = 6$ this algorithm requires 852 scalar multiplications and 738 scalar additions [14].

This algorithm has $o(n)$ computational complexity and is far more efficient than Algorithm 5.2 which also has $o(n)$ computational complexity, but is derived based on the Lagrangian formulation. The difference in the computational complexity of these two algorithms, sparked a debate about which of the two formulations, i.e., the Lagrangian or the Newton-Euler, leads to more efficient computational algorithms for solving the IDP. For some time it was believed, due to lack of deeper understanding of the mathematical representations used to describe the equations of motion, that the algorithms derived from the Newton-Euler formulation are computationally more efficient than those derived using the Lagrangian formulation. Silver [41] resolved the issue by showing that both formulations are equivalent, in the sense that the Lagrangian formulation will yield a similar algorithm to that obtained using the Newton-Euler formula-

tion, if an equivalent representation of the angular velocity is employed.

Currently, the method proposed by Luh, Walker and Paul is probably the best method for deriving recursive algorithms to compute manipulator inverse dynamics. This method can be outlined as follows : Based on moving coordinate systems, kinematic recurrence relations are used to compute velocities and accelerations from the base of the manipulator to the end-effector, link-by-link. Then dynamic recurrence relations are used to compute forces and torques from the end-effector back to the base of the manipulator. In this process, because of the nature of these recurrence relations, the generalized forces are computed by simple projections of vector quantities onto the joint axes. From this outline, it is obvious that the computational efficiency of these algorithms results from the fact that they compute efficiently, only that information needed to characterize rigid-body movements. To illustrate this method further, we present below a modified version of the algorithm proposed by Luh, Walker and Paul. In their original algorithm, the link coordinate systems were assigned following a different convention from that presented in Chapter II. Therefore, for notational convenience, we have chosen the following algorithm which has been taken from Craig [42] and for which the link coordinates are assigned as described in Chapter II.

ALGORITHM 5.3 †

Step 0 : Initialization

$$\mathbf{z}_i^i = [\,0\;0\;1\,]^T$$

$$\sigma_i = \begin{cases} 1 & \text{revolute } i\text{-th joint} \\ 0 & \text{prismatic } i\text{-th joint} \end{cases}, \qquad q_i = \begin{cases} \theta_i & \text{revolute } i\text{-th joint} \\ d_i & \text{prismatic } i\text{-th joint} \end{cases}$$

$$\omega_0^0 = 0\,, \quad \dot{\omega}_0^0 = 0\,, \quad \ddot{\mathbf{s}}_{0,0}^0 = -\,\mathbf{g}\,, \quad \mathbf{A}_{n+1} = 0$$

end

† Craig's algorithm is stated for revolute joints only. Here the necessary modifications are included for the algorithm to be also applicable in the presence of prismatic joints.

Step 1 : Forward recursion :- For $i = 0, n\text{-}1$ do

$$\boldsymbol{\omega}_{i+1}^{i+1} = \mathbf{A}_{i+1}^{T}\,\boldsymbol{\omega}_{i}^{i} + \sigma_{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} \tag{5.2.10a}$$

$$\dot{\boldsymbol{\omega}}_{i+1}^{i+1} = \mathbf{A}_{i+1}^{T}\,\dot{\boldsymbol{\omega}}_{i}^{i} + \sigma_{i+1}\left[\mathbf{A}_{i+1}^{T}\,\boldsymbol{\omega}_{i}^{i}\times\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right] \tag{5.2.10b}$$

$$\ddot{\mathbf{s}}_{0,i+1}^{i+1} = \mathbf{A}_{i+1}^{T}\left[\ddot{\mathbf{s}}_{0,i}^{i} + \dot{\boldsymbol{\omega}}_{i}^{i}\times\mathbf{s}_{i,i+1}^{i} + \boldsymbol{\omega}_{i}^{i}\times\left(\boldsymbol{\omega}_{i}^{i}\times\mathbf{s}_{i,i+1}^{i}\right)\right] + (1-\sigma_{i+1})\left(2\boldsymbol{\omega}_{i}^{i+1}\times\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right) \tag{5.2.10c}$$

$$\ddot{\mathbf{r}}_{0,i+1}^{i+1} = \dot{\boldsymbol{\omega}}_{i+1}^{i+1}\times\mathbf{r}_{i+1,i+1}^{i+1} + \boldsymbol{\omega}_{i+1}^{i+1}\times(\boldsymbol{\omega}_{i+1}^{i+1}\times\mathbf{r}_{i+1,i+1}^{i+1}) + \ddot{\mathbf{s}}_{0,i+1}^{i+1} \tag{5.2.10d}$$

$$\mathbf{F}_{C_{i+1}}^{i+1} = m_{i+1}\ddot{\mathbf{r}}_{0,i+1}^{i+1} \tag{5.2.10e}$$

$$\mathbf{M}_{C_{i+1}}^{i+1} = \mathbf{I}_{C_{i+1}}^{i+1}\dot{\boldsymbol{\omega}}_{i+1}^{i+1} + \boldsymbol{\omega}_{i+1}^{i+1}\times\mathbf{I}_{C_{i+1}}^{i+1}\boldsymbol{\omega}_{i+1}^{i+1} \tag{5.2.10f}$$

end

Step 2 : Backward recursion :- For $i = n, 1$ do

$$\mathbf{f}_{i}^{i} = \mathbf{A}_{i+1}\mathbf{f}_{i+1}^{i+1} + \mathbf{F}_{C_i}^{i} \tag{5.2.11a}$$

$$\boldsymbol{\eta}_{i}^{i} = \mathbf{M}_{C_i}^{i} + \mathbf{s}_{i,i+1}^{i}\times\mathbf{A}_{i+1}\mathbf{f}_{i+1}^{i+1} + \mathbf{r}_{i,i}^{i}\times\mathbf{F}_{C_i}^{i} + \mathbf{A}_{i+1}\boldsymbol{\eta}_{i+1}^{i+1} \tag{5.2.11b}$$

$$\tau_i = \sigma_i(\boldsymbol{\eta}_{i}^{i}\cdot\mathbf{z}_{i}^{i}) + (1-\sigma_i)(\mathbf{f}_{i}^{i}\cdot\mathbf{z}_{i}^{i}) \tag{5.2.11c}$$

end

To improve further the computational efficiency of the algorithms derived from this method, researchers have proposed a number of different ways of assigning the link coordinate systems and various analytical procedures for organizing the computations [42-44]. Also, as in the case of the Lagrangian formulation, different modeling schemes such as those using augmented or articulated bodies have been used for deriving more efficient recurrence relations [45-48]. Moreover, customized algorithms have also been proposed for the Newton-Euler formulation [49-52]. Finally, to facilitate real-time implementation of advanced robot control strategies, parallel processing techniques have been used [53-56] to implement many of the existing algorithms which compute inverse dynamics.

5.2.3 Formulations Based on Kane's Equations

In Kane's approach, one first describes the generalized *active* force and the generalized *inertia* force of a system in terms of generalized coordinates, generalized speeds,

partial linear velocities, and partial angular velocities. Then, the dynamic equations of motion (Kane's equations) are obtained by setting the sum of these two forces equal to zero according to the d'Alembert principle. Kane's equations were first introduced by Kane for general nonholonomic mechanical systems [57] and were used, as the Newton-Euler and Euler-Lagrange equations, in rigid multi-body satellite and spacecraft dynamic analysis [58,59]. Huston and Kelly [60], were among the first to apply Kane's equations in robotics. However, they presented neither an explicit algorithm for inverse dynamics nor a complexity analysis of their method. Kane's equations were also used by Faessler [61], who presented a method which leads to a closed-form algorithm for solving inverse dynamics. Using analytical procedures, Faessler expressed the entries of the coefficient matrices in symbolic form, but did not provide a complexity analysis of his method. Kane and Levinson [62] also presented a customized algorithm for solving inverse dynamics for the Stanford arm. But, although their procedure is conceptually simple, it requires considerable experience with handling complex dynamical systems and involves an extensive manual analysis for setting up a large number of intermediate variables, which are not defined from recurrence relations.

A recursive algorithm for computing inverse dynamics, based on Kane's equations has been proposed by Ma [63]. Using analytical procedures, Ma was able to derive from Kane's equations the following equations for the i-th component ($i = 1, 2, \cdots, n$) of the generalized force τ:

i) When the i-th joint is a revolute joint,

$$\tau_i = \mathbf{z}_i^i \cdot \left(\sum_{j=i}^{n} \left[\mathbf{M}_{C_j}^i + \mathbf{r}_{i,j}^i \times m_j \ddot{\mathbf{r}}_{0,j}^i \right] \right) \tag{5.2.13}$$

ii) When the i-th joint is a prismatic joint,

$$\tau_i = \mathbf{z}_i^i \cdot \left(\sum_{j=i}^{n} m_j \ddot{\mathbf{r}}_{0,j}^i \right) \tag{5.2.14}$$

Equations (5.2.13) and (5.2.14) are identical to the equations derived by Silver [41] using the Euler-Lagrange equations and therefore, as Silver has shown, they can also be derived from the Newton-Euler formulation. Based on these equations and kinematic and dynamic recurrence relations, similar to those introduced by Luh, Walker and Paul [40], Ma proposed a recursive algorithm which is similar in structure to Algorithm 5.3. For a "semi-customized" implementation this algorithm requires $109n - 109$ scalar multiplications and $95n - 108$ scalar additions (where $n \geq 2$).

5.2.4 Observations Concerning Computational Issues of the IDP

Based on this brief survey, we can make the following observations concerning various computational issues in evaluating manipulator inverse dynamics.

It is clear now, that for solving manipulator inverse dynamics, recursive algorithms are computationally more efficient than closed-form ones, since in recursive algorithms unnecessary computations (usually duplications) are avoided. Also, it is clear [64] that the computational efficiency of an algorithm for solving the IDP is independent of the particular equations of motion (Newton-Euler, Euler-Lagrange or Kane's) used to derive it. The computational efficiency of these recursive algorithms, as Silver [41] has pointed out depends mainly on "the structure of the computation and choice of representation" and the survey on inverse dynamics verifies Silver remark. To elaborate more on this important remark, we restate it as follows : The computational efficiency of a recursive algorithm for solving inverse dynamics, depends mainly on the following factors :

(a) The particular representation of various physical quantities appearing in the equations of motion.

(b) The underlying modeling scheme used for the manipulator.

(c) The organization of the computations and the degree of customization involved in its numerical or symbolic implementation.

As is evident from the brief survey presented above, the organization of the computations and the degree of customization is actually the point where most of the existing recursive algorithms differ. The large number of these algorithms reveals that many alternatives have been considered in reducing the computational cost. However, particular analytical organization procedures and customization are usually used for the implementation of an algorithm and not for deriving it. Our basic objective in this thesis is to improve the computational efficiency of algorithms for solving the IDP through a better understanding of the mathematical representations used to describe the equations of motion and not through better implementation of existing formulations. More information on the latter aspect can be found for instance in [31] and in the extensive list of references cited there.

For the class of robot manipulators we are dealing with, namely, rigid-link, open-loop, chain-like manipulators, the modeling scheme is simple and common to most of the existing algorithms. Usually, each link is considered to be a rigid body and the manipulator is modeled as an ideally connected (i.e, without friction or any backlash) open-chain of rigid bodies. This chain is assumed to be a rigid structure when static force analysis is required, as in the case of the Newton-Euler formulation. This modeling scheme, as well as another one which utilizes the ideas of augmented and generalized links will be used in the next section to demonstrate the effects of the underlying modeling scheme on the structure of a recursive algorithm. In particular we shall show that the modeling scheme which is based on augmented and generalized links leads to algorithms which have computational advantages, because they allow for some quantities to be computed *off-line*.

Finally, the choice of representation is the most important factor which effects the computational efficiency of the algorithms we are dealing with. From the survey, it is clear that the debate about the Euler-Lagrange and Newton-Euler formulations, mentioned above, was actually an indirect debate about the proper representation (as far as

computational efficiency is concerned) of the angular velocity. Theoretically, it is known that the two formulations are equivalent. Therefore, the real question, although never stated explicitly as such, was which representation for the angular velocity describes more efficiently the angular motion of a rigid body. As noted by Silver [41], the angular motion of a rigid body can be described equally well by either the angular velocity vector, which is used in the Newton-Euler formulation, or the derivative of a rotation tensor which is used in the Lagrangian formulation. At the time Silver's work was published, the algorithm by Luh, Walker and Paul in particular, and the theory of classical dynamics in general, clearly indicated that the angular motion of a rigid body was described more efficiently by the angular velocity vector. Thus, the vector representation of angular velocity became standard in the dynamic analysis of robotic systems. Based on the vector representation of angular velocity, all the other quantities which are defined in terms of angular velocity are also represented by vectors. Thus, vector representations and vector analysis have been used almost exclusively for deriving computationally efficient algorithms for solving the IDP.

However, as we have shown in Chapter IV, the angular motion of a rigid body is described more efficiently by using a Cartesian tensor representation for the angular velocity. Obviously, this provides another possibility for describing efficiently the dynamic equations of a rigid bodies system, in general, and solving the manipulator IDP in particular. Therefore, the question which arises at this point is the following : Does the tensor representation of the angular velocity lead us to recursive algorithms which compute inverse dynamics more efficiently than those algorithms which are derived based on the traditional vector representation of angular velocity ?

The answer to this question is in the affirmative as will be shown in the next section.

5.3 A CARTESIAN TENSOR APPROACH FOR SOLVING THE IDP

As we mentioned in the previous section, the method proposed by Luh, Walker and Paul is the most suitable method for deriving computationally efficient recursive algorithms for solving the IDP. In this section, we shall use this method and Cartesian tensor analysis to derive recursive algorithms for computing inverse dynamics, which are computationally far more efficient than similar algorithms derived using vector analysis. In particular, employing the methodology and basic theorems introduced in Chapter IV, we shall reformulate Algorithm 5.3, which was presented earlier (in Section 5.2). We shall do this by rewriting the basic vector equations of this algorithm in an equivalent, but computationally more efficient, tensor formulation. Moreover, to increase the computational efficiency of this algorithm further, we shall examine the underlying modeling scheme for the class of robot manipulators we are dealing with. To this end, we shall derive a second algorithm by using a modeling scheme which employs the ideas of augmented and generalized links. Finally, numerical implementation and computational complexity of these algorithms are considered and compared with similar algorithms that can be found in the literature.

5.3.1 New Algorithms for Computing the IDP

In the Newton-Euler formulation, the dynamic equations for robot manipulators are obtained by evaluating recursively the velocities and accelerations for each link followed by applications of Newton's and Euler's equations to each link. In the first step, the recursions are performed from link 1 to link n. Then, using static force and torque analysis, the joint actuator forces/torques are computed in the second step where the recursions are applied from joint n to joint 1. These recursions can be stated in an algorithmic form as was done for Algorithm 5.3.

In Algorithm 5.3, the absolute linear acceleration of the center of mass of each link is computed following the classical vector approach, where the absolute angular velocity

and acceleration are represented by vectors. Also, in this algorithm, the generalized Euler equation is stated in its classical vector formulation in terms of the vector angular velocity and vector angular acceler tion. However, as we have seen in Chapter IV, the absolute linear acceleration of a point on a moving rigid body as well as the Euler equation can be described by the angular acceleration tensor instead of using the vectors of angular velocity and acceleration. Therefore, in an effort to improve the computational efficiency of Algorithm 5.3, we shall use Cartesian tensor analysis to reformulate most of the recurrence relations in this algorithm. The basic tensor equations, proven in Chapters III and IV, will make the process here straightforward and simple. We need only to notice that the equations of motion in Algorithm 5.3 are written with reference to link coordinate systems as opposed to Chapter IV where the equations of motion are written with reference to an inertial coordinate system. However, as we mentioned in Chapter III, Cartesian tensor equations are invariant under orthogonal coordinate transformations. Therefore, all the equations which describe the rigid body motion and which in Chapter IV are written relative to an inertial coordinate system, will be written here relative to link coordinate systems by using appropriate orthogonal coordinate transformations.

To derive a tensor formulation for Algorithm 5.3, we obviously have to abandon Gibb's classical vector cross product operation. As we have shown in Chapter IV, we have to use the tensors $\tilde{\omega}$ and $\Omega = \dot{\tilde{\omega}} + \tilde{\omega}\tilde{\omega}$, written here with reference to appropriate link coordinate systems. Using these two tensors, the equations of Algorithm 5.3 can be modified as follows :

Equation (5.2.10b) can be expressed as

$$\dot{\omega}_{i+1}^{i+1} = \mathbf{A}_{i+1}^{T}\dot{\omega}_{i}^{i} + \sigma_{i+1}\left[\tilde{\omega}_{i}^{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right]. \tag{5.3.1}$$

Also, equation (5 2.10c) can be written as

$$\ddot{\mathbf{s}}_{0,i+1}^{i+1} = \mathbf{A}_{i+1}^{T}\left[\ddot{\mathbf{s}}_{0,i}^{i} + \Omega_{i}^{i}\mathbf{s}_{i,i+1}^{i}\right] + (1-\sigma_{i+1})\left(2\tilde{\omega}_{i}^{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right) \tag{5.3.2}$$

and equation (5.2.10d) can be written as

$$\ddot{\mathbf{r}}_{0,i+1}^{i+1} = \boldsymbol{\Omega}_{i+1}^{i+1}\mathbf{r}_{i+1,i+1}^{i+1} + \ddot{\mathbf{s}}_{0,i+1}^{i+1}. \tag{5.3.3}$$

Newton's equation, i.e., equation (5.2.10e) is very simple in its vector form and therefore we do not modify it. However, Euler's equation, i.e., equation (5.2.10f), assumes a simpler structure when it is written in tensor form and, in particular, when it is expressed in terms of the pseudo-inertia tensor $\mathbf{J}_{C_{i+1}}^{i+1}$. Therefore, using equation (4.3.12) to transfer the inertia tensor $\mathbf{I}_{C_{i+1}}^{i+1}$ to the pseudo-inertia tensor $\mathbf{J}_{C_{i+1}}^{i+1}$, we first write equation (5.2.10f) in the following tensor form

$$\tilde{\mathbf{M}}_{C_{i+1}}^{i+1} = \boldsymbol{\Omega}_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1} - \left(\boldsymbol{\Omega}_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1}\right)^{T} \tag{5.3.4}$$

and then we recover the vector $\mathbf{M}_{C_{i+1}}^{i+1}$ from the skew-symmetric tensor $\tilde{\mathbf{M}}_{C_{i+1}}^{i+1}$ by using the dual operator. The dual operator has been introduced in Chapter III by equation (3.3.33). Thus, in a tensor formulation the vector $\mathbf{M}_{C_{i+1}}^{i+1}$ is computed by the following equations,

$$\mathbf{J}_{C_{i+1}}^{i+1} = \frac{1}{2}tr\,[\mathbf{I}_{C_{i+1}}^{i+1}]\mathbf{1} - \mathbf{I}_{C_{i+1}}^{i+1} \tag{5.3.5a}$$

$$\tilde{\mathbf{M}}_{C_{i+1}}^{i+1} = \boldsymbol{\Omega}_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1} - \left(\boldsymbol{\Omega}_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1}\right)^{T} \tag{5.3.5b}$$

$$\mathbf{M}_{C_{i+1}}^{i+1} = dual\left(\tilde{\mathbf{M}}_{C_{i+1}}^{i+1}\right). \tag{5.3.5c}$$

In the second step of Algorithm 5.3, we need only to reformulate equation (5.2.11b) which can be written in the form

$$\boldsymbol{\eta}_i^i = \mathbf{M}_{C_i}^i + \tilde{\mathbf{s}}_{i,i+1}^i\mathbf{f}_{i+1}^i + \tilde{\mathbf{r}}_{i,i}^i\mathbf{F}_{C_i}^i + \mathbf{A}_{i+1}\boldsymbol{\eta}_{i+1}^{i+1}. \tag{5.3.6}$$

Now, using equations (5.3.1)-(5.3.6), we can state Algorithm 5.3 in a new formulation as follows :

ALGORITHM 5.4

Step 0 : Initialization

$$\mathbf{z}_i^i = [\,0\;0\;1\,]^T$$

$$\sigma_i = \begin{cases} 1 & \text{revolute } i\text{-th joint} \\ 0 & \text{prismatic } i\text{-th joint} \end{cases}, \qquad q_i = \begin{cases} \theta_i & \text{revolute } i\text{-th joint} \\ d_i & \text{prismatic } i\text{-th joint} \end{cases}$$

$$\omega_0^0 = 0\,,\quad \dot{\omega}_0^0 = 0\,,\quad \Omega_0^0 = 0\,,\quad \ddot{\mathbf{s}}_{0,0}^0 = -\mathbf{g}\,,\quad \mathbf{A}_{n+1} = 0$$

$$\mathbf{J}_{C_{i+1}}^{i+1} = \frac{1}{2} tr\,[\mathbf{I}_{C_{i+1}}^{i+1}]\mathbf{1} - \mathbf{I}_{C_{i+1}}^{i+1}$$

end

Step 1 : Forward recursion :- For $i = 0, n\text{-}1$ **do**

$$\omega_{i+1}^{i+1} = \mathbf{A}_{i+1}^T\,\omega_i^i + \sigma_{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} \tag{5.3.7a}$$

$$\dot{\omega}_{i+1}^{i+1} = \mathbf{A}_{i+1}^T\,\dot{\omega}_i^i + \sigma_{i+1}\left[\tilde{\omega}_i^{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right] \tag{5.3.7b}$$

$$\Omega_{i+1}^{i+1} = \tilde{\dot{\omega}}_{i+1}^{i+1} + \tilde{\omega}_{i+1}^{i+1}\tilde{\omega}_{i+1}^{i+1} \tag{5.3.7c}$$

$$\ddot{\mathbf{s}}_{0,i+1}^{i+1} = \mathbf{A}_{i+1}^T\left[\ddot{\mathbf{s}}_{0,i}^i + \Omega_i^i\mathbf{s}_{i,i+1}^i\right] + (1-\sigma_{i+1})\left(2\tilde{\omega}_i^{i+1}\mathbf{z}_{i+1}^{i+1}\dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1}\ddot{q}_{i+1}\right) \tag{5.3.7d}$$

$$\ddot{\mathbf{r}}_{0,i+1}^{i+1} = \Omega_{i+1}^{i+1}\mathbf{r}_{i+1,i+1}^{i+1} + \ddot{\mathbf{s}}_{0,i+1}^{i+1} \tag{5.3.7e}$$

$$\mathbf{F}_{C_{i+1}}^{i+1} = m_{i+1}\ddot{\mathbf{r}}_{0,i+1}^{i+1} \tag{5.3.7f}$$

$$\tilde{\mathbf{M}}_{C_{i+1}}^{i+1} = \Omega_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1} - \left(\Omega_{i+1}^{i+1}\mathbf{J}_{C_{i+1}}^{i+1}\right)^T \tag{5.3.7g}$$

$$\mathbf{M}_{C_{i+1}}^{i+1} = dual\left(\tilde{\mathbf{M}}_{C_{i+1}}^{i+1}\right) \tag{5.3.7h}$$

end

Step 2 : Backward recursion :- For $i = n, 1$ **do**

$$\mathbf{f}_i^i = \mathbf{A}_{i+1}\mathbf{f}_{i+1}^{i+1} + \mathbf{F}_{C_i}^i \tag{5.3.8a}$$

$$\eta_i^i = \mathbf{M}_{C_i}^i + \tilde{\mathbf{s}}_{i,i+1}^i\mathbf{f}_{i+1}^i + \tilde{\mathbf{r}}_{i,i}^i\mathbf{F}_{C_i}^i + \mathbf{A}_{i+1}\eta_{i+1}^{i+1} \tag{5.3.8b}$$

$$\tau_i = \sigma_i(\eta_i^i \cdot \mathbf{z}_i^i) + (1-\sigma_i)(\mathbf{f}_i^i \cdot \mathbf{z}_i^i) \tag{5.3.8c}$$

end

We shall be concerned with the numerical implementation of Algorithm 5.4 in the next section. However, as we can notice here, the structure of this algorithm reveals that for its implementation all the quantities (with the exemption of the pseudo-inertia tensors) have to be computed *on-line*. Obviously, from a computational point of view it is desirable to devise algorithms which allow us to compute *off-line* as many quantities as possible and at the same time, to keep the on-line computations as simple as possible. To see if this is feasible, we have to examine the underlying modeling scheme for the class of robot manipulators we are dealing with, since the structure of an algorithm obviously depends on it.

In general, the robot manipulator is modeled as an ideally connected, open-loop, serial-chain of rigid bodies. When frictional forces at the joints are to be considered we compute them based on the joint velocities and add them directly to the joint generalized forces. This clearly justifies the idealization in the connections of the rigid links. Now, utilizing this modeling scheme, kinematic and dynamic recurrence relations are defined, based on which the recursive Algorithm 5.3 (or Algorithm 5.4) has been derived. As is well known [42], the kinematic recurrence relations are defined by analyzing the velocity "propagation" from link to link starting from the base of the manipulator to the end-effector. The dynamic recurrence relations are defined based on a static force and moment analysis. In particular, to derive the dynamic recurrence relations in Algorithm 5.3, or in Algorithm 5.4, it has been assumed that the manipulator is locked at the joints so that it becomes a structure which is "rigid" in static equilibrium, and that static force and moment analysis has been performed for each link. However, in defining these dynamic recurrence relations, the analysis for the static forces and moments can be modified. For the static analysis, as long as we do not disturb the static equilibrium position of the manipulator, we are free to merge links and thus to generate hypothetical "generalized" links or even to assume the presence of "fictitious" links. In the following, using this "unconventional" static analysis we shall modify the dynamic recurrence relations of Algorithm 5.4. Then, based on these modified dynamic recurrence relations, we shall state a new algorithm which, when applied to most industrial robot manipulators in use today, allows us to compute some quantities off-line.

To proceed, we need to introduce first the concepts of augmented and generalized links, which are shown in Figure 5.1.

Definition 5.1 : An augmented link i is a *fictitious* link composed of link i and the mass of links $i+1, i+2, \cdots, n$, attached to the origin of the $(i+1)$-th coordinate system.

This definition which can be applied regardless of the type of joint (i.e., revolute or prismatic) is slightly different from the one presented in [22] in that the mass of the augmented link is not the total mass of the system (here the robot manipulator). Note that an augmented link is "rigid" (i.e., has fixed geometry) if and only if the (i+1)th joint is revolute because, when the $(i+1)$-th joint is prismatic the position vector of the origin of the $(i+1)$-th coordinate system relative to the origin of the i-th coordinate system, i.e., the vector $\mathbf{a}^i_{i,i+1}$, is not constant.

Definition 5.2 : A generalized link i is a composite link, consisting of links i through n treated as a single rigid body structure.

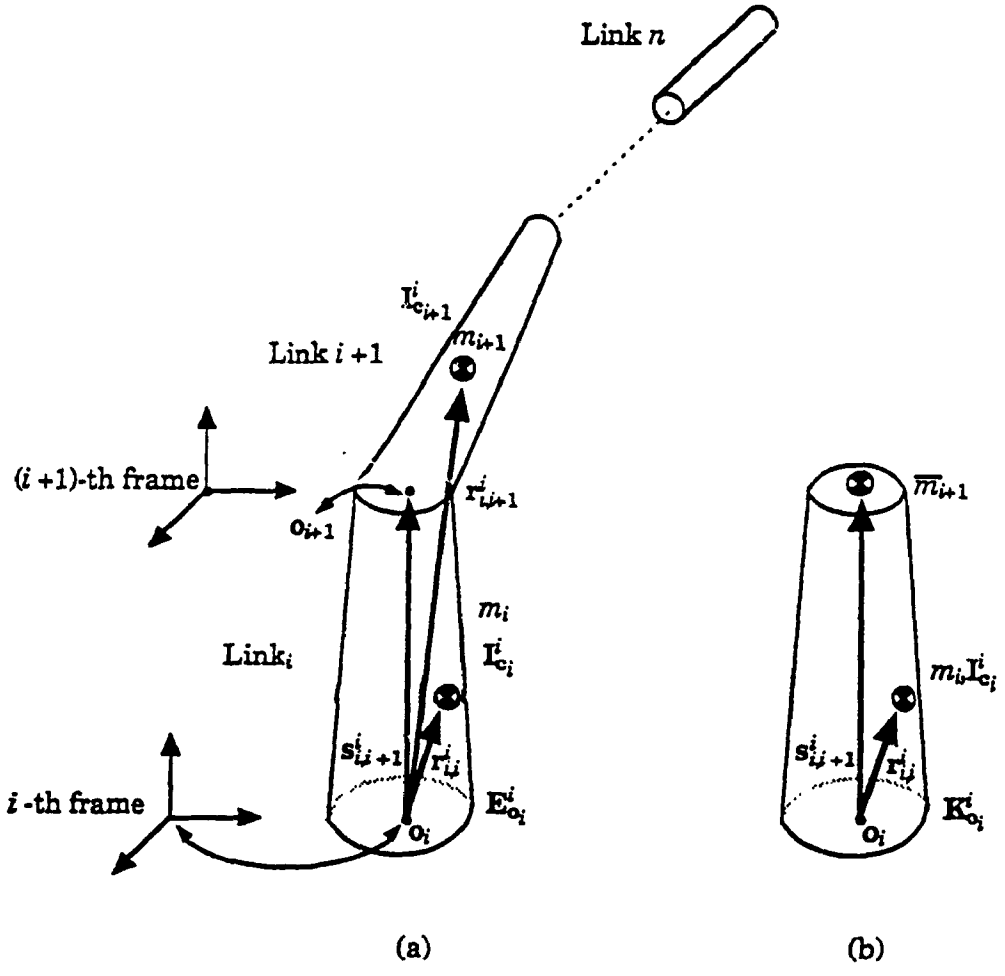

Figure 5.1: (a) The i-th Generalized Link, (b) The i-th Augmented Link

For the modification of the dynamic recurrence relations of Algorithm 5.4 we need to define the following moments :

(a) The 0-th moment or mass of the i-th augmented or the i-th generalized link :

$$\begin{aligned}\overline{m}_i &= \sum_{j=i}^{n} m_j \\ &= m_i + \overline{m}_{i+1}\end{aligned} \tag{5.3.9}$$

where m_i is the mass of the i-th link.

Also, the first and second moments with respect to the origin of the i-th coordinate system of the augmented link i, expressed in the i-th link frame, are defined as follows:

(b) First moment of augmented link i :

$$\mathbf{u}_{O_i}^i = m_i \mathbf{r}_{i,i}^i + \overline{m}_{i+1}\mathbf{s}_{i,i+1}^i. \tag{5.3.10}$$

(c) Second moment or inertia tensor of augmented link i :

$$\mathbf{K}_{O_i}^i = \mathbf{I}_{C_i}^i - m_i \tilde{\mathbf{r}}_{i,i}^i \tilde{\mathbf{r}}_{i,i}^i - \overline{m}_{i+1}\tilde{\mathbf{s}}_{i,i+1}^i \tilde{\mathbf{s}}_{i,i+1}^i \tag{5.3.11}$$

where $\mathbf{I}_{C_i}^i$ is the inertia tensor of link i with respect to its center of mass expressed in the i-th coordinate frame.

Note that when the $(i+1)$-th joint is revolute, the first and second moments are independent of the configuration of the manipulator and can be computed off-line. We also need the first moment about the origin of the i-th coordinate frame of the generalized link i. This is obtained as

(d) First moment of generalized link i :

$$\mathbf{U}_{O_i}^i = \sum_{j=i}^{n} m_j \mathbf{r}_{i,j}^i. \tag{5.3.12}$$

In the equation above, $\mathbf{U}_{O_i}^i$, expressed in the i-th link frame, is configuration dependent, and therefore must be calculated on-line. However, we can compute $\mathbf{U}_{O_i}^i$ recursively as the following lemma shows:

Lemma 5.1 : The first moment of the i-th generalized link satisfies the following recursive equation

$$\mathbf{U}^i_{O_i} = \mathbf{u}^i_{O_i} + \mathbf{A}_{i+1}\mathbf{U}^{i+1}_{O_{i+1}}. \tag{5.3.13}$$

Proof: From equation (5.3.12) we have

$$\begin{aligned}\mathbf{U}^i_{O_i} &= \sum_{j=i}^{n} m_j \mathbf{r}^i_{i,j} \\ &= m_i \mathbf{r}^i_{i,i} + \sum_{j=i+1}^{n} m_j \mathbf{r}^i_{i,j}.\end{aligned}$$

Since for $j > i$, $\mathbf{r}^i_{i,j} = \mathbf{s}^i_{i,i+1} + \mathbf{A}_{i+1}\mathbf{r}^{i+1}_{i+1,j}$, we have

$$\begin{aligned}\mathbf{U}^i_{O_i} &= m_i \mathbf{r}^i_{i,i} + \sum_{j=i+1}^{n} m_j \mathbf{s}^i_{i,i+1} + \sum_{j=i+1}^{n} \mathbf{A}_{i+1} m_j \mathbf{r}^{i+1}_{i+1,j} \\ &= m_i \mathbf{r}^i_{i,i} + \bar{m}_{i+1}\mathbf{s}^i_{i,i+1} + \mathbf{A}_{i+1}\mathbf{U}^{i+1}_{O_{i+1}} \\ &= \mathbf{u}^i_{O_i} + \mathbf{A}_{i+1}\mathbf{U}^{i+1}_{O_{i+1}}. \qquad \square\end{aligned}$$

In the following, we shall analyze the rotational motion of an augmented link, say the i-th one. For the sake of simplicity we assume first that the i-th augmented link has rigid body characteristic, i.e., the $(i+1)$-th joint is assumed to be revolute. Later we shall extend the analysis to include augmented links for which the $(i+1)$-th joint is of prismatic type. We begin by reviewing the rotational motion of the i-th link about the i-th joint since both the i-th link and the i-th augmented link have similar dynamic analysis. In particular, both have the same angular velocity and angular acceleration. However, since the i-th augmented link has different mass from the i-th link, it has obviously different dynamic characteristics.

When the i-th link experiences a rotational motion with center of rotation at the origin of the i-th coordinate system and with angular velocity ω_i^i and angular acceleration $\dot{\omega}_i^i$, a resultant torque or moment of force vector $\mathbf{M}^i_{O_i}$ is developed with respect to the center of rotation which, as shown in Chapter IV, satisfies equation (4.3.36). This equation, expressed in link coordinate system orientation, is written here as

$$\mathbf{M}^i_{O_i} = \mathbf{M}^i_{C_i} + \mathbf{r}^i_{i,i} \times \mathbf{F}^i_{C_i} \tag{5.3.14}$$

where

$$\mathbf{M}^i_{C_i} = \mathbf{I}^i_{C_i}\dot{\omega}^i_i + \tilde{\omega}^i_i \mathbf{I}^i_{C_i}\omega^i_i \tag{5.3.15}$$

is the resultant torque with respect to the center of mass and

$$\mathbf{F}^i_{C_i} = m_i \ddot{\mathbf{r}}^i_{i,i} \tag{5.3.16}$$

is the total force caused at the center of mass of the i-th link due to its acceleration. Using equation (5.3.16), we can write equation (5.3.14) in the form

$$\mathbf{M}^i_{O_i} = \mathbf{M}^i_{C_i} + m_i \tilde{\mathbf{r}}^i_{i,i} \ddot{\mathbf{r}}^i_{i,i}. \tag{5.3.17}$$

Equation (5.3.17) is the basic equation which describes the rotational motion of the i-th link. Now, if instead of the i-th link, the i-th augmented link actually experiences this rotational motion, then equation (5.3.17) needs to be changed to

$$\boldsymbol{\mu}^i_i = \mathbf{M}^i_{C_i} + m_i \tilde{\mathbf{r}}^i_{i,i} \ddot{\mathbf{r}}^i_{i,i} + \overline{m}_{i+1} \tilde{\mathbf{s}}^i_{i,i+1} \ddot{\mathbf{s}}^i_{i,i+1} \tag{5.3.18}$$

where the term $\overline{m}_{i+1} \tilde{\mathbf{s}}^i_{i,i+1} \ddot{\mathbf{s}}^i_{i,i+1}$ has been added to account for the torque which will be caused due to the presence of a mass equal to $\overline{m}_{i+1}$ at a point which has position vector $\mathbf{s}^i_{i,i+1}$ relative to the center of rotation. Obviously, the resultant torque vector is now denoted by a different vector, the vector $\boldsymbol{\mu}^i_i$. Moreover, as we have shown in Chapter IV, equation (5.3.17), which is actually another formulation for the generalized Euler equation, can be written in terms of the link inertia tensor as follows

$$\mathbf{M}^i_{O_i} = \mathbf{I}^i_{O_i} \dot{\boldsymbol{\omega}}^i_i + \tilde{\boldsymbol{\omega}}^i_i \mathbf{I}^i_{O_i} \boldsymbol{\omega}^i_i \tag{5.3.19}$$

where $\mathbf{I}^i_{O_i} = \mathbf{I}^i_{C_i} - m_i \tilde{\mathbf{r}}^i_{i,i} \tilde{\mathbf{r}}^i_{i,i}$ is the inertia tensor of the i-th link considered with respect to the origin of the i-th link coordinate system. Therefore, by analogy, we can say that equation (5.3.18) describes the generalized Euler equation for the i-th augmented link, when this is written with respect to the origin of the i-th link coordinate system. Obviously, as in the case with the i-th link, the generalized Euler equation can be written in terms of the inertia tensor $\mathbf{K}^i_{O_i}$ of the i-th augmented link and the vectors of the angular velocity $\boldsymbol{\omega}^i_i$ and angular acceleration $\dot{\boldsymbol{\omega}}^i_i$, as follows

$$\boldsymbol{\mu}^i_i = \mathbf{K}^i_{O_i} \dot{\boldsymbol{\omega}}^i_i + \tilde{\boldsymbol{\omega}}^i_i \mathbf{K}^i_{O_i} \boldsymbol{\omega}^i_i \tag{5.3.20}$$

where the inertia tensor $\mathbf{K}^i_{O_i}$ of the i-th augmented link is defined by equation (5.3.11).

To see that equation (5.3.18) is indeed equivalent to equation (5.3.20), we proceed as follows. First we need to prove the following relations

$$\tilde{\mathbf{r}}^i_{i,i}\ddot{\mathbf{r}}^i_{i,i} = -\left[\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i}\dot{\omega}^i_i + \tilde{\omega}^i_i\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i}\omega^i_i\right] \tag{5.3.21}$$

and

$$\tilde{\mathbf{s}}^i_{i,i+1}\ddot{\mathbf{s}}^i_{i,i+1} = -\left[\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1}\dot{\omega}^i_i + \tilde{\omega}^i_i\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1}\omega^i_i\right]. \tag{5.3.22}$$

To prove (5.3.21), we first note that since $\mathbf{r}^i_{i,i}$ is a constant vector, its absolute acceleration satisfies the equation

$$\ddot{\mathbf{r}}^i_{i,i} = \Omega^i_i\mathbf{r}^i_{i,i} \tag{5.3.23}$$

or

$$\begin{aligned}\ddot{\mathbf{r}}^i_{i,i} &= \left(\tilde{\dot{\omega}}^i_i + \tilde{\omega}^i_i\tilde{\omega}^i_i\right)\mathbf{r}^i_{i,i}\\ &= \tilde{\dot{\omega}}^i_i\mathbf{r}^i_{i,i} + \tilde{\omega}^i_i\tilde{\omega}^i_i\mathbf{r}^i_{i,i}\\ &= -\tilde{\mathbf{r}}^i_{i,i}\dot{\omega}^i_i - \tilde{\omega}^i_i\tilde{\mathbf{r}}^i_{i,i}\omega^i_i\end{aligned} \tag{5.3.24}$$

where equation (3.4.4) has been used in the last step. Therefore, we can write

$$\tilde{\mathbf{r}}^i_{i,i}\ddot{\mathbf{r}}^i_{i,i} = -\left(\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i}\dot{\omega}^i_i + \tilde{\mathbf{r}}^i_{i,i}\tilde{\omega}^i_i\tilde{\mathbf{r}}^i_{i,i}\omega^i_i\right). \tag{5.3.25}$$

Moreover, using equation (3.4.22) we can write

$$\tilde{\mathbf{r}}^i_{i,i}\tilde{\omega}^i_i\tilde{\mathbf{r}}^i_{i,i} = \tilde{\omega}^i_i\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i} + \tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i}\tilde{\omega}^i_i - \frac{1}{2}tr\left[\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i}\right]\omega^i_i. \tag{5.3.26}$$

Then, by substituting equation (5.3.26) into equation (5.3.25) and since $\tilde{\omega}^i_i\omega^i_i = 0$, equation (5.3.25) becomes equation (5.3.21). Also, since the $(i+1)$-th joint has been assumed to be of revolute type, the vector $\mathbf{s}^i_{i,i+1}$ is constant and so, we have

$$\ddot{\mathbf{s}}^i_{i,i+1} = \Omega^i_i\mathbf{s}^i_{i,i+1}. \tag{5.3.27}$$

Therefore, Equation (5.3.22) can be proved following the same arguments as in the proof of equation (5.3.21). Now, substituting equations (5.3.15), (5.3.21) and (5.3.22) into equation (5.3.18) and using the definition of $\mathbf{K}^i_{o_i}$ from equation (5.3.11), we get equation (5.3.20).

In this analysis, the generalized Euler equation for both the i-th link and the i-th augmented link has been stated in its vector form. However, as we have seen in Chapter IV, the generalized Euler equation can also be stated in its tensor formulation. Moreover, the tensor formulation of the generalized Euler equation assumes a simpler formulation when it is stated in terms of the rigid body pseudo-inertia tensor. Therefore, for a simple tensor formulation of equation (5.3.20) we need to define the pseudo-inertia tensor of the i-th generalized link. This can be done easily, if one uses equation (4.3.12) which transforms the rigid body inertia tensor into the pseudo-inertia tensor. Thus, we shall use the symbol $\mathbb{K}^i_{O_i}$ to denote the pseudo-inertia tensor of the i-th augmented link and we define it as follows

$$\mathbb{K}^i_{O_i} = \frac{1}{2} tr\left[\mathbf{K}^i_{O_i}\right] 1 - \mathbf{K}^i_{O_i} \tag{5.3.28}$$

where $\mathbf{K}^i_{O_i}$ is the inertia tensor of the i-th augmented link. Using this pseudo-inertia tensor, we can state the generalized Euler equation of the i-th augmented link in a tensor form as follows,

$$\tilde{\mu}^i = \Omega^i_i \mathbb{K}^i_{O_i} - \left[\Omega^i_i \mathbb{K}^i_{O_i}\right]^T . \tag{5.3.29}$$

where Ω^i_i is the angular acceleration of the i-th augmented link. Obviously, from the skew-symmetric tensor $\tilde{\mu}^i_i$, we can recover the vector invariant μ^i_i by using the dual operator, i.e., we can write

$$\mu^i_i = dual\left(\tilde{\mu}^i\right).$$

From the foregoing, the dynamic analysis for the i-th augmented link, when the $(i+1)$-th joint is revolute, is very simple. However, this analysis has to be modified when the $(i+1)$-th joint is prismatic because, in this case the i-th augmented link is not a rigid body. As we have mentioned above, the vector $\mathbf{s}^i_{i,i+1}$ is not constant in this case and therefore, the vector $\ddot{\mathbf{s}}^i_{i,i+1}$ is not equal to $\Omega^i_i \mathbf{s}^i_{i,i+1}$. The correct expression for the vector $\ddot{\mathbf{s}}^i_{i,i+1}$ follows from equation (5.3.7d) and is the following.

$$\ddot{\mathbf{s}}_{i,i+1}^{i} = \Omega_i^i \mathbf{s}_{i,i+1}^{i} + 2\tilde{\omega}_i^i \mathbf{z}_{i+1}^{i} \dot{q}_{i+1} + \mathbf{z}_{i+1}^{i} \ddot{q}_{i+1} \tag{5.3.30}$$

or,

$$\ddot{\mathbf{s}}_{i,i+1}^{i} = \Omega_i^i \mathbf{s}_{i,i+1}^{i} + (1 - \sigma_{i+1}) \varsigma_{i,i+1}^{i} \tag{5.3.31}$$

where

$$\varsigma_{i,i+1}^{i} = 2\tilde{\omega}_i^i \mathbf{z}_{i+1}^{i} \dot{q}_{i+1} + \mathbf{z}_{i+1}^{i} \ddot{q}_{i+1}. \tag{5.3.32}$$

and

$$\sigma_{i+1} = \begin{cases} 1 & \text{if the } (i+1)\text{-th joint is revolute} \\ 0 & \text{if the } (i+1)\text{-th joint is prismatic} \end{cases} \tag{5.3.33}$$

Therefore, when the $(i+1)$-th joint is prismatic we have to modify equation (5.3.20). In this case, we denote the resultant torque by the vector $\mathbf{v}_i^i$, i.e., we write equation (5.3.18) as

$$\mathbf{v}_i^i = \mathbf{M}_{C_i}^{i} + m_i \tilde{\mathbf{r}}_{i,i}^{i} \ddot{\mathbf{r}}_{i,i}^{i} + \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^{i} \ddot{\mathbf{s}}_{i,i+1}^{i}. \tag{5.3.34}$$

Now, following an analysis similar to that applied to equation (5.3.18) and using equation (5.3.31) instead of equation (5.3.28), we can show that equation (5.3.34) can be written as

$$\begin{aligned} \mathbf{v}_i^i &= \mathbf{K}_{O_i}^{i} \dot{\omega}_i^i + \tilde{\omega}_i^i \mathbf{K}_{O_i}^{i} \omega_i^i + \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^{i} \varsigma_{i,i+1}^{i}. \\ &= \mu_i^i + \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^{i} \varsigma_{i,i+1}^{i}. \end{aligned} \tag{5.3.35}$$

From the foregoing, the resultant torque vector at the origin of the i-th coordinate system, due to the rotational motion of the i-th augmented link can be described by a single equation as

$$\mu_i^i + (1 - \sigma_{i+1}) \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^{i} \varsigma_{i,i+1}^{i} = \mathbf{M}_{C_i}^{i} + m_i \tilde{\mathbf{r}}_{i,i}^{i} \ddot{\mathbf{r}}_{i,i}^{i} + \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^{i} \ddot{\mathbf{s}}_{i,i+1}^{i} \tag{5.3.36}$$

where σ_{i+1} is defined by equation (5.3.33).

With this preliminary result, we now assume that the links are augmented links and we proceed to modify Algorithm 5.4 so as to make it applicable to manipulators whose modeling scheme utilizes the ideas of augmented and generalized links. Since the kinematic analysis of an augmented link is the same as that of the corresponding actual link, only the dynamic recurrence relations need to be modified. Thus we proceed by

reformulating the recurrence relation for the moment vector η_i^i, which is exerted on link i by link $i-1$. To do this, we first write (5.3.8a) in its expanded form *i.e.*, we write

$$\mathbf{f}_i^i = \sum_{j=i}^{n} \mathbf{F}_{C_j}^i \tag{5.3.37}$$

where, from (5.3.7e), $\mathbf{F}_{C_j}^i = m_j \ddot{\mathbf{r}}_{0,j}^i$. Now, since $\ddot{\mathbf{r}}_{0,j}^i = \ddot{\mathbf{s}}_{0,i}^i + \ddot{\mathbf{r}}_{i,j}^i$ for $j \geq i$, we have

$$\begin{aligned}\mathbf{f}_i^i &= \sum_{j=i}^{n} m_j (\ddot{\mathbf{s}}_{0,i}^i + \ddot{\mathbf{r}}_{i,j}^i) \\ &= \sum_{j=i}^{n} m_j \ddot{\mathbf{s}}_{0,i}^i + \sum_{j=i}^{n} m_j \ddot{\mathbf{r}}_{i,j}^i \\ &= \overline{m}_i \ddot{\mathbf{s}}_{0,i}^i + \ddot{\mathbf{U}}_{O_i}^i\end{aligned} \tag{5.3.38}$$

where $\overline{m}_i$ is defined by (5.3.9) and

$$\ddot{\mathbf{U}}_{O_i}^i = \sum_{j=i}^{n} m_j \ddot{\mathbf{r}}_{i,j}^i. \tag{5.3.39}$$

Equation (5.3.39) is a consequence of (5.3.12). However, we do not need to use equation (5.3.39) for computing the vector $\ddot{\mathbf{U}}_{O_i}^i$. The vector $\ddot{\mathbf{U}}_{O_i}^i$ can be computed recursively as the following lemma shows.

Lemma 5.2 : The absolute derivative of the first moment of the i-th generalized link satisfies the following recursive relation

$$\ddot{\mathbf{U}}_{O_i}^i = \ddot{\mathbf{u}}_{O_i}^i + \mathbf{A}_{i+1} \left[\ddot{\mathbf{U}}_{O_{i+1}}^{i+1} + (1 - \sigma_{i+1}) \overline{m}_{i+1} \varsigma_{i,i+1}^{i+1} \right] \tag{5.3.40}$$

where σ_{i+1} is defined by equation (5.3.33), $\overline{m}_{i+1}$ is the 0-th moment of the $(i+1)$-th generalized link and $\varsigma_{i,i+1}^i$ is defined by equation (5.3.32)

Proof: From equation (5.3.12) we have

$$\ddot{\mathbf{U}}_{O_i}^i = m_i \ddot{\mathbf{r}}_{i,i}^i + \sum_{j=i+1}^{n} m_j \ddot{\mathbf{r}}_{i,j}^i$$

Now, since for $j > i$, $\ddot{\mathbf{r}}_{i,j}^i = \ddot{\mathbf{s}}_{i,i+1}^i + \mathbf{A}_{i+1} \ddot{\mathbf{r}}_{i+1,j}^{i+1}$, we have

$$\begin{aligned}\ddot{\mathbf{U}}_{O_i}^i &= m_i \ddot{\mathbf{r}}_{i,i}^i + \sum_{j=i+1}^{n} m_j \ddot{\mathbf{s}}_{i,i+1}^i + \mathbf{A}_{i+1} \sum_{j=i+1}^{n} m_j \ddot{\mathbf{r}}_{i+1,j}^{i+1} \\ &= m_i \ddot{\mathbf{r}}_{i,i}^i + \overline{m}_{i+1} \ddot{\mathbf{s}}_{i,i+1}^i + \mathbf{A}_{i+1} \ddot{\mathbf{U}}_{O_{i+1}}^{i+1}\end{aligned} \tag{5.3.41}$$

Further, using equations (5.3.23) and (5.3.31) we can write

$$
\begin{aligned}
m_i \ddot{\mathbf{r}}^i_{i,i} + \overline{m}_{i+1}\ddot{\mathbf{s}}^i_{i,i+1} &= \Omega^i_i \left[m_i \mathbf{r}^i_{i,i} + \overline{m}_{i+1}\mathbf{s}^i_{i,i+1} \right] + (1-\sigma_{i+1})\overline{m}_{i+1}\zeta^i_{i,i+1} \\
&= \Omega^i_i \mathbf{u}^i_{O_i} + (1-\sigma_{i+1})\overline{m}_{i+1}\zeta^i_{i,i+1} \\
&= \ddot{\mathbf{u}}^i_{O_i} + (1-\sigma_{i+1})\overline{m}_{i+1}\mathbf{A}_{i+1}\zeta^{i+1}_{i,i+1} \qquad (5.3.42)
\end{aligned}
$$

where, in the last step, equation (5.3.10) has been used. Finally, equation (5.3.40) follows on substituting equation (5.3.42) into equation (5.3.41). □

Now, for the vector $\boldsymbol{\eta}^i_i$, we have from (5.3.8b)

$$
\begin{aligned}
\boldsymbol{\eta}^i_i &= \mathbf{M}^i_{C_i} + \tilde{\mathbf{r}}^i_{i,i}\mathbf{F}^i_{C_i} + \tilde{\mathbf{s}}^i_{i,i+1}\mathbf{A}_{i+1}\mathbf{f}^{i+1}_{i+1} + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1} \\
&= \mathbf{M}^i_{C_i} + m_i \tilde{\mathbf{r}}^i_{i,i}\ddot{\mathbf{r}}^i_{0,i} + \tilde{\mathbf{s}}^i_{i,i+1}[\overline{m}_{i+1}\ddot{\mathbf{s}}^i_{0,i+1} + \ddot{\mathbf{U}}^i_{O_{i+1}}] + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1} \\
&= \mathbf{M}^i_{C_i} + m_i \tilde{\mathbf{r}}^i_{i,i}[\ddot{\mathbf{s}}^i_{0,i} + \ddot{\mathbf{r}}^i_{i,i}] + \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}[\ddot{\mathbf{s}}^i_{0,i} + \ddot{\mathbf{s}}^i_{i,i+1}] \\
&\quad + \tilde{\mathbf{s}}^i_{i,i+1}\ddot{\mathbf{U}}^i_{O_{i+1}} + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1} \\
&= \mathbf{M}^i_{C_i} + m_i \tilde{\mathbf{r}}^i_{i,i}\ddot{\mathbf{r}}^i_{i,i} + \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}\ddot{\mathbf{s}}^i_{i,i+1} \\
&\quad + [m_i \tilde{\mathbf{r}}^i_{i,i} + \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}]\ddot{\mathbf{s}}^i_{0,i} + \tilde{\mathbf{s}}^i_{i,i+1}\ddot{\mathbf{U}}^i_{O_{i+1}} + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1}.
\end{aligned}
$$

Further, using (5.3.10) and (5.3.36), we can write

$$
\begin{aligned}
\boldsymbol{\eta}^i_i &= \boldsymbol{\mu}^i_i + (1-\sigma_{i+1})\overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}\zeta^i_{i,i+1} + \tilde{\mathbf{u}}^i_{O_i}\ddot{\mathbf{s}}^i_{0,i} + \tilde{\mathbf{s}}^i_{i,i+1}\ddot{\mathbf{U}}^i_{O_{i+1}} + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1} \\
&= \boldsymbol{\mu}^i_i + \tilde{\mathbf{u}}^i_{O_i}\ddot{\mathbf{s}}^i_{0,i} + \tilde{\mathbf{s}}^i_{i,i+1}\left[\ddot{\mathbf{U}}^i_{O_{i+1}} + (1-\sigma_{i+1})\overline{m}_{i+1}\zeta^i_{i,i+1}\right] + \mathbf{A}_{i+1}\boldsymbol{\eta}^{i+1}_{i+1} \qquad (5.3.43)
\end{aligned}
$$

We can see from (5.3.43) that in computing the vector $\boldsymbol{\eta}^i_i$ we do not use the vectors $\mathbf{F}^i_{C_i}$ and $\mathbf{f}^i_i$. Therefore, we do not need to compute equations (5.3.7e) and (5.3.8a). In their place, we use equation (5.3.40). Also, we do not need to compute the vector $\mathbf{M}^i_{C_i}$, since in (5.3.43), we use the vector $\boldsymbol{\mu}^i_i$.

We are now in a position to formally outline an algorithm which efficiently computes the joint actuator torques for a rigid-link open-chain robot manipulator.

ALGORITHM 5.5

Step 0 : Initialization

$$
\mathbf{z}^i_i = |\,0\;0\;1\,|^T
$$

$$
\sigma_i = \begin{cases} 1 & \text{revolute } i\text{-th joint} \\ 0 & \text{prismatic } i\text{-th joint} \end{cases}, \qquad q_i = \begin{cases} \theta_i & \text{revolute } i\text{-th joint} \\ d_i & \text{prismatic } i\text{-th joint} \end{cases}
$$

$$
\omega^0_0 = 0\,, \quad \dot{\omega}^0_0 = 0\,, \quad \Omega^0_0\,, \quad \ddot{\mathbf{s}}^0_{0,0} = -\mathbf{g}\,, \quad \mathbf{A}_{n+1} = 0
$$

end

Step 1 : Backward recursion :- For $i = n, 1$ **do**

$$\overline{m}_i = m_i + \overline{m}_{i+1} \tag{5.3.44a}$$

$$\tilde{\mathbf{u}}_{\mathrm{O}_i}^i = m_i \tilde{\mathbf{r}}_{i,i}^i + \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^i \tag{5.3.44b}$$

$$\mathbf{K}_{\mathrm{O}_i}^i = \mathbf{I}_{\mathrm{C}_i}^i - m_i \tilde{\mathbf{r}}_{i,i}^i \tilde{\mathbf{r}}_{i,i}^i - \overline{m}_{i+1} \tilde{\mathbf{s}}_{i,i+1}^i \tilde{\mathbf{s}}_{i,i+1}^i \tag{5.3.44c}$$

$$\hat{\mathbf{K}}_{\mathrm{O}_i}^i = \frac{1}{2} tr\left[\mathbf{K}_{\mathrm{O}_i}^i\right] \mathbf{1} - \mathbf{K}_{\mathrm{O}_i}^i \tag{5.3.44d}$$

end

Step 2 : Forward recursion :- For $i = 0, n\text{-}1$ **do**

$$\omega_{i+1}^{i+1} = \mathbf{A}_{i+1}^T \omega_i^i + \sigma_{i+1} \mathbf{z}_{i+1}^{i+1} \dot{q}_{i+1} \tag{5.3.45a}$$

$$\dot{\omega}_{i+1}^{i+1} = \mathbf{A}_{i+1}^T \dot{\omega}_i^i + \sigma_{i+1}\left[\tilde{\omega}_i^{i+1} \mathbf{z}_{i+1}^{i+1} \dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1} \ddot{q}_{i+1}\right] \tag{5.3.45b}$$

$$\Omega_{i+1}^{i+1} = \tilde{\dot{\omega}}_{i+1}^{i+1} + \tilde{\omega}_{i+1}^{i+1} \tilde{\omega}_{i+1}^{i+1} \tag{5.3.45c}$$

$$\varsigma_{i,i+1}^{i+1} = (1 - \sigma_{i+1})\left(2\tilde{\omega}_i^{i+1} \mathbf{z}_{i+1}^{i+1} \dot{q}_{i+1} + \mathbf{z}_{i+1}^{i+1} \ddot{q}_{i+1}\right) \tag{5.3.45d}$$

$$\ddot{\mathbf{s}}_{0,i+1}^{i+1} = \mathbf{A}_{i+1}^T\left[\ddot{\mathbf{s}}_{0,i}^i + \Omega_i^i \mathbf{s}_{i,i+1}^i\right] + (1 - \sigma_{i+1}) \varsigma_{i,i+1}^{i+1} \tag{5.3.45e}$$

$$\tilde{\mu}_{i+1}^{i+1} = \Omega_{i+1}^{i+1} \hat{\mathbf{K}}_{\mathrm{O}_{i+1}}^{i+1} - \left[\Omega_{i+1}^{i+1} \hat{\mathbf{K}}_{\mathrm{O}_{i+1}}^{i+1}\right]^T. \tag{5.3.45f}$$

$$\mu_{i+1}^{i+1} = dual\left(\tilde{\mu}_{i+1}^{i+1}\right) \tag{5.3.45g}$$

end

Step 3 : Backward recursion :- For $i = n, 1$ **do**

$$\ddot{\mathbf{U}}_{\mathrm{O}_i}^i = \Omega_i^i \mathbf{u}_{\mathrm{O}_i}^i + \mathbf{A}_{i+1}\left[\ddot{\mathbf{U}}_{\mathrm{O}_{i+1}}^{i+1} + (1 - \sigma_{i+1}) \overline{m}_{i+1} \varsigma_{i,i+1}^{i+1}\right] \tag{5.3.46a}$$

$$\eta_i^i = \mu_i^i + \tilde{\mathbf{u}}_{\mathrm{O}_i}^i \ddot{\mathbf{s}}_{0,i}^i + \tilde{\mathbf{s}}_{i,i+1}^i\left[\ddot{\mathbf{U}}_{\mathrm{O}_{i+1}}^i + (1 - \sigma_{i+1}) \overline{m}_{i+1} \varsigma_{i,i+1}^i\right] + \mathbf{A}_{i+1} \eta_{i+1}^{i+1} \tag{5.3.46b}$$

$$\tau_i = \sigma_i (\eta_i^i \cdot \mathbf{z}_i^i) + (1 - \sigma_i) \mathbf{z}_i^i \cdot [\overline{m}_i \ddot{\mathbf{s}}_{0,i}^i + \ddot{\mathbf{U}}_{\mathrm{O}_i}^i] \tag{5.3.46c}$$

end

As we can see, in Step 1 of Algorithm 5.5, we compute the dynamic parameters for the augmented links. These parameters are configuration independent when the augmented links have rigid body characteristics and in this case can be computed off-line. An augmented link, say the i-th, is not a rigid body only when the $(i+1)$-th joint is a prismatic joint. Thus, for robot manipulators which have all joints of revolute type, e.g., a PUMA type robot, Step 1 of Algorithm 5.5 can be computed off-line. Even for robot

manipulators with one prismatic joint, e.g., a Stanford-arm type robot, Step 1 of Algorithm 5.5 can be computed practically off-line, because for this type of robot manipulators only minor modifications are needed and these can be easily incorporated in the on-line computations. Therefore, since almost all industrial robot manipulators are either of PUMA or of Stanford-arm type, we can say that Step 1 of Algorithm 5.5 can be computed off-line for almost all industrial robot manipulators in use today.

In the following, we shall consider the numerical implementation of the algorithms derived in this section. In particular, first some observations will be made about the most computationally intensive operations appearing in these algorithms. Then the implementation of Algorithm 5.5 will be looked at in more details when it is applied to robot manipulators which have all joints of the revolute type. We examine this case in more details since, as is well known, solving inverse dynamics for robot manipulators of this type is computationally more intensive than for robot manipulators which have some joints of prismatic type.

5.3.2. Implementation and Computational Considerations

In this section, we shall demonstrate how the algorithms developed earlier can be implemented efficiently. We consider two cases; robot manipulators with a general geometric structure and those for which the twist angle is, by design, either 0 or 90 degrees. We consider the latter case, since in most industrial robots manipulators the twist angles have this characteristic.

In the following, we are concerned with the numerical implementation of Algorithms 5.4 and 5.5. Therefore, to be technically correct we have to rewrite these Algorithms in terms of the corresponding coordinate matrix equations. However, as we mentioned in Chapter III, a tensor equation and its corresponding coordinate matrix equation (with respect to a Cartesian coordinate system) are formally the same. Therefore, in a coordinate matrix form these two algorithms have the same structure and appearance

and therefore there is no need to actually rewrite these algorithms in a coordinate matrix form. Based on this observation, by a slight abuse of the notation, we shall refer to the computation of the coordinate matrix, say Ω_i^i, relative to the i-th link coordinate system, of the tensor Ω_i^i as the computation of the matrix Ω_i^i.

From Algorithms 5.4 and 5.5, it is clear that the maximum number of operations required to implement them results from various matrix-vector or matrix-matrix multiplications. These matrix operations can be implemented in a straightforward manner by using general purpose standard subroutines. However, the structure of these matrices (e.g., symmetric or skew-symmetric) are standard and common for all robot manipulators that we are dealing with. Therefore, for efficient implementations, the structure of the matrices should be taken into account. Obviously, this approach does not restrict the applicability of these algorithms to a general robot manipulator.

The matrix-vector multiplications which are involved in Algorithms 5.4 and 5.5 can be categorized in the following classes.

Class (a) : consists of those operations where the matrix under consideration is a coordinate transformation matrix.

Class (b) : consists of those operations where the matrix involved is a skew-symmetric matrix.

Class (c) : consists of those operations where the matrix involved is the matrix Ω.

For a general manipulator, to implement a matrix-vector multiplication of class (a), we need 8 scalar multiplications and 4 scalar additions. For the case of manipulators, when the twist angle α is either 0 or 90 degrees, we need only 4 multiplications and 2 additions. A matrix-vector multiplication of class (b) can be implemented in 6 scalar multiplications and 3 scalar additions; and finally, for the multiplication of class (c), we need 9 scalar multiplications and 6 scalar additions.

Also, the following observations are important for an efficient implementation. To

compute the matrix Ω_{i+1}^{i+1}, we require a matrix-matrix multiplication. Moreover, since the product $\tilde{\omega}_{i+1}^{i+1}\tilde{\omega}_{i+1}^{i+1}$ is symmetric, and the matrices $\tilde{\omega}_{i+1}^{i+1}$ and $\dot{\tilde{\omega}}_{i+1}^{i+1}$ are skew-symmetric, we can do this with 6 scalar multiplications and 9 scalar additions. Similarly, since the matrix $\tilde{\mu}_{i+1}^{i+1}$ (or $\tilde{M}_{C_{i+1}}^{i+1}$) is skew-symmetric, only three of its elements need to be computed. Thus, by taking into account the symmetry of $\mathcal{K}_{O_{i+1}}^{i+1}$, we can compute the skew-symmetric matrix $\tilde{\mu}_{i+1}^{i+1}$ with only 15 scalar multiplications and 15 scalar additions. Moreover, for implementing the dual operator, we do not need any computations because of the one-to-one correspondence between a skew-symmetric matrix and its dual vector or vector invariant. Finally, since $\mathbf{z}_i^i = [0\ 0\ 1]^T$, evaluating the scalar torque (or force) τ_i does not require any operations for Algorithm 5.4. In Algorithm 5.5 we need only 1 multiplication and 1 addition if the joint is prismatic.

Besides these general observations, for an even more efficient implementation of these algorithms we note the following : For most of the equations, the initial conditions are zero. Therefore, the first cycle (iteration) in Steps 2 and 3 can be computed with almost no computational cost. For example, since $\Omega_0^0 = 0$, the functional expression for the vector $\ddot{\mathbf{s}}_{0,1}^1$ can be easily defined, specially when the gravity vector has only one non-zero component. Thus, with proper initial conditions for the recursive equations, the computational cost of implementing these algorithms is reduced considerably. This is obviously also true for other algorithms solving inverse dynamics. However, for these particular algorithms, the effort for performing the first cycle (or even the second) by hand is tolerable. Also, as we have mentioned above, the general organization of the computations is important for an efficient implementation. Thus, for example in Step 3, when evaluating η_i^i for $i=-1$, only the last component of that vector needs to be evaluated. Finally, knowledge of the geometry (zero components of various vectors or matrices) of a particular class of robot manipulators can reduce considerably the cost of computation.

A breakdown of the number of scalar multiplications and additions required by each equation of Algorithm 5.5, when this algorithm is applied to a robot manipulator which has all joints of revolute type, is given in the following table.

Steps 2 & 3	General manipulator		Manipulator with $\alpha=0^o$ or 90^o	
Equation	Multiplications	Additions	Multiplications	Additions
5.3.45a	$8(n-1)$	$5(n-1)$	$4(n-1)$	$3(n-1)$
5.3.45b	$10(n-1)$	$7(n-1)$	$6(n-1)$	$5(n-1)$
5.3.45c	$6(n-1)+1$	$9(n-1)$	$6(n-1)$	$9(n-1)$
5.3.45e	$17(n-1)+3$	$13(n-1)$	$13(n-1)$	$11(n-1)$
5.3.45f	$15(n-1)+5$	$15(n-1)+3$	$15(n-1)+5$	$15(n-1)+3$
5.3.45g	0	0	0	0
5.3.46a	$17(n-1)+9$	$13(n-1)+6$	$13(n-1)+9$	$11(n-1)+6$
5.3.46b	$20(n-1)+6$	$19(n-1)+6$	$16(n-1)+6$	$17(n-1)+6$
5.3.46c	0	0	0	0
Total	$93(n-1)+24$	$81(n-1)+15$	$73(n-1)+23$	$71(n-1)+15$
$n=6$	489	420	388	370

Table 5.1 : Operations Count for Implementing Algorithm 5.5.

For this implementation of Algorithm 5 5 we have assumed, as is usually the practice, that the equations in the forward and backward recursions for $i = 0$ and $i = n$, respectively, are computed outside the main loops. Therefore, there are only $n - 1$ cycles to be performed in the actual implementation. For these iterations no effort has been made to reduce further the computations, since we wish to keep customization at a minimum. However, some saving in the computations are obvious and can be easily taken into account for a more efficient implementation. Thus, for example, in Step 2 :

a) in computing ω_2^2, we can save 4 multiplications and 4 additions, since $\omega_1^1 = [0, 0, \dot{q}_1]^T$,

b) in computing $\dot{\omega}_2^2$, we can save 5 multiplications and 4 additions, since $\dot{\omega}_1^1 = [0, 0, \ddot{q}_1]^T$, and

c) in computing $\ddot{s}_{0,2}^2$, we can save 5 multiplications and 5 additions, since

$$\Omega_1^1 = \begin{bmatrix} -\dot{q}_1^2 & -\ddot{q}_1 & 0 \\ \ddot{q}_1 & -\dot{q}_1^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Also, as we mentioned above in Step 3, to compute τ_1 we do not need to compute all the entries in the vector η_i^1. Only the last entry is needed. Thus we can save 9 multiplications and 9 additions in equation (5.3.46a), since we need to compute only the term $\ddot{\mathbf{U}}^i_{O_{i+1}} = \mathbf{A}_{i+1}\ddot{\mathbf{U}}^{i+1}_{O_{i+1}}$, and 12 multiplications and 12 additions in equation (5.3.46b).

A breakdown of the number of scalar multiplications and additions required by each equation of Algorithm 5.5 for this "semi-customized" implementation, which is valid for $n \geq 2$, is given in the following table.

Steps 2 & 3	General manipulator		Manipulator with $\alpha=0^o$ or 90^o	
Equation	Multiplications	Additions	Multiplications	Additions
5.3.45a	$8n-12$	$5n-9$	$4n-7$	$3n-5$
5.3.45b	$10n-15$	$7n-11$	$6n-10$	$5n-8$
5.3.45c	$6n-5$	$9n-9$	$6n-5$	$9n-9$
5.3.45e	$17n-19$	$13n-16$	$13n-16$	$11n-14$
5.3.45f	$15n-13$	$15n-14$	$15n-13$	$15n-14$
5.3.45g	0	0	0	0
5.3.46a	$17n-17$	$13n-16$	$13n-13$	$11n-3$
5.3.46b	$20n-27$	$19n-25$	$16n-20$	$17n-22$
5.3.46c	0	0	0	0
Total	$93n-108$	$81n-100$	$73n-84$	$71n-75$
$n=6$	450	386	354	351

Table 5.2 : Operations Count for Implementing Algorithm 5.5, (Valid for $n \geq 2$).

The figures, in Tables 5.1 and 5.2, represent the operations count for steps 2 and 3 of Algorithm 5.5 for computing *all* the joint actuator torques for a particular point along a trajectory. It may be noted that the computational efficiency of this algorithm results, mainly, from the use of the tensor Ω in evaluating linear accelerations and Euler's equation. For example, the implementation of Euler's equation in Algorithm 5.3, i.e., in its traditional vector formulation, requires $24(n-1)+8$ scalar multiplications and $18(n-1)+6$ scalar additions whereas the corresponding computations in Algorithms 5.4 and 5.5, where Euler's equation is stated in its tensor formulation, require $15(n-1)+5$ scalar multiplications and $15(n-1)+3$ scalar additions. This clearly indicates that the

tensor representation for the angular velocity leads to a tensor description for the rigid body angular motion (Euler's equation) which is computationally far more efficient that the classical vector description. Therefore, the question which has been raised in the previous section and is concerned with the computational efficiency of possible descriptions of rigid body angular motion has been answered here in the affirmative.

For the sake of comparison, we have given in the following table the operations counts for a number of algorithms reported in the literature for solving the problem of inverse dynamics.

Algorithm	Multiplications	Additions
Hollerbach (4×4) [14]	$830n - 592$ (4388)†	$675n - 464$ (3586)
Hollerbach (3×3) [14]	$412n - 277$ (2195)	$320n - 201$ (1719)
Vucobratovic et al. [24]	$\frac{3}{2}n^3+\frac{35}{2}n^2+9n-16$ (992)	$\frac{7}{6}n^3+\frac{23}{2}n^2+\frac{64}{3}n-28$ (776)
Luh et al. [14]	$150n - 48$ (852)	$131n - 48$ (738)
Craig [42]	$126n - 99$ (657)	$106n - 92$ (544)
Khosla and Neuman [52]	$123n - 60$ (678)	$96n - 55$ (521)
Li [26]	$120n - 104$ (616)	$98n - 94$ (494)
Ma [63]	$109n - 109$ (545)††	$95n - 108$ (462)
Khalil et al. [50]	$105n - 92$ (538)	$94n - 86$ (478)
Alg. 5.4 in this thesis	$96n - 77$ (499)	$84n - 70$ (434)
Alg. 5.5 in this thesis	$93n - 69$ (489)	$81n - 66$ (420)
Alg. 5.5 in this thesis	$93n - 108$ (450)††	$81n - 100$ (386)

† Number of Operations for $n = 6$, †† Implementation Valid for $n \geq 2$

Table 5.3 : Comparison of Operations Counts for Algorithms Which Solve the IDP.

The computational effort required for a particular algorithm, shown in the above table depends on the degree of optimization in the operations involved in its implementation. Therefore, for an accurate comparison of the relative performance of these algorithms it is required that they be implemented fairly. Thus for example, the algorithm by Khalil, Kleinfinger and Gautier which is included in Table 5.3 is implemented based on a recursive symbolic procedure and on an analysis of the inertial parameters of the

links, both of which help to reduce the number of operations. However, despite such specialized features in some of the other algorithms presented in Table 5.3, the algorithms derived in this thesis have a significantly higher computational efficiency. This has been obtained primarily through the tensor representation of the angular velocity and thus a tensor description for the angular rigid body motion, and not because of a specialized implementation.

5.4 THE USE OF EULER-LAGRANGE'S AND KANE'S FORMULATIONS IN DERIVING ALGORITHM 5.5

In this section, we shall demonstrate that the computationally efficient algorithm (Algorithm 5.5), which in the previous section is derived based on the Newton-Euler equations, can also be derived by using the Euler-Lagrange or Kane's dynamic equations of motion. This demonstration will make clear that the computational efficiency of an algorithm for solving inverse dynamics is indeed completely independent from the particular procedure of classical mechanics which has been used to derive that algorithm. For the sake of simplicity in this demonstration we consider manipulators with revolute joints only. However, the analysis can be easily extended to include prismatic joints also.

5.4.1 The Euler-Lagrange Formulation

As we mentioned in Section 5.2, a number of algorithms for solving inverse dynamics have been derived based on the Euler-Lagrange formulation and among them is the recursive Algorithm 5.2. This algorithm has been devised by Hollerbach [14] who was aiming to implement efficiently the following equation

$$\tau_i = \sum_{j=i}^{n} \left\{ tr\left[m_j \frac{\partial \mathbf{a}_{0,j}}{\partial q_i} \ddot{\mathbf{g}}_{0,j}^T + m_j \frac{\partial \mathbf{a}_{0,j}}{\partial q_i} (\mathbf{r}_{j,j}^j)^T \ddot{\mathbf{W}}_j^T + m_j \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j \ddot{\mathbf{a}}_{0,j}^T + \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \right] - m_j \mathbf{g}^T \frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{r}_{i,j}^i \right\}, \qquad i = 1, \cdots, n \qquad (5.4.1)$$

which has been derived from the Euler-Lagrange dynamical equations of motion. However, if we can show that the right-hand side of the equation (5.4 1) is exactly the same as the right-hand side of the equation

$$\tau_i = \mathbf{z}_i^i \cdot \boldsymbol{\eta}_i^i \ , \qquad i = 1, \cdots, n \, , \tag{5.4.2}$$

which is simply equation (5.3.46c) of Algorithm 5.5 (stated here for revolute jc.nts only), then it is clear that the Algorithm 5.5 can be derived from the Euler-Lagrange equations instead the Newton-Euler ones. Therefore, our aim in this section is to show that equation (5.4.1) can assume the same formulation as equation (5.4.2). We proceed as follows :

Our first bjective is to eliminate from equation (5.4.1) the term which contains the effects of gravity. To achieve this, we notice that from a simple comparison of equations (A.7) and (A.10) in Appendix A, we have

$$\frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{r}_{i,j}^i = \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} + \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j \, .$$

Therefore, using the fact that the dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$ satisfies the equation

$$\mathbf{a} \cdot \mathbf{b} = tr\left[\mathbf{a}\,\mathbf{b}^T\right], \tag{5.4.3}$$

we can write the term which contains the gravitational effects in equation (5.4.1) as follows,

$$m_j \mathbf{g}^T \frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{r}_{i,j}^i = tr\left[m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} \mathbf{g}^T\right] + tr\left[m_j \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j \mathbf{g}^T\right]. \tag{5.4.4}$$

Now, substitution of equation (5.4.4) into (5.4.1) yields

$$\tau_i = \sum_{j=i}^{n} tr\left[m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} (\ddot{\mathbf{s}}_{0,j} - \mathbf{g})^T + m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} (\mathbf{r}_{j,j}^j)^T \ddot{\mathbf{W}}_j^T \right.$$
$$\left. + m_j \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j (\ddot{\mathbf{s}}_{0,j} - \mathbf{g})^T + \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{o_j}^j \ddot{\mathbf{W}}_j^T \right] , \qquad i = 1, \cdots, n$$

$$= \sum_{j=i}^{n} tr\left[m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} \ddot{\mathbf{s}}_{0,j}^T + m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} (\mathbf{r}_{j,j}^j)^T \ddot{\mathbf{W}}_j^T \right.$$

$$\left. + m_j \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j \ddot{\mathbf{s}}_{0,j}^T + \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \right] \quad i = 1, \cdots, n \tag{5.4.5}$$

where the initial condition $\ddot{\mathbf{s}}_{0,0}$ for the vector $\ddot{\mathbf{s}}_{0,j}$ is now equal to $-\mathbf{g}$, instead of being zero as is usually done in the Lagrangian formulation. Furthermore, since

$$\ddot{\mathbf{r}}_{j,j} = \ddot{\mathbf{W}}_j \mathbf{r}_{j,j}^j \qquad \text{and} \qquad \ddot{\mathbf{r}}_{0,j} = \ddot{\mathbf{s}}_{0,j} + \ddot{\mathbf{r}}_{j,j},$$

equation (5.4.5) can be simplified to

$$\tau_i = \sum_{j=i}^{n} tr\left[m_j \frac{\partial \mathbf{s}_{0,j}}{\partial q_i} \ddot{\mathbf{r}}_{0,j}^T + m_j \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{r}_{j,j}^j \ddot{\mathbf{s}}_{0,j}^T + \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \right], \quad i = 1, \cdots, n. \tag{5.4.6}$$

Moreover, as we have shown in Appendices B and C, we can write

$$\frac{\partial \mathbf{s}_{0,j}}{\partial q_i} = \tilde{\mathbf{z}}_i \mathbf{s}_{i,j} \tag{5.4.7}$$

$$\frac{\partial \mathbf{W}_j}{\partial q_i} = \tilde{\mathbf{z}}_i \mathbf{W}_j \tag{5.4.8}$$

$$tr\left[\frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{O_j} \ddot{\mathbf{W}}_j^T \right] = \mathbf{z}_i \cdot \mathbf{M}_{O_j} \tag{5.4.9}$$

Therefore, using equations (5.4.7)-(5.4.9) and equation (5.4.3), equation (5.4.6) can be simplified further to yield

$$\tau_i = \sum_{j=i}^{n} \left\{ m_j \tilde{\mathbf{z}}_i \mathbf{s}_{i,j} \cdot \ddot{\mathbf{r}}_{0,j} + m_j \tilde{\mathbf{z}}_i \mathbf{r}_{j,j} \cdot \ddot{\mathbf{s}}_{0,j} + \mathbf{z}_i \cdot \mathbf{M}_{O_j} \right\}, \quad i = 1, \cdots, n. \tag{5.4.10}$$

Finally, since for any vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ we have

$$\tilde{\mathbf{a}}\,\mathbf{c} \cdot \mathbf{b} = \mathbf{a} \cdot \tilde{\mathbf{c}}\,\mathbf{b},$$

equation (5.4.10) can be written as

$$\tau_i = \mathbf{z}_i \cdot \sum_{j=i}^{n} \left\{ m_j \tilde{\mathbf{s}}_{i,j} \ddot{\mathbf{r}}_{0,j} + m_j \tilde{\mathbf{r}}_{j,j} \ddot{\mathbf{s}}_{0,j} + \mathbf{M}_{O_j} \right\}, \qquad i = 1, \cdots, n,$$

from which we get

$$\tau_i = \mathbf{z}_i^i \cdot \sum_{j=i}^{n} \left\{ m_j \tilde{\mathbf{s}}_{i,j}^i \ddot{\mathbf{r}}_{0,j}^i + m_j \tilde{\mathbf{r}}_{j,j}^i \ddot{\mathbf{s}}_{0,j}^i + \mathbf{M}_{O_j}^i \right\}, \quad i = 1, \cdots, n \tag{5.4.11}$$

since the dot product is invariant under coordinate transformations.

Now, for $i = 1, 2, \cdots, n$, let us define the vectors

$$\upsilon_i^i = \sum_{j=i}^{n} \left\{ m_j \tilde{s}_{i,j}^i \ddot{r}_{0,j}^i + m_j \tilde{r}_{j,j}^i \ddot{s}_{0,j}^i + M_{O_j}^i \right\}. \tag{5.4.12}$$

We shall show that for $i = 1, \cdots, n$, the vector υ_i^i is equal to the following vector

$$\eta_i^i = \mu_i^i + \tilde{u}_{O_i}^i \ddot{s}_{0,i}^i + \tilde{s}_{i,i+1}^i \ddot{U}_{O_{i+1}}^i + \eta_{i+1}^i \tag{5.4.13}$$

which results from equation (5.3.46b) of Algorithm 5.5 when the $(i+1)$-th joint is assumed to be of revolute type.

First, since for any i the vector $s_{i,i}^i$ is equal to zero by definition, we notice that the vector υ_i^i can also be written as

$$\begin{aligned} \upsilon_i^i &= m_i \tilde{r}_{i,i}^i \ddot{s}_{0,i}^i + M_{O_j}^i + \sum_{j=i+1}^{n} \left\{ m_j \left[\tilde{s}_{i,i+1}^i + \tilde{s}_{i+1,j}^i \right] \ddot{r}_{0,j}^i + m_j \tilde{r}_{j,j}^i \ddot{s}_{0,j}^i + M_{O_j}^i \right\} \\ &= m_i \tilde{r}_{i,i}^i \ddot{s}_{0,i}^i + M_{O_j}^i + \tilde{s}_{i,i+1}^i \sum_{j=i+1}^{n} m_j \ddot{r}_{0,j}^i + \upsilon_{i+1}^i. \end{aligned} \tag{5.4.14}$$

Moreover, since

$$\begin{aligned} \sum_{j=i+1}^{n} m_j \ddot{r}_{0,j}^i &= \sum_{j=i+1}^{n} m_{i+1} \left(\ddot{s}_{0,i}^i + \ddot{s}_{i,i+1}^i + \ddot{r}_{i+1,j}^i \right) \\ &= \overline{m}_{i+1} \ddot{s}_{0,i}^i + \overline{m}_{i+1} \ddot{s}_{i,i+1}^i + \sum_{j=i+1}^{n} m_j \ddot{r}_{i+1,j}^i \\ &= \overline{m}_{i+1} \ddot{s}_{0,i}^i + \overline{m}_{i+1} \ddot{s}_{i,i+1}^i + \ddot{U}_{O_{i+1}}^i, \end{aligned} \tag{5.4.15}$$

equation (5.4.14) can be written as

$$\upsilon_i^i = (m_i \tilde{r}_{i,i}^i + \overline{m}_{i+1} \tilde{s}_{i,i+1}^i) \ddot{s}_{0,i}^i + M_{O_i}^i + \overline{m}_{i+1} \tilde{s}_{i,i+1}^i \ddot{s}_{i,i+1}^i + \tilde{s}_{i,i+1}^i \ddot{U}_{O_i}^i + \upsilon_{i+1}^i. \tag{5.4.16}$$

Now, using equations (5.3.10) and (5.3.18) we can write

$$\tilde{u}_{O_i}^i = m_i \tilde{r}_{i,i}^i + \overline{m}_{i+1} \tilde{s}_{i,i+1}^i$$

and

$$\mu_i^i = M_{O_i}^i + \overline{m}_{i+1} \tilde{s}_{i,i+1}^i \ddot{s}_{i,i+1}^i,$$

respectively. Therefore, equation (5.4.16) can be written in the following form

$$\upsilon_i^i = \tilde{\mathbf{u}}_{O_i}^i \ddot{\mathbf{s}}_{0,i}^i + \mu_i^i + \tilde{\mathbf{s}}_{i,i+1}^i \ddot{\mathbf{U}}_{O_{i+1}}^i + \upsilon_{i+1}^i, \tag{5.4.17}$$

which shows that indeed $\upsilon_i^i = \eta_i^i$ for $i = 1, \cdots, n$. From the foregoing, equations (5.4.1) and (5.4.2) are equivalent and this shows that Algorithm 5.5 can also be derived by using the Lagrangian formulation instead of the Newton-Euler equations.

5.4.2 Kane's Formulation

In this section, we shall show that, as with the Lagrangian formulation, Algorithm 5.5 can also be derived using Kane's equations.

As we mentioned in Section 5.2, Ma has shown [63] that, based on Kane's equations, we can determine the actuator torques τ_i for a robot manipulator with all joints of revolute type from the following equation

$$\tau_i = \mathbf{z}_i^i \cdot \sum_{j=i}^{n} \left\{ \mathbf{M}_{C_j}^i + m_j \tilde{\mathbf{r}}_{i,j}^i \ddot{\mathbf{r}}_{0,j}^i \right\}, \qquad i = 1, \cdots, n. \tag{5.4.18}$$

To evaluate equation (5.4.18) Ma has proposed an algorithm which has the same structure as Algorithm 5.3. As with the Euler-Lagrange case, we shall show that equation (5.4.18) is equivalent to equation (5.4.2). To do this let us define the following vector for $i = 1, \cdots, n$.

$$\mathbf{h}_i^i = \sum_{j=i}^{n} \left\{ \mathbf{M}_{C_j}^i + m_j \tilde{\mathbf{r}}_{i,j}^i \ddot{\mathbf{r}}_{0,j}^i \right\}. \tag{5.4.19}$$

As before, our aim is to show that $\mathbf{h}_i^i = \eta_i^i$ for all i, $i = 1, \cdots, n$, where η_i^i is defined by equation (5.4.13). By expanding the summation in equation (5.4.19), we obtain the following equation after a few manipulations

$$\mathbf{h}_i^i = \mathbf{M}_{C_i}^i + m_i \tilde{\mathbf{r}}_{i,i}^i \ddot{\mathbf{r}}_{0,i}^i + \tilde{\mathbf{s}}_{i,i+1}^i \sum_{j=i+1}^{n} m_j \ddot{\mathbf{r}}_{0,j}^i + \mathbf{h}_{i+1}^i.$$

Furthermore, using equations (5.3.10), (5.3.18) and (5.4.15), the above equation may be

simplified to

$$\mathbf{h}_i^i = \boldsymbol{\mu}_i^i + \tilde{\mathbf{u}}_{O_i}^i \ddot{\mathbf{s}}_{O,i}^i + \tilde{\mathbf{s}}_{i,i+1}^i \ddot{\mathbf{U}}_{O_{i+1}}^i + \mathbf{h}_{i+1}^i \tag{5.4.20}$$

which obviously shows that $\mathbf{h}_i^i = \boldsymbol{\eta}_i^i$ for all i. Thus, Algorithm 5.5 can also be derived from Kane's equations.

From the foregoing, starting from the Euler-Lagrange or Kane's equations and following a proper analysis we can derive not only equivalent but exactly the same formulations for the vector of the generalized forces, namely, equation (5.4.2). This equation was derived earlier (in Section 5.3) from the Newton-Euler formulation. Therefore, independently of which approaches from classical mechanics are used to derive the dynamical equations of motion, we can devise the same computational algorithm for their implementation. This result clearly indicates that apart from personal preference or experience there is nothing to be gained, in terms of computational efficiency, by choosing one approach over another for solving the manipulator IDP. However, it should be noted that the choice of a particular approach is important because it determines the nature of the analysis and the amount of effort needed to devise an algorithm for solving the IDP. Moreover, the availability of various approaches for deriving the dynamic equations of motion results in a greater variety of viewpoints. When properly used, this may help in clarifying what is essential for an efficient numerical implementation of the equations of motion.

5.5 Concluding Remarks

In this chapter, the Cartesian tensor methodology, developed earlier in chapter IV, has been used to analyze the dynamics equations of motion of rigid-links open-chain robot manipulators. Also, the ideas of augmented and generalized links have been used in the underlying modeling scheme for the said class of robot manipulators. Based on this modeling scheme, we proposed an algorithm for computing the problem of inverse manipulator dynamics which allows us to compute several configuration independent

parameters of the manipulator off-line. The same time, the Cartesian tensor formulation for the quantities to be computed on-line enables us to propose implementations for this algorithm which are computationally very efficient. In fact, we have shown, by comparing the computational complexity of his algorithm with that of other existing ones, that the proposed algorithm is computationally the most efficient non-customized algorithm which is available today for solving the problem of inverse manipulator dynamics. The computational efficiency of this algorithm has been achieved mainly because a tensor representation, instead of a vector one, has been used for the angular velocity. Finally, in this chapter, we have shown that the Newton-Euler, Euler-Lagrange or Kane's formulations of robot dynamics, with proper analysis, can lead us into the same computational algorithms. Thus, we have established that from an algorithmic point of view, the solution of the inverse dynamics problem does not depend on which of these formulations is used for deriving the equation of motion. This result clearly indicates that apart from personal preference or experience there is nothing to be gained, in terms of computational efficiency, by choosing one approach over another for solving the problem of inverse dynamics for rigid-links open-chain robot manipulators.

5.6 REFERENCES

[1] O. Khatib, "Dynamic Control of Manipulators in Operational Space", *6th IFTOMM Congress on Theory of Machines and Mechanisms,* New Delhi, Dec. 15-20, pp. 1-10, 1983.

[2] T. Yoshikawa, "Dynamic Hybrid Position/Force Control of Robot Manipulators Description of Hand Constraints and Calculation of Joint Driving Force", *Proc. 1986 IEEE Int. Conf. Robotics and Automation,* San Francisco, CA, pp. 1393-1398, Apr. 1986.

[3] P. Misra, R. V. Patel and C. A. Balafoutis, "Robust Control of Robot Manipulators Using Linearized Dynamic Models", *Recent Trends in Robotics : Modeling, Control and Education,* M. Jamshidi, J. Y. S. Luh and M. Shahinpoor, Eds., North-Holland, Elsevier Science Publishing Co., Inc., New York, 1986.

[4] P. Misra, R. V. Patel and C. A. Balafoutis, "Robust Control of Robot manipulators in Cartesian Space", *Proc. American Control Conference,* pp. 1351-1356, Atlanta, Georgia, June 15-17, 1988.

[5] M. W. Spong, J. S. Thorp and J. M. Kleinwaks, "The Control of Robot Manipulators with Bounded Input", *IEEE Trans. on Automatic Control,* Vol. AC-31, No. 6, pp. 483-490, 1986.

[6] K. G. Shin and N. D. Mckay, "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators", *IEEE Trans. on Automatic Control,* Vol. AC-31, No. 6, pp. 491-500, 1986.

[7] H. H. Tan and R. B. Potts, "Minimum Time Trajectory Planner for the Discrete Dynamic Robot Model With Dynamic Constraints", *IEEE J. of Robotics and Automation,* Vol. RA-4, No. 2, pp. 174-185, 1988.

[8] J. M. Hollerbach, "Dynamic Scaling of Manipulator Trajectories", *ASME J. of Dynamic Systems, Measurement, and Control,* Vol. 106, pp. 102-106, 1984.

[9] T. Yoshikawa, "Dynamic Manipulability of Robot Manipulators", *Journal of Robotic Systems,* Vol. 2, No. 1, pp. 113-124, 1985.

[10] J. J. Murray and C. P. Neuman, "ARM : An Algebraic Robot Dynamic Modeling Program", *Proc. 1st Int. IEEE Conf. on Robotics,* pp. 103-114, Atlanta, GA, Mar. 13-15, 1984.

[11] A. P. Tzes, S. Yurkovich and F. D. Langer, "A Symbolic Manipulation Package for Modeling of Rigid or Flexible Manipulators", *Proc. 1986 IEEE Int. Conf. Robotics and Automation,* Philadelphia, PA, pp. 1526-1531, Apr. 1988.

[12] J. J. Uicker, *On the Dynamic Analysis of Spatial Linkages Using* 4×4 *matrices,* Ph.D. Dissertation, Northwestern University, August 1965.

[13] M. E. Kahn, *The Near Minimum Time Control of Open Articulated Kinematic Chains,* Ph.D. Thesis, Stanford University, 1969.

[14] J. M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC-10, no. 11, pp. 730-736, 1980.

[15] R. Paul, "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm", *A. I. Memo. 177,* Stanford Artificial Intelligence Lab., Sept. 1972.

[16] A. K. Bejczy, "Robot Arm Dynamics and Control", *Memo. 33-669,* Jet Propulsion Labs. Tech. Feb. 1974.

[17] M. Brady *et al.*, Eds., *Robot Motion : Planning and Control,* MIT Press, Cambridge, MA, 1982.

[18] J. S. Albus, "A New Approach to Manipulator Control : The Cerebellar Model Articulation Controller (CMAC)", *IEEE J. Dynamics Systems, Measurement, Control,* Vol. 97, pp. 270-277, 1975.

[19] M. H. Raibert, "Analytical Equations vs. table Look-up for Manipulation : A Unifying Concept", *Proc IEEE Conf. Decision and Control,* New Orleans, pp. 576-579, Dec. 1977.

[20] B. K. P. Horn and M. H. Raibert, "Configuration Space Control", *The Industrial Robot,* pp. 69-73, June, 1978.

[21] R. C. Waters, "Mechanical Arm Control", M.I.T. Artificial Intelligence Lab. *Memo. 549,* Oct. 1979.

[22] I. J. Wittenburg, *Dynamics of Systems of Rigid Bodies,* B. G. Teubner, Stuttgart, 1977.

[23] M. Renaud, "An Efficient Iterative Analytical Procedure for Obtaining a Robot manipulator Dynamic Model", *Proc. of First International Symp. of Robotics Research,* Bretton Woods, New Hampshire, pp. 749-762, 1983.

[24] M. Vucobratovic, S. Li and N. Kircanski, "An Efficient Procedure for Generating Dynamic Manipulator Models", *Robotica,* Vol. 3, No. 3, pp. 147-152, 1985.

[25] J. W. Burdick, "An Algorithm for Generation of Efficient Manipulator Dynamic Equations, *Proc. 1986 IEEE Int. Conf. Robotics and Automation,* San Francisco, CA, pp. 212-218, Apr. 1986.

[26] C. J. Li, "A Fast Computational Method of Lagrangian Dynamics for Robot Manipulators", *Int. J. of Robotics and Automation,* Vol. 3, No. 1, pp. 14-20, 1988.

[27] J. Angeles and S. K. Lee, "The Formulation of Dynamical Equations of Holonomic Mechanical Systems Using a Natural Orthogonal Complement", *ASME J. of Applied Mechanics,* Vol. 55, pp. 243-244, 1988.

[28] J. Angeles and S. K. Lee, "Dynamic Modeling of Holonomic Mechanical Systems Using a Natural Orthogonal Complement, Part I : Formulation, Part II : Applications," *Proc. 9th Symposium on Engineering Applications of Mechanics,* pp. 615-630, London, Ontario, May 29-31, 1988.

[29] C. P. Neuman and J. J. Murray, "The Complete Dynamic Model and Customized algorithms of the Puma Robot", *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC-17, No. 4, pp. 635-644, 1987.

[30] S. Ramos and P. K. Khosla, "Scheduling Parallel Computation of Inverse Dynamics Formulation" *Robotics and Manufacturing : Recent Trends in Research, Education, and Applications,* M. Jamshidi, J. Y. S. Luh, H. Seraji and G. P. Starr, Eds., ASME Press, New York, 1988.

[31] J. J. Murray and C. P. Neuman, "Organizing Customized Robot Dynamics Algorithms for Efficient Numerical Evaluation", *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC-18, No. 1, pp. 115-125, 1988.

[32] K. Youcef-Toumi and H. Asada, "The Design of Open-Loop Manipulator Arms With Decoupled and Configuration-Invariant Inertia Tensors", *Proc. 1986 IEEE Int. Conf. Robotics and Automation,* San Francisco, CA, pp. 2018-2026, Apr. 1986.

[33] D. C. H. Yang and S. W. Tzeng, "Simplification and Linearization of Manipulator Dynamics by the Design of Inertia Distribution", *The Int. Journal of Robotics Research,* Vol. 5, No. 3, pp. 120-128, 1986.

[34] W. W. Hooker and G. Margulies, "The Dynamical Attitude Equations for an n-Body Satellite", *J. Astronautical Sciences,* Vol. 12, No. 4, pp. 123-128, 1965.

[35] Y. Stepanenko and M. Vucobratovic, "Dynamics of Articulated Open-Chain Active Mechanisms", *Mathematical Biosciences,* Vol. 28, pp. 137-170, 1976.

[36] M. Vucobratovic, "Dynamics of Active Articulated Mechanisms and Synthesis of Artificial Motion", *Mechanism and Machine Theory,* Vol. 13, pp. 1-56, 1978.

[37] J. Y. L. Ho, "Direct Path Method for Flexible Multibody Spacecraft Dynamics", *AIAA J. Spacecraft and Rockets* , Vol. 14, No. 2, pp. 102-110, 1977.

[38] P. C. Hughes, "Dynamics of a Chain of Flexible bodies", *J. Astronautical Sciences,* Vol. 27, No. 4, pp. 359-380, 1979.

[39] D. E. Orin, R. B. McGhee, M. Vucobratovic, and G. Hartoch, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods", *Mathematical Biosciences,* Vol. 43, No. 1/2, pp. 107-130, 1979.

[40] J. Y. S. Luh, M. W. Walker, and R. P. Paul, "On-Line Computational Scheme for Mechanical Manipulators", *ASME J. Dyn. Syst. Meas. and Contr.,* Vol. 102, pp. 69-79, 1980.

[41] W. M. Silver, "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators", *Int. J. Robotics Research,* Vol. 1, pp. 60-70, 1982.

[42] J. J. Craig, *Introduction to Robotics : Mechanics & Control,* Addison-Wesley, Reading, MA, 1986.

[43] K. Kazerounian and K. C. Gupta, "Manipulator Dynamics Using the Extended Zero reference Position Description", *IEEE J. Robotics and Automation,* Vol. RA-2, No. 4, pp. 221-224, 1986.

[44] L. T. Wang and B. Ravani, "Recursive Computations of Kinematics and Dynamics Equations for Mechanical Manipulators", *IEEE J. Robotics and Automation,* Vol. RA-1, No. 3, pp. 124-131, 1985.

[45] W. Khalil and J. F. Kleinfinger, "Minimum Operations and Minimum Parameters of the Dynamic Models of Tree Structure Robots", *IEEE J. Robotics and Automation,* Vol. RA-3, No. 6, pp. 517-526, 1987.

[46] C. A. Balafoutis, P. Misra and R. V. Patel, "A Cartesian Tensor Approach for Fast Computation of Manipulator Dynamics", *Proc. 1986 IEEE Int. Conf. Robotics and Automation*, Philadelphia, PA, pp 1348-1353, Apr. 1988.

[47] C. A. Balafoutis, R. V. Patel and P. Misra, "Efficient Modeling and Computation of Manipulator Dynamics Using Orthogonal Cartesian Tensors", *IEEE Journal of Robotics and Automation,* Vol. 4, No. 6, pp. 665-676, 1988.

[48] R. Featherstone, *Robot Dynamics Algorithms,* Kluwer Academic Publishers, Boston MA, 1987.

[49] M. Renaud, "Quasi-Minimal Computation of the Dynamic Model of a Robot Manipulator Utilizing the Newton-Euler Formalism and the Notion of Augmented Body", *Proc. 1986 IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, pp. 1677-1682, Apr. 1987.

[50] W. Khalil and J. F. Kleinfinger, and M. Gautier, "Reducing the Computational Burden of the Dynamic Models of Robots", *Proc. 1986 IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 525-531, Apr. 1986.

[51] B. Armstrong, O. Khatib, and J. Burdick, "The Explicit Dynamic Model and Inertia Parameters of the PUMA 560 Arm", *Proc. 1986 IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 510-518, Apr. 1986.

[52] P. K. Khosla and C. P. Neuman, "Computational Requirements of Customized Newton-Euler Algorithms", *J. of Robotic Systems,* Vol. 2, No. 3, pp. 309-327, 1985.

[53] H. Kasahara and S. Narita, "Parallel Processing of Robot-Arm Control Computation on a Multimicroprocessor System", *IEEE J. Robotics and Automation,* Vol. RA-1, No. 2, pp. 104-113, 1985.

[54] R. Nigam and C. S. G. Lee, "A Multiprocessor-Based Controller for the Control of Mechanical Manipulators", *IEEE J. Robotics and Automation,* Vol. RA-1, No. 4, pp. 173-182, 1985.

[55] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation", *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. SMC-16, No. 4, pp. 532-542, 1986.

[56] M. Vucobratovic, N. Kircanski and S. G. Li, "An Approach to Parallel Processing of Dynamic Robot Models", *The Int. Journal of Robotics Research,* Vol. 7, No. 2, pp. 64-71, 1988.

[57] T. R. Kane, "Dynamics of Holonomic Systems", *ASME J. Applied Mechanics,* Vol. 28, pp. 574-578, 1961.

[58] T. R. Kane and C. F. Wang, "On the Derivation of Equations of Motion", *J. Soc. for Ind. and Appl. Math.,* Vol. 13, pp. 487-492, 1965.

[59] R. L. Huston, C. E. Passerello and M. W. Harlow, "Dynamics of Multirigid-Body Systems", *ASME J. Applied Mechanics,* Vol. 45, pp. 889-894, 1978.

[60] R. L. Huston and F. A. Kelly, "The Development of Equations of Motion of Single-Arm Robots", *IEEE Trans. Systems, Man, and Cybernetics,* Vol. SMC-12, No. 3. pp. 259-266, 1982.

[61] H. Faessler, "Computer-Assisted Generation of Dynamical Equations for Multi-Body systems", *The Int. Journal of Robotics Research,* Vol. 5, No. 3, pp. 129-141, 1986.

[62] T. R. Kane and D. A. Levinson, "The Use of Kane's Dynamical Equations in Robotics", *The Int. Journal of Robotics Research,* Vol. 2, No. 3, pp. 3-21, 1983.

[63] O. Ma, "Dynamics of Serial-type Robotic Manipulators", *Master Thesis,* Department of Mech. Eng., McGill University, Montreal, 1987.

[64] C. A. Balafoutis, R. V. Patel and J. Angeles, "A Comparative Study of Lagrange, Newton-Euler and Kane's Formulation for Robot Manipulator Dynamics" *Robotics and Manufacturing : Recent Trends in Research, Education, and Applications,* M. Jamshidi, J. Y. S. Luh, H. Seraji and G. P. Starr, Eds., ASME Press, New York, 1988.

CHAPTER VI

FORWARD DYNAMICS OF RIGID-LINK OPEN-CHAIN ROBOT MANIPULATORS

6.1 INTRODUCTION

The problem of evaluating *forward* or *direct* dynamics involves the calculation of joint accelerations (and through integration, joint velocities and positions) given the actuator torques/forces and any external torques/forces exerted on the last link of the manipulator. Forward dynamics is used primarily in simulation, so that, it is not so important for forward dynamics to meet the stringent speed requirements of inverse dynamics applications unless real-time simulation is required in which case computational efficiency is an important issue. Real-time simulation is often desirable since it provides more powerful, flexible, and economic ways of developing new robot designs and new control algorithms. Also, as fast 3-D computer graphics is becoming more easily available, robot kinematic motions have begun to be displayed on graphic work stations. Therefore, to study robot motions completely, real-time dynamics of robot manipulators need to be included in computer simulation of robotic systems.

Mathematically, the problem of forward dynamics can be described by a vector differential equation of the form

$$\ddot{\mathbf{q}}(t) = \mathbf{h}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \tau(t), \text{manipulator parameters}, \mathbf{k}(t)) \qquad (6.1)$$

where, $\mathbf{q}(t)$ is the vector of generalized coordinates (joint variables), $\dot{\mathbf{q}}(t)$ and $\ddot{\mathbf{q}}(t)$ are its derivatives with respect to time, $\tau(t)$ is the (input) generalized force vector, i.e., the vector of joint torques and/or joint forces, "manipulator parameters" are all those parameters which characterize the particular geometry and dynamics of a robot manipulator, and $\mathbf{k}(t)$ is the vector of the external torques/forces. In general, equation (6.1) is not a simple equation for which an analytic solution can be provided easily. For a gen-

eral robot manipulator equation (6.1) is very complex since, it is highly nonlinear with strong coupling between the joint variables. Hence, to solve equation (6.1) for $\mathbf{q}$, requires complicated procedures for evaluati ~ and for performing numerical integration. These procedures, as in the case of inverse dynamics, are defined by structured algorithms which are evaluated in stages.

In this chapter, we shall review the basic approaches taken to solve forward dynamics and, by introducing a new algorithm, we shall improve upon the computational efficiency of one of these methods, namely, the *composite rigid body* method which is the most efficient one, currently available, for solving forward dynamics. The outline of this chapter is as follows : Section 6.2 contains a review of existing methods for solving the forward dynamics problem (FDP). In Section 6.3, a new algorithm is devised for computing efficiently the *generalized inertia tensor* of a robot manipulator, which is a basic ingredient of the composite rigid body method. In Section 3.4 the computational complexity of this algorithm is analyzed, and the computational cost of the composite rigid body method is examined, when algorithms derived in this and the previous chapter are used to solve basic subproblems associated with this method. Finally, Section 6.5 concludes this chapter.

6.2 Previous Results on Forward Dynamics

In the past few years, two basic approaches have been taken for solving the FDP, which may be outlined as follows :

i) Obtain and solve a set of simultaneous equations in the unknown joint accelerations or,

ii) Calculate, recursively, the coefficients which propagate motion and force constraints along the mechanism allowing the problem to be solved directly.

Most of the published algorithms for solving forward dynamics adopt the first approach which involves the *composite rigid body* method. Algorithms which are derived based on

this approach can achieve $O(n^3)$ computational complexity, as oppose to $O(n)$ which can be achieved by the second approach. This is so, since in the first approach, a set of n simultaneous equations has to be solved. However, these algorithms can be computationally very efficient for small values of n, since the coefficient of n in the measure of complexity is very small.

In the framework of the first approach, a well defined scheme for deriving algorithms for computing forward dynamics has been proposed by Walker and Orin [1]. To define the algorithms, the dynamic equations of motion of a robot manipulator are written in a vector form as

$$\tau = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^T \mathbf{f} \tag{6.2.1}$$

where τ is the vector of the applied joint torques/forces, $\mathbf{D}(\mathbf{q})$ is the $n \times n$ positive definite generalized inertia tensor of the robot manipulator, $\mathbf{q}$ $(\dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is the vector of the joint positions (velocities, accelerations), $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is the vector of Coriolis and centrifugal forces, $\mathbf{G}(\mathbf{q})$ is the vector of the gravitational effects, $\mathbf{J}(\mathbf{q})$ is the $n \times n$ Jacobian tensor and $\mathbf{f}$ is a vector of external forces. Equation (6.2.1) can be written in a more compact form as

$$\tau = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q},\dot{\mathbf{q}},\mathbf{g},\mathbf{f}) \tag{6.2.2}$$

where $\mathbf{b}$ is a *bias* vector containing the gravity, centrifugal, Coriolis and external forces, i.e.,

$$\mathbf{b} = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^T \mathbf{f} \ . \tag{6.2.3}$$

Now, based on equation (6.2.2), the solution of forward dynamics can be derived by solving the following subproblems :

(i) Computation of the generalized inertia tensor : $\mathbf{D}(\mathbf{q})$.

(ii) Computation of the bias vector : $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{g}, \mathbf{f})$

(iii) Solution of the linear system of equations : $\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} = (\tau - \mathbf{b})$

(iv) Solution of a set of ordinary differential equations.

From the foregoing, for the dynamic simulation of a robot manipulator one has to solve problems directly related to manipulator dynamics (steps (i) and (ii)) and problems from numerical analysis (steps (iii) and (iv)). Therefore, since from a dynamic analysis point of view, one is concerned with the problems in steps (i) and (ii), we shall review methods for solving these two problems only. The problems of the type (iii) and (iv) have been extensively studied in the numerical analysis literature and efficient methods of solving them exist, e.g. see [18-20].

To solve part (ii), one can use non-recursive inverse dynamics algorithms, which explicitly calculate the terms of the bias vector $\mathbf{b}$ as is shown in equation (6.2.3). However, as it turns out, this approach is not computationally efficient. Walker and Orin [1], have proposed another more efficient method for computing the bias vector $\mathbf{b}$. In this method, a recursive inverse dynamics algorithm is used to solve for the actuator torques/forces, assuming that the accelerations are zero, i.e., $\ddot{\mathbf{q}} = \mathbf{0}$. As is obvious from equation (6.2.2), since the vector of the generalized forces τ is equal to the bias vector $\mathbf{b}$, an inverse dynamics algorithm suffices for solving problem (*ii*). Moreover, with significant improvements in the computational efficiency of algorithms for solving the inverse dynamics problem (see Chapter V) the solution for the bias vector $\mathbf{b}$, by this method, can be computed very efficiently. Furthermore, to improve further the computational efficiency of this method, we can formulate a specialized version of an inverse dynamics algorithm, with the assumption $\ddot{\mathbf{q}} = \mathbf{0}$ built in. Therefore, using inverse dynamics algorithms, this subproblem of forward dynamics can be solved in an efficient manner.

Solving the first problem, i.e., computing the manipulator inertia tensor $\mathbf{D}(\mathbf{q})$ is the point where most of the existing algorithms for solving the FDP, by using the first approach, really differ. Walker and Orin [1] considered three methods for computing the inertia tensor $\mathbf{D}(\mathbf{q})$. The first two methods are based on an algorithm which solves the

inverse dynamics problem and from which all the velocity terms, the gravitational effects and the effects due to the external forces and torques have been eliminated. In this approach, as we can see from the equation (6.2.2), the columns of the inertia matrix D, which represent the generalized inertia tensor $\mathbf{D}$ in joint space coordinates, are computed by applying a unit vector acceleration to the joints. That is, for the i-th column of D we have

$$d_i = (\tau - \mathbf{b})\big|_{\ddot{\mathbf{q}} = [0, \cdots, 1, \cdots, 0]^T} \tag{6.2.4}$$

where the 1 is the i-th component of $\ddot{\mathbf{q}}$. By repeating the above process n times (not necessarily recursively), all the components of D may be computed. The first two methods are basically the same, with the exception that in the second method, since D is symmetric, only the diagonal and the bottom half of the off-diagonal elements of D are computed. However, as Walker and Orin have shown in their computational complexity analysis, this approach for computing the manipulator joint space inertia matrix is computational expensive. The third method by Walker and Orin has been known as the *composite rigid body* method and is, computationally, more efficient than their previous two methods. The basic idea in the composite rigid body method is as follows :

As in the first two methods, we assume that unit acceleration is applied to a joint (for instance $\ddot{q}_i = 1$ at joint i) with all joint velocities and other joint accelerations equal to zero. Under this action the manipulator chain is divided into two sets of composite rigid bodies with one degree of freedom between them. The lower composite body, i.e., links 1 to $i-1$ is stationary and the upper composite body, which is composed of links i through n and to which we shall refer to as the i-th composite rigid body, moves as a single rigid body with a *composite mass* ($\overline{m}_i$), *composite center of mass* ($\mathbf{R}^i_{i,i}$), relative to the origin of the i-th coordinate system, and *composite moment of inertia* or *inertia tensor* ($\mathbf{E}^i_{C_i}$), with respect to the composite center of mass. A diagram for the i-th composite rigid body is shown in the following figure.

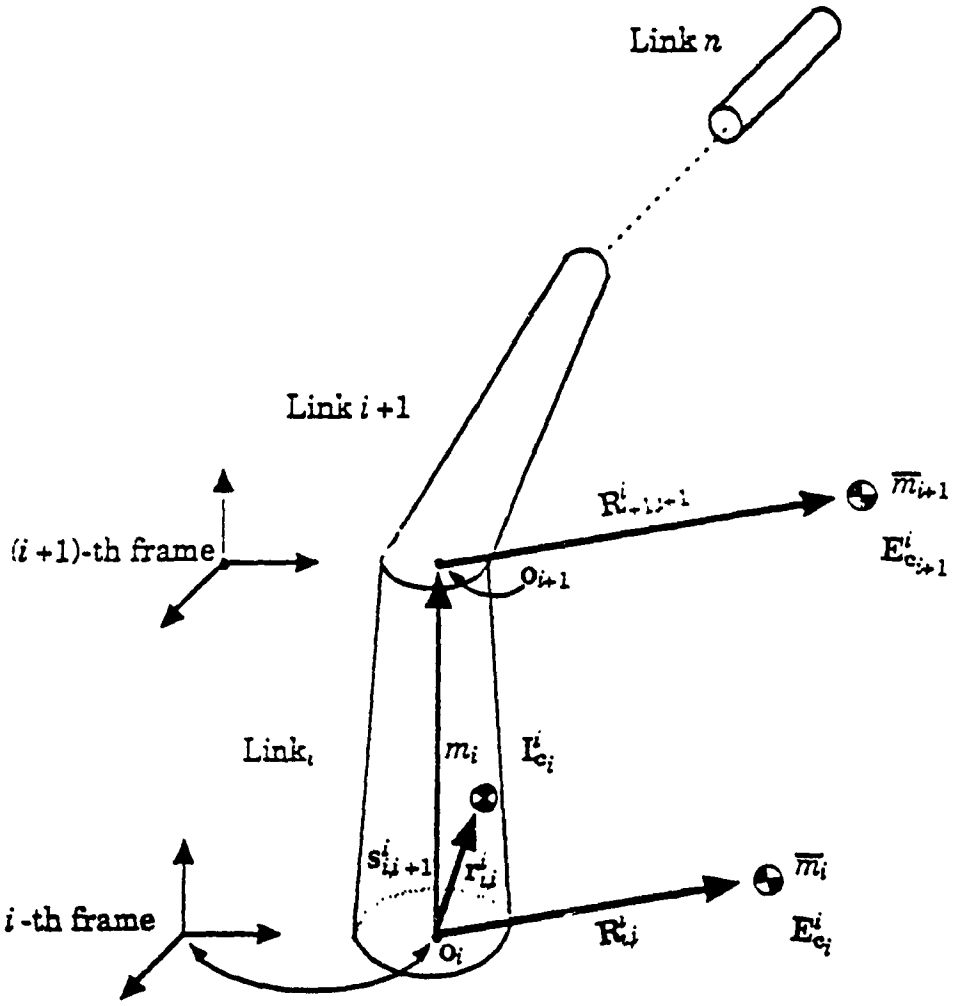

Figure 6.1 : The i-th Composite Rigid Body

Due to the motion of the i-th composite rigid body, forces and moments will be developed at the joints $1, \cdots, i$ which can be computed as follows : The force and moment at the i-th joint can be computed by applying Newton's and Euler's equations, respectively, to the i-th composite rigid body. Moreover, since the acceleration at the joints $1, \cdots, i-1$ is assumed to be zero, the forces and moments at these joints result only from the propagation down the chain of the forces and moments of the i-th joint. Now, having compute these forces or moments, to define the elements of the joint space inertia matrix D, we simply need to consider their projections onto appropriate joint axes of the manipulator.

From this computational scheme, it is obvious that the computation of $\overline{m}_i$, $\mathbf{R}^i_{i,i}$ and $\mathbf{E}^i_{C_i}$ is important (as far as computational efficiency is concerned) for the determination of the generalized inertia tensor $\mathbf{D}$. To achieve computational efficiency for these quantities, Walker and Orin proposed linear recurrence relations. These relations, formulated in the notation of this thesis, can be stated as follows :

$$\overline{m}_i = \overline{m}_{i+1} + m_i \tag{6.2.5}$$

$$\mathbf{R}^i_{i,i} = \frac{1}{\overline{m}_i}[m_i \mathbf{r}^i_{i,i} + \overline{m}_{i+1}(\mathbf{s}^i_{i,i+1} + \mathbf{A}_{i+1}\mathbf{R}^{i+1}_{i+1,i+1})] \tag{6.2.6}$$

$$\begin{aligned}\mathbf{E}^i_{C_i} = \mathbf{I}^i_{C_i} &+ m_i\left[(\mathbf{r}^i_{i,i} - \mathbf{R}^i_{i,i})\cdot(\mathbf{r}^i_{i,i} - \mathbf{R}^i_{i,i})\mathbf{1} - (\mathbf{r}^i_{i,i} - \mathbf{R}^i_{i,i})(\mathbf{r}^i_{i,i} - \mathbf{R}^i_{i,i})^T\right] \\ &+ \overline{m}_{i+1}\left[(\mathbf{s}^i_{i,i+1} + \mathbf{R}^i_{i+1,i+1} - \mathbf{R}^i_{i,i})\cdot(\mathbf{s}^i_{i,i+1} + \mathbf{R}^i_{i+1,i+1} - \mathbf{R}^i_{i,i})\mathbf{1}\right. \\ &\qquad \left. - (\mathbf{s}^i_{i,i+1} + \mathbf{R}^i_{i+1,i+1} - \mathbf{R}^i_{i,i})(\mathbf{s}^i_{i,i+1} + \mathbf{R}^i_{i+1,i+1} - \mathbf{R}^i_{i,i})^T\right] \\ &+ \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{C_{i+1}}\mathbf{A}^T_{i+1}.\end{aligned} \tag{6.2.7}$$

Based on these equations and using the Newton-Euler equations to analyze the composite rigid body dynamics, Walker and Orin proposed an algorithm for computing the upper triangular part of the symmetric joint space inertia matrix D which, in the notation of this thesis, can be stated as follows :

ALGORITHM 6.1

Step 0 : Initialization

$$\overline{m}_{n+1} = 0\,, \quad \mathbf{R}^{n}_{n+1,n+1} = 0\,, \quad \mathbf{E}^{n+1}_{C_{n+1}} = 0$$

$$\mathbf{A}_{n+1} = 0\,, \quad z_i = [\,0\;0\;1\,]^T$$

$$\sigma_i = \begin{cases} 1 & \text{revolute } i\text{-}th \text{ joint} \\ 0 & \text{prismatic } i\text{-}th \text{ joint} \end{cases}$$

Step 1: For $i = n, 1$ **do**

$$\overline{m}_i = \overline{m}_{i+1} + m_i \tag{6.2.8a}$$

$$\mathbf{R}^{i}_{i,i} = \frac{1}{\overline{m}_i}[m_i \mathbf{r}^{i}_{i,i} + \overline{m}_{i+1}(\mathbf{s}^{i}_{i,i+1} + \mathbf{A}_{i+1}\mathbf{R}^{i+1}_{i+1,i+1})] \tag{6.2.8b}$$

$$\begin{aligned} \mathbf{E}^{i}_{C_i} = \mathbf{I}^{i}_{C_i} &+ m_i \left[(\mathbf{r}^{i}_{i,i} - \mathbf{R}^{i}_{i,i})\cdot(\mathbf{r}^{i}_{i,i} - \mathbf{R}^{i}_{i,i})1 - (\mathbf{r}^{i}_{i,i} - \mathbf{R}^{i}_{i,i})(\mathbf{r}^{i}_{i,i} - \mathbf{R}^{i}_{i,i})^T\right] \\ &+ \overline{m}_{i+1}\left[(\mathbf{s}^{i}_{i,i+1} + \mathbf{R}^{i}_{i+1,i+1} - \mathbf{R}^{i}_{i,i})\cdot(\mathbf{s}^{i}_{i,i+1} + \mathbf{R}^{i}_{i+1,i+1} - \mathbf{R}^{i}_{i,i})1\right. \\ &\qquad \left. - (\mathbf{s}^{i}_{i,i+1} + \mathbf{R}^{i}_{i+1,i+1} - \mathbf{R}^{i}_{i,i})(\mathbf{s}^{i}_{i,i+1} + \mathbf{R}^{i}_{i+1,i+1} - \mathbf{R}^{i}_{i,i})^T\right] \\ &+ \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{C_{i+1}}\mathbf{A}^{T}_{i+1} \end{aligned} \tag{6.2.8c}$$

$$\mathbf{F}^{i}_{C_i} = \sigma_i(z_i \times \overline{m}_i \mathbf{R}^{i}_{i,i}) + (1 - \sigma_i)(\overline{m}_i z_i) \tag{6.2.8d}$$

$$\mathbf{M}^{i}_{C_i} = \sigma_i(\mathbf{E}^{i}_{C_i}\cdot z_i) \tag{6.2.8e}$$

$$\mathbf{f}^{i}_{i,i} = \mathbf{F}^{i}_{C_i} \tag{6.2.8f}$$

$$\eta^{i}_{i,i} = \mathbf{M}^{i}_{C_i} + \mathbf{R}^{i}_{i,i} \times \mathbf{F}^{i}_{C_i} \tag{6.2.8g}$$

$$d_{i,i} = \sigma_i(\eta^{i}_{i,i}\cdot z_i) + (1 - \sigma_i)(\mathbf{f}^{i}_{i,i}\cdot z_i) \tag{6.2.8h}$$

For $j = i-1, 1$ **do**

$$\mathbf{f}^{j}_{j,i} = \mathbf{A}_{j+1}\mathbf{f}^{j+1}_{j+1,i} \tag{6.2.9a}$$

$$\eta^{j}_{j,i} = \mathbf{A}_{j+1}\eta^{j+1}_{j+1,i} + \mathbf{s}^{j}_{j,j+1} \times \mathbf{f}^{j}_{j+1,i} \tag{6.2.9b}$$

$$d_{j,i} = \sigma_i(\eta^{j}_{j,i}\cdot z_i) + (1 - \sigma_i)(\mathbf{f}^{j}_{j,i}\cdot z_i) \tag{6.2.9c}$$

end

end

For its implementation, Algorithm 6.1 requires (see Table 6.2) $12n^2 + 26n + 27$ scalar multiplications and $7n^2 + 67n - 56$ scalar additions, which for $n = 6$ amounts to 741 scalar multiplications and 601 scalar additions, respectively. Featherstone has shown [2] that based on the composite rigid body method and using spatial notation to combine

the representation of rotational and translational quantities, and spatial algebra to manipulate efficiently these quantities, another more efficient algorithm can be devised which requires $10n^2 + 31n - 41$ scalar multiplications and $6n^2 + 40n - 46$ scalar additions. For $n = 6$, these give 505 scalar multiplications and 410 scalar additions, respectively. However, the computational efficiency of Featherstone's algorithm results, from the special purpose spatial arithmetic package which he developed to handle spatial operations efficiently.

Walker and Orin also describe another method for calculating $\ddot{\mathbf{q}}$ which by-passes the need to calculate $\mathbf{D}$ explicitly (method 4 in [1]). This method uses an iterative technique, namely, the conjugate gradient technique for solving the linear system in step (*iii*). Based on an initial estimate for the joint acceleration the method uses successive adjustments to these variables until they converge to the correct solution. If there are no round-off errors, the solution for $\ddot{\mathbf{q}}$ can be achieved in a maximum of n iterations. The complexity of this method is $o(n^2)$, but the coefficient of n^2 is large enough that it ends up being less efficient for all but very large values of n ($n \geq 12$) than the composite rigid body method, which has $o(n^3)$ computational complexity.

Following a similar decomposition of forward dynamics into subproblems, as suggested by Walker and Orin, Angeles and Ma [3] have proposed a method for computing the generalized inertia tensor $\mathbf{D}$ which in terms of computational efficiency is comparable to the composite rigid body method. The basic idea in Angeles and Ma's approach is as follows :

First the $6n$-dimensional vector of *generalized twist*

$$\mathbf{t} \equiv [\mathbf{t}_1^T, \cdots \mathbf{t}_n^T]^T \tag{6.2.10}$$

and the $6n \times 6n$ block diagonal tensor of *generalized extended mass*

$$\mathbf{M} \equiv \text{diag } (\mathbf{M}_1, \cdots \mathbf{M}_n), \tag{6.2.11}$$

are defined, where $\mathbf{t}_i$ is a 6-dimensional vector representing the twist of the i-th link,

namely,

$$\mathbf{t}_i \equiv \begin{pmatrix} \omega_i \\ \dot{\mathbf{r}}_{0,i} \end{pmatrix} \tag{6.2.12}$$

and $\mathbf{M}_i$ is a 6×6 tensor defined as

$$\mathbf{M}_i \equiv \begin{pmatrix} \mathbf{I}_{C_i} & \mathbf{0} \\ \mathbf{0} & m_i \mathbf{1} \end{pmatrix} \tag{6.2.13}$$

in which $\mathbf{1}$ is the 3×3 identity tensor and $\mathbf{0}$ is the 3×3 zero tensor. Then, from the linear transformation

$$\mathbf{t} = \mathbf{T}\,\dot{\mathbf{q}} \tag{6.2.14}$$

the $6n \times n$ tensor $\mathbf{T}$ is defined. (The tensor $\mathbf{T}$ is referred to as *the natural orthogonal complement* since, as shown in [4], it is an orthogonal complement of the tensor which defines the kinematic constraints of the manipulator). Moreover, from kinetic energy considerations the generalized inertia tensor $\mathbf{D}$ may be defined as

$$\mathbf{D}(\mathbf{q}) = \mathbf{T}^T \mathbf{M}\,\mathbf{T}\,.$$

Furthermore, the tensor $\mathbf{M}$ can be factored as $\mathbf{M} = \mathbf{N}^T\mathbf{N}$ since it is symmetric and positive definite. From the foregoing, the generalized inertia tensor $\mathbf{D}$ can be decomposed as

$$\mathbf{D} = \mathbf{P}^T\mathbf{P} \tag{6.2.15}$$

where $\mathbf{P} \equiv \mathbf{N}\,\mathbf{T}$ is a lower block triangular tensor. Based on this analysis, Angeles and Ma proposed an algorithm for computing the generalized inertia tensor $\mathbf{D}$ which, as we mentioned above, (see also Table 6.2) has almost the same comp" 'ional complexity as Algorithm 6.1. Angeles and Ma, also proposed another method which avoids the determination of the generalized inertia tensor $\mathbf{D}$. In this method, based on equation (6.2.15), the linear system

$$\mathbf{D}\,\ddot{\mathbf{q}} = \tau - \mathbf{b} \tag{6.2.16}$$

is decomposed as follows,

$$\mathbf{P}^T\mathbf{x} = \tau - \mathbf{b} \tag{6.2.17a}$$

$$\mathbf{P}\ddot{\mathbf{q}} = \mathbf{x} \tag{6.2.17b}$$

where $\mathbf{x}$ is a $6n$-dimensional vector. Equation (6.2.17a) represents an underdetermined system (n equations with $6n$ unknowns) and equation (6.2.17b) represents an overdetermined system ($6n$ equations with n unknowns). Based on these equations, Angeles and Ma have shown that the vector $\ddot{\mathbf{q}}$ can be computed as the least squares approximation to equation (6.2.17b), if $\mathbf{x}$ is computed, first, as the minimum norm solution of equation (6.2.17a). The computational efficiency of this second method, by Angeles and Ma, is comparable (see [3]) to that of their first method.

As we have mentioned above, an approach for solving the forward dynamics problem is to calculate, recursively, the coefficients which propagate motion and force constraints along the mechanism allowing the problem to be solved directly. However, although is theoretically more sound, currently few methods adopt this approach. This is because first, it requires an extensive analysis and second, and more important, algorithms derived from this approach are computationally expensive despite the fact that one can usually achieve $o(n)$ computational complexity. This is so, since the coefficient of n in the measure of complexity is quite large.

Probably, the best known method in this approach is the *articulated-body* method proposed by Featherstone [2]. The basic idea in this method is to regard the robot as consisting of a base member (whose motion is known), a single joint, and a single moving link which is in fact an articulated body (i.e., a collection of rigid bodies connected by joints) representing the rest of the robot. The forward dynamics problem for this one-joint robot is easily solved once the apparent inertia of the moving link is known. Having found the acceleration of the first joint, the articulated body itself can be treated as a robot and the same process applied to obtain the acceleration of the next joint, and so on. So the articulated-body method consists of the calculation of a series of

articulated-body inertias which are used to solve the forward dynamics problem one joint at a time. Thus, this approach leads to algorithms which have $o(n)$ computational complexity. To facilitate the analysis of his method, Featherstone introduced a spatial notation which provides a uniform combined representation of rotational and translational quantities and developed a spatial algebra for manipulating these spatial quantities. Also, to implement his algorithm efficiently (see Table 6.3), Featherstone developed a spatial arithmetic package with special-purpose arithmetic functions to operate on these compact spatial representations.

Other examples of methods which solve the FDP by the constraint propagation approach are described by Armstrong [5], Rodriguez [6] and Rodriguez and Kreutz [8]. In particular, Armstrong's method also achieves $o(n)$ complexity, and uses recursion coefficients playing a similar role to articulated-body inertias This method, in its basic form, is applicable to robots with spherical joints but a modification applicable for revolute joints is outlined in one of the appendixes in Armstrong's paper. However, this modification increases the computational requirement significantly, although the method remains $o(n)$. Rodriguez and Kreutz [8] have developed a two-step algorithm for computing forward dynamics which has $o(n)$ computational complexity. Based on a linear operator approach for formulating and analyzing the manipulator dynamics developed by Rodriguez [6,7], the two-step algorithm by Rodriguez and Kreutz first computes and subtracts out the Coriolis, centrifugal, gravity and contact force bias terms, exactly as in Walker and Orin's approach, to obtain a "bias-free" robot dynamic equation. Then, in the second step, using techniques for solving linear operator equations by operator factorization, the joint space accelerations are obtained in $o(n)$ iterations. Also, based on certain operator identities, they proposed alternative algorithms for which the need for a preliminary bias vector computation and subtraction is avoided. The dynamic analysis of these algorithms, as in Featherstone's approach, is based on spatial notation and spatial algebra. The approach by Rodriguez and Kreutz is important because it provides a

method to formulate, analyze and understand spatial recursions in multibody dynamics. This analysis leads them to a simple factorization of the generalized inertia tensor $\mathbf{D}$ from which an immediate inversion of $\mathbf{D}$ is readily available. In particular, they established the following factorization for the generalized inertia tensor $\mathbf{D}$ and its inverse :

$$\mathbf{D} = \left(1 + \mathbf{H}\,\Phi\mathbf{L}\right)\hat{\mathbf{D}}\left(1 + \mathbf{H}\,\Phi\mathbf{L}\right)^{T} \tag{6.2.18}$$

and

$$\mathbf{D}^{-1} = \left(1 - \mathbf{H}\,\Psi\mathbf{L}\right)^{T}\hat{\mathbf{D}}^{-1}\left(1 - \mathbf{H}\,\Psi\mathbf{L}\right) \tag{6.2.19}$$

where $\mathbf{H}$ and Φ are given by known geometric link parameters, and $\mathbf{L}$, Ψ and $\hat{\mathbf{D}}$ are obtained recursively by a spatial discrete-step Kalman filter and by the corresponding Riccati equation associated with this filter. The factors $\left(1 + \mathbf{H}\,\Phi\mathbf{L}\right)$ and $\left(1 - \mathbf{H}\,\Psi\mathbf{L}\right)$ are lower triangular tensors which are inverses to each other, and $\hat{\mathbf{D}}$ is a diagonal tensor. This analytic factorization and inversion is obviously important because it avoids numerical triangular decomposition and inversion and with that it avoids problems such as round-off errors or ill-conditioned problems. However, in Rodriguez and Kreutz's report a computational complexity analysis of these algorithms has not been included and this makes a fair comparison of their method with others difficult.

Finally, an approach for solving the FDP which is quite different from those presented above has been proposed by Chou, Baciu and Kesavan [9,10]. In this approach, the problem is formulated as a graph-theoretic system theory problem. This formulation uses graph-theoretic models for the joints and the open-loop kinematic chains of rigid bodies, and Euler parameters instead of the conventional direction cosines to describe relative orientations. The final mathematical model derived by this formulation is a large system ($20n$ scalar equations with $20n$ unknowns) of differential and algebraic equations. A complete computational complexity analysis has not been provided for the method. However, because of the large system of equations that have to be solved, the approach is almost certainly very expensive computationally.

Concluding this review on forward dynamics computation, it is worth mentioning that, as with inverse dynamics computation, to improve the computational efficiency parallel algorithms and special architectures have been proposed. For example, parallel processing techniques have been proposed by Lee and Chang [11] and systolic architectures have been used by Javaheri and Orin [12]. Also in [13], Han has examined possible applications of parallel and pipeline processing, as well as, VLSI systolic array processors, for solving forward dynamics in real-time.

6.3 The Generalized Manipulator Inertia Tensor

As we mentioned in section 6.2, one of the methods which may be used for the computation of the generalized inertia tensor $\mathbf{D}$ is the composite rigid body method. This method leads to algorithms (e.g., Algorithm 6.1) which compute efficiently the manipulator inertia tensor $\mathbf{D}$ by utilizing recurrence relations for some of its basic equations. However, a drawback of this method is that it leads to algorithms which require all the quantities to be computed online. Moreover, as we shall see in this section, the recurrence relations on which these algorithms are based can be stated in, computationally, more efficient formulations.

In order to reduce the computational complexity of the said algorithm two other methods are proposed in this section. The first method is similar to the third method proposed by Walker and Orin [1]. In particular, a similar decomposition, i.e., a set of stationary and moving links, is used as the underlying modeling scheme and the dynamic analysis is based on the Newton-Euler equations. However, the dynamic analysis of the moving set of links in this method uses the concepts of generalized and augmented links instead of that of the composite rigid body alone. The concepts of augmented and generalized links have been introduced in Chapter V to facilitate more efficient solutions of the inverse dynamics problem. Here, as in the case of inverse dynamics, these concepts will allow us to devise an algorithm which is applicable to gen-

eral robot manipulators and, in almost all cases, its computational burden may split into computations which can be performed *off-line* and computations which have to be performed *online*. Moreover, based on these concepts and using Cartesian tensor analysis, computationally more efficient recurrence relations will be devise to facilitate the online computations. The second method, also utilizes the concepts of augmented and generalized links and Cartesian tensor analysis, but the dynamic analysis is based now on the Euler-Lagrange equations instead of the Newton-Euler ones.

As we shall see, both methods lead to the same algorithm for computing the generalized inertia tensor **D** of a robot manipulator. Thus, from an algorithmic point of view, it may seem that this duplication in the analysis is unnecessary, and this is definitely true. However, the main reason for that duplication is to show, as we did with inverse dynamics, that apart from personal preference or experience there is nothing to be gained, in terms of computational efficiency, by choosing one or the other set of dynamic equations in our analysis.

6.3.1 Generalized Links and their Inertia Tensor

A generalized link has been defined in Chapter V (see Definition 5.2). It is obvious from this definition that a generalized link is simply a composite rigid body as defined by Walker and Orin. However, since the analysis to follow is different from that presented by Walker and Orin we shall continue to refer to the set of moving links as a generalized link. Basically, the analysis here is different from that of Walker and Orin in that all moments concerning a generalized link are considered about the origin of one of the link coordinate systems instead of its center of mass. This modification allows us to use the inertia tensor of an augmented link (which, can be computed off-line for most industrial robots) for a computationally more efficient formulation of the inertia tensor of a generalized link.

The definition for an augmented link has been given in Chapter V (see Definition

5.1). Also, in Chapter V, the definitions for the first and second moments for augmented links, as well as, the definition for the first moment of generalized links have been given. Here, for quick reference, these definitions are repeated and the list of these definitions is completed with the definition for the second moment (inertia tensor) of a generalized link. These definitions, for the i-th augmented and i-th generalized link may be stated as follows :

(1) First moment about the origin o_i, of the i-th link coordinate system, of the i-th augmented link :

$$\mathbf{u}^i_{o_i} = m_i \mathbf{r}^i_{i,i} + \overline{m}_{i+1}\mathbf{s}^i_{i,i+1} \tag{6.3.1}$$

(2) Second moment (inertia matrix) about o_i of the i-th augmented link :

$$\mathbf{K}^i_{o_i} = \mathbf{I}^i_{C_i} - m_i \tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i} - \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1} \tag{6.3.2}$$

(3) First moment about o_i of the i-th generalized link :

$$\begin{aligned}\mathbf{U}^i_{o_i} &= \sum_{j=i}^{n} m_j \mathbf{r}^i_{i,j} \\ &= \mathbf{u}^i_{o_i} + A_{i+1}\mathbf{U}^{i+1}_{o_{i+1}}.\end{aligned} \tag{6.3.3}$$

where the last step follows from Lemma 5.1, and

(4) Second moment (inertia matrix) about o_i of the i-th generalized link :

$$\mathbf{E}^i_{o_i} = \sum_{k=i}^{n} [\mathbf{I}^i_{C_k} - m_k \tilde{\mathbf{r}}^i_{i,k}\tilde{\mathbf{r}}^i_{i,k}]. \tag{6.3.4}$$

Also, as we may recall from Chapter V, the zero-th moment or mass of the i-th augmented link which is equal to the zero-th moment of the i-th generalized link is defined to be

$$\overline{m}_i = m_i + \overline{m}_{i+1} \tag{6.3.5}$$

The formulation of equation (6.3.4), which defines the inertia tensor of the i-th generalized link, is important because it provides physical insight into the structure of this inertia tensor. However, this formulation is obviously computational very expensive

for any use in practical applications. Therefore, to be able to use generalized links effectively, we have to define their inertia tensor by using a computationally more efficient equation. This equation is provided by the following lemma [17].

Lemma 6.1: The inertia tensor of the i-th generalized link with respect to the origin o_i of the i-th link coordinate system may be defined by the following recurrence equation :

$$\mathbf{E}^i_{O_i} = \mathbf{K}^i_{O_i} - \left[\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} + \tilde{\mathbf{U}}^i_{O_{i+1}}\tilde{\mathbf{s}}^i_{i,i+1}\right] + \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{O_{i+1}}\mathbf{A}^T_{i+1} \tag{6.3.6}$$

where $\mathbf{K}^i_{O_i}$ is the inertia tensor of the i-th augmented link with respect to the origin o_i, $\tilde{\mathbf{s}}^i_{i,i+1}$ is the dual tensor of the position vector of o_{i+1} relative to o_i and $\tilde{\mathbf{U}}^i_{O_{i+1}}$ is the dual tensor of the first moment about o_{i+1} of the $(i+1)$-th generalized link.

Proof : Since, for $k > i$, $\tilde{\mathbf{r}}^i_{i,k} = \tilde{\mathbf{s}}^i_{i,i+1} + \tilde{\mathbf{r}}^i_{i+1,k}$ we have

$$\tilde{\mathbf{r}}^i_{i,k}\tilde{\mathbf{r}}^i_{i,k} = \tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1} + \tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{r}}^i_{i+1,k} + \tilde{\mathbf{r}}^i_{i+1,k}\tilde{\mathbf{s}}^i_{i,i+1} + \tilde{\mathbf{r}}^i_{i+1,k}\tilde{\mathbf{r}}^i_{i+1,k},$$

equation (6.3.4) can be written as

$$\begin{aligned}
\mathbf{E}^i_{O_i} &= \mathbf{I}^i_{C_i} - m_i\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i} - \sum_{k=i+1}^{n} m_k\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1} \\
&\quad - \sum_{k=i+1}^{n}\left[m_k(\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{r}}^i_{i+1,k} + \tilde{\mathbf{r}}^i_{i+1,k}\tilde{\mathbf{s}}^i_{i,i+1})\right] + \sum_{k=i+1}^{n}\left[\mathbf{I}^i_{C_k} - m_k\tilde{\mathbf{r}}^i_{i+1,k}\tilde{\mathbf{r}}^i_{i+1,k}\right] \\
&= \mathbf{I}^i_{C_i} - m_i\tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i} - \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}\tilde{n}^i_{i,i+1} - \left[\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} + \tilde{\mathbf{U}}^i_{O_{i+1}}\tilde{\mathbf{s}}^i_{i,i+1}\right] \\
&\quad + \mathbf{A}_{i+1}\sum_{k=i+1}^{n}\left[\mathbf{I}^{i+1}_{C_k} - m_k\tilde{\mathbf{r}}^{i+1}_{i+1,k}\tilde{\mathbf{r}}^{i+1}_{i+1,k}\right]\mathbf{A}^T_{i+1} \\
&= \mathbf{K}^i_{O_i} - \left[\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} + \tilde{\mathbf{U}}^i_{O_{i+1}}\tilde{\mathbf{s}}^i_{i,i+1}\right] + \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{O_{i+1}}\mathbf{A}^T_{i+1}
\end{aligned}$$

where the definitions of the inertia tensors for the i-th augmented and i-th generalized links have been used in the last step. Thus, equation (6.3.6) is valid and this completes the proof. □

As we mentioned above, a generalized link is another name for the concept of the composite rigid body. Therefore, it is obvious that equations (6.2.6) and (6.2.7) which define the composite center of mass and the composite inertia tensor, respectively, of the

i-th composite rigid body are closely related to equations (6.3.3) and (6.3.6) which define the first and second moments, respectively, of the i-th generalized link. To see this, by substituting $\mathbf{u}^i_{O_i}$ from equation (6.3.1) into equation (6.3.3), and using equation (6.2.6) we can write the first moment of the i-th generalized link as follows :

$$\mathbf{U}^i_{O_i} = \overline{m}_i \mathbf{R}^i_{i,i}. \tag{6.3.7}$$

where $\mathbf{R}^i_{i,i}$ is the position vector of the center of mass of the i-th composite rigid body (or the i-th generalized link) with respect to the origin o_i of the i-th link coordinate system. Note that, based on physical considerations, we could use equation (6.3.7) instead of equation (6.3.3) as the definition of the first moment of the i-th generalized link with respect to the origin o_i of the i-th link coordinate system. However, the chosen definition provides physical insight which facilitates the analysis of the dynamic equations of motion. Similarly, using the parallel axis theorem, the inertia tensor of the i-th generalized link $\mathbf{E}^i_{O_i}$ with respect to the origin o_i can be defined by the equation

$$\mathbf{E}^i_{O_i} = \mathbf{E}^i_{C_i} - \overline{m}_i \tilde{\mathbf{R}}^i_{i,i} \tilde{\mathbf{R}}^i_{i,i} \tag{6.3.8}$$

where $\mathbf{E}^i_{C_i}$ (see equation (6.2.7)) is the inertia tensor of the i-th generalized link with respect to its center of mass, and $\tilde{\mathbf{R}}^i_{i,i}$ is the dual tensor of the position vector $\mathbf{R}^i_{i,i}$.

Now, using these moments of the augmented and generalized links, we can proceed to derive an algorithm for computing efficiently the joint-space matrix D of the generalized inertia tensor $\mathbf{D}$ of a rigid-link open-chain robot manipulator.

6.3.2 The Use of Newton-Euler Equations in Computing the Manipulator Inertia Tensor

To simplify the analysis we shall assume that all joints are of revolute type. However, the final equations in Algorithm 6.2 have been modified to make the algorithm applicable to both revolute and prismatic joints. These modifications are simple and self explanatory.

Following the approach by Walker and Orin, let us assume that unit acceleration is applied at the i-th joint of a robot manipulator with all joint velocities and other joint accelerations equal to zero. Under these assumptions only the i-th generalized link moves. To describe the motion of the i-th generalized link, we shall use Newton's and Euler's equations.

As is well known, Newton's and Euler's equations describe the motion of a rigid body relative to an inertia frame and are given by the equations [14]

$$\mathbf{F}_C = m\,\ddot{\mathbf{r}}_C \tag{6.3.9}$$

$$\mathbf{M}_C = \mathbf{I}_C \cdot \dot{\omega} + \tilde{\omega}\mathbf{I}_C \cdot \omega \tag{6.3.10}$$

respectively, where m is the mass, $\ddot{\mathbf{r}}_C$ is the absolute acceleration of the center of mass, $\mathbf{I}_C$ is the inertia tensor of the rigid body about its center of mass and ω ($\dot{\omega}$) is the absolute angular velocity (acceleration) of the motion.

In the case of the i-th composite link , equations (6.3.9) and (6.3.10) take the form

$$\mathbf{F}_{C_i} = \overline{m}_i \ddot{\mathbf{R}}_{0,i} \tag{6.3.11}$$

and

$$\mathbf{M}_{C_i} = \mathbf{E}_{C_i} \cdot \dot{\omega}_i + \tilde{\omega}_i \mathbf{E}_{C_i} \cdot \omega_i \,. \tag{6.3.12}$$

Note that, (6.3.11) and (6.3.12) are expressed in the base frame orientation. However, since by assumption the links 1 to $i-1$ are stationary, we can assume that the origin of the inertia frame is at the same position as the origin o_i of the i-th link coordinate frame. This implies that $\ddot{\mathbf{R}}_{0,i} = \ddot{\mathbf{R}}_{i,i}$. Therefore, using equations (6.3.7) and (6.3.8) we can rewrite equations (6.3.11) and (6.3.12) in the following form

$$\mathbf{F}_{C_i} = \ddot{\mathbf{U}}_{O_i} \tag{6.3.13}$$

and

$$\mathbf{M}_{C_i} = \mathbf{E}_{O_i} \cdot \dot{\omega}_i + \tilde{\omega}_i \mathbf{E}_{O_i} \cdot \omega_i + \frac{1}{\overline{m}_i}\left[\tilde{\mathbf{U}}_{O_i}\tilde{\mathbf{U}}_{O_i} \cdot \dot{\omega}_i + \tilde{\omega}_i \tilde{\mathbf{U}}_{O_i}\tilde{\mathbf{U}}_{O_i} \cdot \omega_i\right] \tag{6.3.14}$$

Moreover, under the assumptions made above, we have

$$\omega_i = 0 \quad \text{and} \quad \dot{\omega}_i = \mathbf{z}_i . \tag{6.3.15}$$

which implies that $\Omega_j = \tilde{\mathbf{z}}_i$ for all $j \geq i$. Therefore, using Lemma 5.2, we can show that

$$\ddot{\mathbf{U}}_{O_i} = \tilde{\mathbf{z}}_i \mathbf{U}_{O_i} . \tag{6.3.16}$$

From the foregoing, equations (6.3.13) and (6.3.14) can be written as

$$\mathbf{F}_{C_i} = \tilde{\mathbf{z}}_i \mathbf{U}_{O_i} \tag{6.3.17}$$

and

$$\mathbf{M}_{C_i} = \mathbf{E}_{O_i} \cdot \mathbf{z}_i + \frac{1}{\overline{m}_i} \tilde{\mathbf{U}}_{O_i} \tilde{\mathbf{U}}_{O_i} \mathbf{z}_i \tag{6.3.18}$$

respectively. Moreover, equations (6.3.17) and (6.3.18) as tensor equations are invariant under coordinate transformations. Therefore, following the usual approach in the Newton-Euler formulation of robot dynamics, we express them in the i-th frame orientation as

$$\mathbf{F}^i_{C_i} = \tilde{\mathbf{z}}^i_i \mathbf{U}^i_{O_i} \tag{6.3.19}$$

and

$$\mathbf{M}^i_{C_i} = \mathbf{E}^i_{O_i} \cdot \mathbf{z}^i_i + \frac{1}{\overline{m}_i} \tilde{\mathbf{U}}^i_{O_i} \tilde{\mathbf{U}}^i_{O_i} \mathbf{z}^i_i \tag{6.3.20}$$

respectively.

Now, due to the motion of the i-th joint, forces ($\mathbf{f}^j_{j,i}$) and moments ($\eta^j_{j,i}$) will be developed at all joints j for $j \leq i$. These forces and moments are the inter-link constraints which keep the links in the lower part of the manipulator fixed with respect to each other. Projections of these forces or moments onto the joint axes are just the elements of the desired joint-space inertia matrix D of the generalized inertia tensor $\mathbf{D}$. To determine these constraint forces and moments we use a backward recursion from joint i through joint 1. At the initial step of this recursion, i.e., for $j = i$, $\mathbf{f}^i_{i,i}$ and $\eta^i_{i,i}$ can be determined by simply resolving $\mathbf{F}^i_{C_i}$ and $\mathbf{M}^i_{C_i}$ to the origin of the i-th coordi-

nates. Thus we have

$$\mathbf{f}^i_{i,i} = \mathbf{F}^i_{C_i} \tag{6.3.21}$$

and

$$\boldsymbol{\eta}^i_{i,i} = \mathbf{M}^i_{C_i} + \tilde{\mathbf{R}}^i_{i,i}\mathbf{F}^i_{C_i} \tag{6.3.22}$$

Further, by using (6.3.19) and (6.3.20), equation (6.3.22) can be written as

$$\boldsymbol{\eta}^i_{i,i} = \mathbf{E}^i_{O_i} \cdot \mathbf{z}^i_i + \frac{1}{\overline{m}_i}\tilde{\mathbf{U}}^i_{O_i}\tilde{\mathbf{U}}^i_{O_i}\mathbf{z}^i_i + \tilde{\mathbf{R}}^i_{i,i}\tilde{\mathbf{z}}^i_i\mathbf{U}^i_{O_i} \tag{6.3.23}$$

and finally, since $\tilde{\mathbf{z}}^i_i\mathbf{U}^i_{O_i} = -\tilde{\mathbf{U}}^i_{O_i}\mathbf{z}^i_i$ and $\tilde{\mathbf{R}}^i_{i,i} = \frac{1}{\overline{m}_i}\tilde{\mathbf{U}}^i_{O_i}$, we have

$$\boldsymbol{\eta}^i_{i,i} = \mathbf{E}^i_{O_i} \cdot \mathbf{z}^i_i. \tag{6.3.24}$$

In the rest of the recursion, i.e., for $j < i$, the constraint forces and moments are computed by using the equations

$$\begin{aligned}\mathbf{f}^j_{j,i} &= \mathbf{A}_{j+1}\mathbf{f}^{j+1}_{j+1,i} \\ &= \mathbf{f}^j_{j+1,i}\end{aligned} \tag{6.3.25}$$

and

$$\begin{aligned}\boldsymbol{\eta}^j_{j,i} &= \mathbf{A}_{j+1}\boldsymbol{\eta}^{j+1}_{j+1,i} + \tilde{\mathbf{s}}^j_{j,j+1}\mathbf{A}_{j+1}\mathbf{f}^{j+1}_{j+1,i} \\ &= \mathbf{A}_{j+1}\left[\boldsymbol{\eta}^{j+1}_{j+1,i} + \tilde{\mathbf{s}}^{j+1}_{j,j+1}\mathbf{f}^{j+1}_{j+1,i}\right].\end{aligned} \tag{6.3.26}$$

Note that these equations can also be derived from a compatible inverse dynamic algorithm, say Algorithm 5.4, when the accelerations at the j-th joints, for $j \neq i$, are assumed to be zero. Finally, the elements of the symmetric joint-space inertia matrix D of the generalized inertia tensor $\mathbf{D}$ are determined by using the following equation

$$d_{j,i} = \boldsymbol{\eta}^j_{j,i} \cdot \mathbf{z}^j_j. \tag{6.3.27}$$

Based on the above equations, we can state the following algorithm for computing the elements of the joint-space inertia matrix D.

ALGORITHM 6.2

Step 0 : Initialization :

$$\overline{m}_{n+1} = 0\,, \quad \mathbf{A}_{n+1} = 0\,, \quad \mathbf{U}_{O_{n+1}} = 0\,, \quad z_i = [\,0\;0\;1\,]^T$$

$$\sigma_i = \begin{cases} 1 & \text{revolute } i\text{-}th \text{ joint} \\ 0 & \text{prismatic } i\text{-}th \text{ joint} \end{cases}$$

Step 1: For $i = n, \ldots, 1$, **do**

$$\overline{m}_i = m_i + \overline{m}_{i+1} \tag{6.3.28a}$$

$$\mathbf{u}^i_{O_i} = m_i \mathbf{r}^i_{i,i} + \overline{m}_{i+1}\mathbf{s}^i_{i,i+1} \tag{6.3.28b}$$

$$\mathbf{K}^i_{O_i} = \mathbf{I}^i_{C_i} - m_i \tilde{\mathbf{r}}^i_{i,i}\tilde{\mathbf{r}}^i_{i,i} - \overline{m}_{i+1}\tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{s}}^i_{i,i+1} \tag{6.3.28c}$$

end

Step 2: For $i = n, \ldots, 1$, **do**

$$\mathbf{U}^i_{O_i} = \mathbf{u}^i_{O_i} + \mathbf{A}_{i+1}\mathbf{U}^{i+1}_{O_{i+1}} \tag{6.3.29a}$$

$$\mathbf{s}^{i+1}_{i,i+1} = \mathbf{A}^T_{i+1}\mathbf{s}^i_{i,i+1} \tag{6.3.29b}$$

$$\mathbf{E}^i_{O_i} = \mathbf{K}^i_{O_i} + \mathbf{A}_{i+1}\left\{\mathbf{E}^{i+1}_{O_{i+1}} - \left[\tilde{\mathbf{s}}^{i+1}_{i,i+1}\tilde{\mathbf{U}}^{i+1}_{O_{i+1}} + (\tilde{\mathbf{s}}^{i+1}_{i,i+1}\tilde{\mathbf{U}}^{i+1}_{O_{i+1}})^T\right]\right\}\mathbf{A}^T_{i+1} \tag{6.3.29c}$$

$$\mathbf{f}^i_{i,i} = \sigma_i \tilde{\mathbf{z}}^i_i \mathbf{U}^i_{O_i} + (1-\sigma_i)\overline{m}_i\,\mathbf{z}^i_i \tag{6.3.29d}$$

$$\eta^i_{i,i} = \sigma_i \mathbf{E}^i_{O_i}\cdot\mathbf{z}^i_i \tag{6.3.29e}$$

$$d_{i,i} = \sigma_i(\eta^i_{i,i}\cdot\mathbf{z}^i_i) + (1-\sigma_i)(\mathbf{f}^i_{i,i}\cdot\mathbf{z}^i_i) \tag{6.3.29f}$$

Step 3: For $j = i\text{-}1, \ldots, 1$, **do**

$$\mathbf{f}^j_{j,i} = \mathbf{A}_{j+1}\mathbf{f}^{j+1}_{j+1,i} \tag{6.3.30a}$$

$$\eta^j_{j,i} = \mathbf{A}_{j+1}\left[\tilde{\mathbf{s}}^{j+1}_{j,j+1}\mathbf{f}^{j+1}_{j+1,i} + \eta^{j+1}_{j+1,i}\right] \tag{6.3.30b}$$

$$d_{j,i} = \sigma_j(\eta^j_{j,i}\cdot\mathbf{z}^j_j) + (1-\sigma_j)(\mathbf{f}^j_{j,i}\cdot\mathbf{z}^j_j) \tag{6.3.30c}$$

end

end

Note that since the joint-space inertia matrix D is symmetric, Algorithm 6.2 computes only the upper triangular part of it. Also, as we mentioned above, certain equations of this algorithm have been modified to be applicable for both revolute and prismatic joints.

6.3.3 The Use of Euler-Lagrange Equations in Computing the Manipulator Inertia Tensor

In this section we demonstrate how the Euler-Lagrange equations can be used to derive Algorithm 6.2. To simplify the derivation we consider revolute joints only. However, with slight modifications the analysis can be extended to include prismatic joints as well.

In the Euler-Lagrange approach, we first compute the Lagrangian of the manipulator, which is defined by

$$\mathbf{L} = \Psi - \Phi \tag{6.3.31}$$

where Ψ is the kinetic energy and Φ is the potential energy of the manipulator. Then to derive the generalized torques (i.e., forces and torques acting at the joints), we use the Euler-Lagrange equations

$$\tau_i = \frac{d}{dt}\frac{\partial \mathbf{L}}{\partial \dot{q}_i} - \frac{\partial \mathbf{L}}{\partial q_i} \qquad i = 1,2, \ldots, n \tag{6.3.32}$$

where, q_i ($\dot{q}_i$) are the generalized coordinates (velocities) of the manipulator. Performing the differentiation, involved in (6.3.32), we can write the equation for the generalized torques (when there are no external forces acting on the manipulator) in vector form as

$$\tau = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{6.3.33}$$

where, $\mathbf{D}(\mathbf{q})$ is the generalized inertia tensor of the manipulator, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is a vector which contains Coriolis and centrifugal forces and $\mathbf{G}(\mathbf{q})$ is the vector of gravitational forces.

Since the potential energy is independent of the joint velocities, it is obvious that the generalized inertia tensor $\mathbf{D}(\mathbf{q})$ results from the kinetic energy only. Actually, if we write the kinetic energy of the manipulator in the form

$$\Psi = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{H}(\mathbf{q})\dot{\mathbf{q}} \tag{6.3.34}$$

where $\mathbf{H}(\mathbf{q})$ is the kinetic energy tensor, we can compute the generalized inertia tensor $\mathbf{D}(\mathbf{q})$ directly by setting

$$\mathbf{D}(\mathbf{q}) = \mathbf{H}(\mathbf{q}) \tag{6.3.35}$$

i.e., the generalized inertia tensor of a manipulator is simply the kinetic energy tensor. Therefore, to compute the tensor $\mathbf{D}(\mathbf{q})$, we have to derive the kinetic energy of the manipulator in the form given by equation (6.3.34).

Kinetic energy is one of the most important physical quantities in rigid body dynamics and is defined by a number of equivalent equations [14]. Here, to define the kinetic energy of the k-th link of a robot manipulator, we use the following equation [14]

$$\Psi_k = \frac{1}{2} m_k \dot{\mathbf{r}}_{0,k} \cdot \dot{\mathbf{r}}_{0,k} + \frac{1}{2} \omega_k \cdot \mathbf{I}_{C_k} \cdot \omega_k \tag{6.3.36}$$

where the first term defines the translational kinetic energy and the second term defines the rotational kinetic energy of the k-th link. Now, using the superposition theorem, we get the total kinetic energy of a robot manipulator as

$$\Psi = \frac{1}{2} \sum_{k=1}^{n} [m_k \dot{\mathbf{r}}_{0,k} \cdot \dot{\mathbf{r}}_{0,k} + \omega_k \cdot \mathbf{I}_{C_k} \cdot \omega_k]. \tag{6.3.37}$$

The absolute linear and angular velocities of the k-th link can be defined explicitly by the following equations

$$\dot{\mathbf{r}}_{0,k} = \sum_{i=1}^{k} (\mathbf{z}_i \times \mathbf{r}_{i,k}) \dot{q}_i \tag{6.3.38}$$

and

$$\omega_k = \sum_{i=1}^{k} \mathbf{z}_i \dot{q}_i . \tag{6.3.39}$$

Now, by substituting (6.3.38) and (6.3.39) in (6.3.37) and noticing that from equation (3.4.9) we have

$$(\mathbf{z}_j \times \mathbf{r}_{j,k}) \cdot (\mathbf{z}_i \times \mathbf{r}_{i,k}) = - \mathbf{z}_j \cdot \tilde{\mathbf{r}}_{j,k} \tilde{\mathbf{r}}_{i,k} \cdot \mathbf{z}_i$$

we can write (6.3.39) as

$$\Psi = \frac{1}{2}\sum_{k=1}^{n}\sum_{i,j=1}^{k}\left[\mathbf{z}_j\cdot(\mathbf{I}_{C_k} - m_k\tilde{\mathbf{r}}_{j,k}\tilde{\mathbf{r}}_{i,k})\cdot\mathbf{z}_i\right]\dot{q}_i\,\dot{q}_j\,. \tag{6.3.40}$$

Finally, the permutation of the two summation symbols in (6.3.40) gives

$$\Psi = \frac{1}{2}\sum_{i,j=1}^{n}\sum_{k=max(i,j)}^{n}\left[\mathbf{z}_j\cdot(\mathbf{I}_{C_k} - m_k\tilde{\mathbf{r}}_{j,k}\tilde{\mathbf{r}}_{i,k})\cdot\mathbf{z}_i\right]\dot{q}_i\,\dot{q}_j\,. \tag{6.3.41}$$

Therefore, from (6.3.34), (6.3.35) and (6.3.41), we have that the elements of the joint-space inertia matrix D of the generalized inertia tensor D (q) satisfy the following equation,

$$d_{j,i} = \mathbf{z}_j\cdot\left[\sum_{k=max(i,j)}^{n}(\mathbf{I}_{C_k} - m_k\tilde{\mathbf{r}}_{j,k}\tilde{\mathbf{r}}_{i,k})\right]\cdot\mathbf{z}_i \tag{6.3.42}$$

Equation (6.3.42) can be simplified if one uses equations (6.3.3) and (6.3.4) and Lemma 6.1. To see this, let us assume that $j \leq i$, then since by definition $\mathbf{r}_{j,k} = \mathbf{s}_{j,i} + \mathbf{r}_{i,k}$, we have

$$\sum_{k=i}^{n}[\mathbf{I}_{C_k} - m_k\tilde{\mathbf{r}}_{j,k}\tilde{\mathbf{r}}_{i,k}] = \sum_{k=i}^{n}[\mathbf{I}_{C_k} - m_k\tilde{\mathbf{r}}_{i,k}\tilde{\mathbf{r}}_{i,k}] - \sum_{k=i}^{n}m_k\tilde{\mathbf{s}}_{j,i}\tilde{\mathbf{r}}_{i,k}$$

$$= \mathbf{E}_{O_i} - \tilde{\mathbf{s}}_{j,i}\tilde{\mathbf{U}}_{O_i} \tag{6.3.43}$$

Therefore, we can write equation (6.3.42) as

$$d_{j,i} = \mathbf{z}_j\cdot\left[\mathbf{E}_{O_i} - \tilde{\mathbf{s}}_{j,i}\tilde{\mathbf{U}}_{O_i}\right]\cdot\mathbf{z}_i \tag{6.3.44}$$

Moreover, equation (6.3.44) as a tensor equation is invariant under coordinate transformations. Therefore, it can be written in the j-th frame orientation as

$$d_{j,i} = \mathbf{z}_j^j\cdot\left[\mathbf{E}_{O_i}^j - \tilde{\mathbf{s}}_{j,i}^j\tilde{\mathbf{U}}_{O_i}^j\right]\cdot\mathbf{z}_i^j \tag{6.3.45}$$

Now, we shall show that (6.3.45) is equivalent to (6.3.27). First we note that the vector $\boldsymbol{\eta}_{j,i}^j$, defined by (6.3.26), can also be written as

$$\eta^j_{j,i} = \left[\mathbf{E}^j_{O_i} - \tilde{\mathbf{s}}^j_{j,i}\bar{\mathbf{U}}^j_{O_i}\right]\cdot \mathbf{z}_i^j \tag{6.3.46}$$

To see this, we write equation (6.3.26) in its expand form as

$$\begin{aligned}\eta^j_{j,i} &= A_{j+1}\eta^{j+1}_{j+1,i} + \tilde{\mathbf{s}}^j_{j,j+1}\mathbf{f}^j_{j,i}\\ &= \tilde{\mathbf{s}}^j_{j,j+1}\mathbf{f}^j_{j,i} + \tilde{\mathbf{s}}^j_{j+1,j+2}\mathbf{f}^j_{j,i} + \cdots + \tilde{\mathbf{s}}^j_{i-1,i}\mathbf{f}^j_{j,i} + \mathbf{E}^j_{O_i}\mathbf{z}_i^j\\ &= \tilde{\mathbf{s}}^j_{j,i}\mathbf{f}^j_{j,i} + \mathbf{E}^j_{O_i}\mathbf{z}_i^j\end{aligned}$$

and since, $\mathbf{f}^j_{j,i} = \bar{\mathbf{z}}_i^j\,\mathbf{U}^j_{O_i} = -\bar{\mathbf{U}}^j_{O_i}\,\mathbf{z}_i^j$, we get equation (6.3.46).

From the foregoing, equations (6.3.26) and (6.3.46) are equivalent. Moreover, since $\mathbf{s}^i_{i,i} = 0$ for all i, it is obvious that equation (6.3.46) contains equation (6.3.24). Thus the joint-space inertia matrix of a robot manipulator can be computed using either (6.3.27) or (6.3.45). Therefore, Newton-Euler and Euler-Lagrange formulations both lead to the same equations for defining the generalized inertia tensor of a robot manipulator.

6.4 Implementation and Computational Considerations

In this section, we shall demonstrate how Algorithm 6.2, can be implemented numerically in an efficient manner. Similar observations as those made in the numerical implementation of Algorithms 5.4 and 5 5 in Chapter V, can be made here. Thus, for example, when we talk about the computation of the matrix $\mathbf{E}^i_{C_i}$ we actually mean the computation of the coordinate matrix $E^i_{C_i}$ of the tensor $\mathbf{E}^i_{C_i}$. Moreover, since most robot manipulators have only one prismatic joint we shall assume that the moments of an augmented link, i.e., Step 1 of Algorithm 6.2, can be computed off-line. Therefore, in the following, we shall be concerned with the numerical implementation of the second and third steps of Algorithm 6.2, and we shall consider two cases : robot manipulators with a general geometric structure and robot manipulators for which the twist angle is, by design, either 0 or 90 degrees.

From the structure of the equations in these two steps, it is clear that the maximum number of operations required for implementing Algorithm 6.2 results from various matrix-vector and matrix-matrix multiplications. As we mentioned in Section 5.3.2, for a matrix-vector multiplication where the matrix under consideration is a coordinate transformation matrix, we need 9 scalar multiplications and 4 scalar additions. When the twist angle α, in the transformation matrix, is either 0 or 90 degrees, we need only 4 scalar multiplications and 2 scalar additions. For a matrix-vector multiplication where the matrix under consideration is a skew-symmetric matrix, we need 6 scalar multiplications and 3 scalar additions. Finally, the implementation of the product of two skew-symmetric matrices requires 9 scalar multiplications and 3 scalar additions.

Now, as we can see from Algorithm 6.2, computationally, the most intensive equation is equation (6.3.20c) which computes the inertia tensor of a generalized link. For a computationally efficient implementation of this equation we notice the following : The symmetric matrix $\left[(\tilde{s}_{i,i+1}^{i+1}\bar{U}_{C_{i+1}}^{i+1} + (\tilde{s}_{i,i+1}^{i+1}\bar{U}_{O_{i+1}}^{i+1})^T\right]$ can be implemented with only 9 scalar multiplications and 9 scalar additions. Moreover, it can be shown that by using the following trigonometric identities :

$$a) \quad \sin(2\theta) = 2\sin(\theta)\cos(\theta)$$

$$b) \quad \cos^2(\theta) = \frac{1 + \cos(2\theta)}{2}$$

$$c) \quad \sin^2(\theta) = \frac{1 - \cos(2\theta)}{2}$$

the transformation of a symmetric matrix from one coordinate system to another can be implemented very efficiently. Thus, it can be shown that when the twist angle of the transformation matrix $\mathbf{A}_i$ is different from 0 or 90 degrees, for the transformation of a symmetric matrix we require 21 scalar multiplications and 18 scalar additions. When the twist angle of $\mathbf{A}_i$ is equal to 0 or 90 degrees, we need 11 scalar multiplications and 11 scalar additions. Thus, to implement equation (6.3.20c), in the general case, we need 30 scalar multiplications and 39 scalar additions.

Based on these general remarks, we shall now analyze the implementation of Algorithm 6.2, when this algorithm is applied to a robot with all revolute joints. We first notice that since $z_i^i = [\,0\;0\;1\,]^T$, equations (6.3.29d)-(6.3.29f) and (6.3.30c) do not require any computations for their implementation. To implement equation (6.3.29b) we can avoid the matrix-vector multiplication, since, as we can see from equations (2.3.3) and (2.3.4), the vector $\mathbf{s}_{i,i+1}^{i+1}$ can be defined as

$$\mathbf{s}_{i,i+1}^{i+1} = \begin{bmatrix} a_i \cos(q_{i+1}) \\ -a_i \sin(q_{i+1}) \\ d_i \end{bmatrix}$$

where a_i and d_i are known link parameters. Thus, we can implement equation (6.3.29b) with only 2 scalar multiplications. Moreover, for $i = n$, since $\mathbf{A}_{n+1} = 0$, we have $\mathbf{U}_{O_n}^n = \mathbf{u}_{O_n}^n$ and $\mathbf{E}_{O_n}^n = \mathbf{K}_{O_n}^n$.

From the foregoing, no computations are involved in Step 2, when $i = n$. Also, since $d_{1,1}$ is the (3×3) entry of the matrix $\mathbf{E}_{O_1}^1$, when $i = 1$ we need to compute only the (3×3) entry of $\mathbf{E}_{O_1}^1$ and not the complete matrix. This also implies that $\mathbf{U}_{O_1}^1$ need not be computed. In Step 3, equations (6.3.30a) and (6.3.30b) each need to be evaluated $n(n-1)/2$ times, since there are $n(n-1)/2$ off-diagonal elements in the upper triangular part of the joint space inertia matrix D. However, when $j = 1$, there is no need to compute the vector $\mathbf{f}_{1,i}^1$ (since it is not used any-where) and from the vector $\eta_{1,i}^1$ we need only to compute its last entry. Thus, some saving can be made in computing the $n-1$ elements $d_{1,i}$ of D in Step 3 of the algorithm if these considerations are taken into account.

Following the observations made above, a breakdown of the number of scalar multiplications and additions required for the online implementation by each equation of Algorithm 6.2 is given in Table 6.1. The total figure represents the operations count for computing the inertia matrix (upper triangular part) of a robot manipulator with all joints of revolute type, and is valid for $n \geq 2$.

Step 2 & 3	General manipulator		Manipulator with $\alpha=0^o$ or 90^o	
Equation	Multiplications	Additions	Multiplications	Additions
6.3.29a	$8n-16$	$7n-10$	$4n-8$	$5n-10$
6.3.29b	$2n-2$	0	$2n-2$	0
6.3.29c	$30n-42$	$39n-52$	$20n-25$	$32n-42$
6.3.29d	0	0	0	0
6.3.29e	0	0	0	0
6.3.29f	0	0	0	0
6.3.30a	$4n^2-12n+8$	$2n^2-6n+4$	$2n^2-6n+4$	n^2-3n+2
6.3.30b	$7n^2-17n+10$	$3.5n^2-7.5n+4$	$5n^2-13n+8$	$2.5n^2-5.5n+3$
6.3.30c	0	0	0	0
Total	$11n^2+11n-42$	$5.5n^2+32.5n-54$	$7n^2+7n-23$	$3.5n^2+28.5n-47$
$n=6$	420	339	271	250

Table 6.1 : Operations Count for Implementing Algorithm 6.2.

For the sake of comparison, the operations counts for a number of algorithms reported in the literature for computing the inertia matrix of a robot manipulator is given in the following table :

Authors	Remarks	Multiplications	Additions
Walker and Orin	Composite rigid bodies	$12n^2+56n-27$ (741)†	$7n^2+67n-56$ (601)
Angeles and Ma	Natural Orthogonal Complements	$n^3+17n^2-15n+8$ (746)	$n^3+14n^2-16n+5$ (629)
Featherstone	Composite rigid bodies	$10n^2+31n-41$* (505)	$6n^2+40n-46$* (410)
Alg. 6.2 in this thesis	Generalized and augmented links	$11n^2+11n-42$ (420)	$5.5n^2+32.5n-54$ (339)

(.)† Number of operations for $n = 6$.

*A spatial arithmetic package has been used for this implementation.

Table 6.2 : Comparison of Computational Complexities of Several Algorithms for Computing the Joint-Space Inertia Matrix.

From Table 6.2, it is obvious that the proposed algorithm is computationally more efficient than other well known algorithms reported in the literature. The significantly higher computational efficiency of Algorithm 6.2 is obtained primarily through appropriate modeling and use of tensor analysis in the formulation of its basic equations.

Finally, as we mentioned in Section 6.2, one of the basic approach for solving the forward dynamics problem, is to obtain and solve a set of simultaneous equations in the unknown joint accelerations, i.e., steps (*i*)–(*iii*) in Walker and Orin's approach. Following this approach, one can use Algorithm 6.2 for computing the joint space inertia matrix D in step (*i*), Algorithm 5.5 for computing the bias vector b in step (*ii*) and any standard method [18-20] for solving the system of linear equations in step (*iii*). The total computational cost for solving these three subproblems of forward dynamics is given in the following table. Also, for the sake of comparison, the following table contains the computational cost of computing the joint accelerations by other similar methods reported in the literature.

Authors	Remarks	Multiplications	Additions
Kazerounian and Gupta	Zero reference positions	2468	1879
Walker and Orin	Composite rigid bodies	1627	1261
Wang and Ravani	Modified Walker and Orin's	1650	1252
Featherstone	Articulated bodies in spatial notation	1533	1415
Featherstone	Composite rigid bodies in spatial notation	1303	1019
Angeles and Ma	Natural orthogonal complement	1353	1165
(i) Alg. 6.2 in this thesis (ii) inverse dynamics (Alg. 5.5) (iii) Sol. of a linear system [19]	Generalized and augmented links	956	700

Table 6.3 : Computational Cost for Solving Steps (i)-(iii) of the Forward Dynamics Problem for $n = 6$.

6.5 Concluding Remarks

In this chapter, we have presented a new algorithm for computing the joint space inertia matrix D of the generalized inertia tensor $\mathbf{D}$ of a robot manipulator. We have shown that this algorithm can be derived based on either Newton-Euler or Euler-Lagrange formulations of robot dynamics. Thus, we have established that from an algorithmic point of view, the solution of the forward dynamics problem does not depend on which of these formulations is used. A comparison of the computational complexity of this algorithm with that of other existing ones shows that the proposed algorithm is significantly more efficient. This efficiency is achieved mainly because the underlying modeling scheme used here for the dynamic analysis allows us to compute several quantities off-line. Moreover, the computational efficiency is improved since the tensor formulation for the equations to be computed online is computationally more efficient than the traditional vector formulation. Finally, we have shown that by using inverse dynamics algorithms from Chapter V to evaluate the bias vector $\mathbf{b}$, and the proposed algorithm to evaluate the generalized inertia tensor $\mathbf{D}$, the computational cost for solving the forward dynamics problem can be reduced considerably.

6.6 REFERENCES

[1] M. W. Walker and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", *ASME J. Dynamic Systems, Measurement and Control*, Vol. 104 pp 205-211, 1982.

[2] R. Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Boston, MA, 1987.

[3] J. Angeles and O. Ma, "Dynamic Simulation of n-Axis Serial Robotic Manipulators Using a Natural Orthogonal Complement", *Int. Journal of Robotics Research*, pp. 32-45, Vol. 7, No. 5, 1988.

[4] J. Angeles and S. K. Lee. "The Formulation od Dynamical Equations of Holonomic Mechanical Systems Using a Natural Orthogonal Complement", *ASME J. of Applied Mechanics*, Vol. 55, pp. 243-244, 1988.

[5] W. W. Armstrong, "Recursive Solution to the Equations of Motion of an n Link Manipulator", *Proc. 5th World Congress on the Theory of Machines and Mechanisms*, pp. 1343-1346, Vol. 2, Montreal, July, 1979.

[6] G. Rodriguez, "Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics", *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 6, pp. 624-639, 1987.

[7] G. Rodriguez, "Recursive Forward Dynamics for Multiple Robot Arms Moving a Common Task Object" *Robotics and Manufacturing : Recent Trends in Research, Education, and Applications*, M. Jamshidi, J. Y. S. Luh, H. Seraji and G. P. Starr, Eds., ASME Press, New York, 1988.

[8] G. Rodriguez and K. Kreutz, "Recursive and Mass Matrix Factorization and Inversion : An Operator Approach to Manipulator Forward Dynamics", *JPL Publication 88-11*, March 1988.

[9] J. C. K. Chou, G. Baciu and H. K. Kesavan, "Graph-Theoretic Models for Simulating Robot Manipulators", *Proc IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp 953-959, 1987.

[10] J. C. K. Chou, G. Baciu and H. K. Kesavan, "Computational Scheme for Simulating Robot Manipulators", *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp 961-966, 1987.

[11] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithms For Robot Forward Dynamics Computation", *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp 654-659, 1987.

[12] M. Amin-Javaheri and D. E. Orin, "A Systolic Architecture for Computation of the Manipulator Inertia Matrix", *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp 647-653, 1987.

[13] J. Y. Han, "Computational Aspects of Real Time Simulation of Robotic Systems", *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp 967-972, 1987.

[14] J. Wittenburg, *Dynamics of Systems of Rigid Bodies*, Stuttgart : B. G. Teubner, 1977.

[15] K. Kazerounian and K. C. Gupta, "Manipulator Dynamics Using the Extended Zero Reference Position Description", *IEEE Journal of Robotics and Automation*, Vol. RA-2, pp 221-224, 1986.

[16] L. T. Wang and B. Ravani, "Recursive Computations of Kinematic and Dynamic Equations for Mechanical Manipulators", *IEEE Journal of Robotics and Automation*, Vol. RA-1, pp 124-131, 1985.

[17] C. A. Balafoutis and R. V. Patel, "Efficient Computation of Manipulator Inertia Matrices and the Direct Dynamics Problem", to appear in *IEEE Trans. on Systems, Man, and Cybernetics*, September, 1989.

[18] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, N.Y., 1973.

[19] J. J. Dongarra at al., *LINPACK : user's guide*, SIAM, Philadelphia, PA, 1973.

[20] R. L. Burden, J. D. Faires and A. C. Reynolds, *Numerical Analysis*, Prindle, Weder & Schmidt, Boston, MA, 1978.

CHAPTER VII

LINEARIZED DYNAMIC MODELS FOR RIGID-LINK OPEN-CHAIN ROBOT MANIPULATORS

7.1 INTRODUCTION

As is well known, our modeling approach to the real world is based on idealizations and usually the working conditions are not the predicted ones. Therefore, in practice, one has to take into account the effects of perturbations on the applications being considered. For example, let us consider the trajectory tracking problem : In this problem, an important objective is to ensure that the end-effector of a manipulator tracks a desired "nominal" trajectory as closely as possible. Under ideal conditions a dynamic robot model, such as these presented in Chapter V, will provide the generalized force τ which will drive the end-effector of a robot manipulator along the desired trajectory. However, in practice, if only *feedforward* control signals (calculated from a nonlinear dynamic model) are applied, the end-effector will not necessarily track the nominal trajectory. This is due to factors such as modeling uncertainties, gear backlash and friction, actuator and sensor errors, payload variations, etc. which are not taken into account in the dynamic model. Therefore, a feedback/feedforward control system is needed to remedy this situation.

In general, manipulator control is a challenging problem since, as we showed in Chapter V, a dynamic robot model is described by equations which are highly non-linear and dynamically coupled. Obviously, for these inherently nonlinear dynamical systems much of the well established linear control theory is not directly applicable. However, many proposed solutions to the manipulator control problem [1-6] involve the use of methods from linear control theory. For example, in the aforementioned trajectory tracking problem, a control strategy which allows linear control methods to be used can

be outlined as follows : First, feedforward control signals are applied to drive the manipulator along a desired nominal trajectory. Then, a correction term for the feedforward signals is generated by *feedback* of the perturbations (errors) in the physical state-variables (positions and velocities) from their nominal values. This is done using a control algorithm designed for the linearized dynamic equations of the manipulator -henceforth called the *linearized dynamic robot model.* Thus, linearized dynamic robot models which can be obtained in a computationally efficient manner are important in manipulator control. Furthermore, linearized dynamic robot models may be used in other aspects of robotics as well. For example, linearized robot models lead naturally to *trajectory sensitivity functions* [7-9] which characterize the sensitivity of the manipulator motion (along a nominal path) to fixed kinematic or dynamic parameters. Among other applications, these functions can be used [9,10] to describe the variations in a manipulator's trajectory introduced by an unknown pay-load.

The outline of this Chapter is as follows : In Section 7.2 we briefly discuss some basic approaches taken to linearize the dynamic equations of robot manipulators. In Section 7.3 we present a procedure for deriving joint space linearized robot models in a computationally efficient manner. In Section 7.4, we introduce the Cartesian configuration space description for the dynamic equations of robot manipulators and propose a method for their linearization. Finally, Section 7.5 concludes the Chapter.

7.2 LINEARIZATION TECHNIQUES

In this section, we briefly discuss various linearization techniques that have been employed to linearize the dynamic equations of a robot manipulator. Broadly, these techniques may categorized as *global* linearization techniques and *local* linearization techniques. As the name suggests, in global linearization the resulting linearized dynamic robot model is valid over the whole domain where the nonlinear model itself is valid. Alternatively, in local linearization the resulting linearized dynamic robot model is valid

only at a particular point of a given trajectory which is known as the *nominal trajectory.* In local linearization, if the manipulator does not operate over a small range of its variables, then as the manipulator moves the operating point has to be moved along with it and so, in this case, at each new operating point a new (local) linearization has to be performed. Briefly, global and local linearization can be achieved by using the following methods.

7.2.1 Global Linearization

Global linearization resulted from efforts to control nonlinear systems, and can be achieved by using either feedback action alone or feedback action combined with coordinate transformations.

In the first approach, i.e., when feedback action alone is used, the main idea is to "cancel" the nonlinearities in the nonlinear system by using an appropriate feedback law which makes the overall closed-loop system to behave as a linear system. The resolved acceleration technique [5] is probably the simplest and most common technique used for achieving "feedback linearization" of a robotic system. Also, another interesting feedback linearization technique is described in [6]. In these methods, under the feedback action, the closed-loop response of the system (which may be error signals or end-effector position variables) is described by a linear second order differential equation which is viewed as the linearized dynamic robot model. The computational complexity in this "linearizing control" approach varies with the method and the particular feedback law. However, in general a large amount of computation is required which in some cases includes matrix inversion in real-time [5]. Moreover, besides the complexity, a practical difficulty with this approach is that inexact cancellation of the nonlinearities can result in a 'linearized' model which gives very poor stability performance for the closed-loop system.

The feedback and coordinate transformation approach for linearizing robot mani-

pulators resulted from advances made in the past few decades in the differential geometric system theory. The main idea in this approach, is to use a diffeomorphic† feedback-transformation (which includes state space change of coordinates, additive feedback and control (input) space change of coordinates) and transfer the nonlinear dynamic robot model to an equivalent linear and output decoupled system. In general, a diffeomorphic feedback transformation of this type exists under certain necessary and sufficient conditions for a class of nonlinear systems [11-12]. Necessary and sufficient conditions for the existence of a diffeomorphic feedback-transformation applicable to dynamic robot models and a method for constructing it have been given in [13]. At first, this method of linearization is attractive because it guarantees exact linearization. However, it requires an extensive analysis and the feedback transformation law is quite complex. Thus, the algorithms which compute this control law are computationally expensive and, in practical applications, this offsets the advantages gained from the exact linearization.

7.2.2 Local Linearization

In this approach, fundamental to linearizing a nonlinear system is the concept of a nominal trajectory which will be denoted here by $(\mathbf{q}^o(t), \dot{\mathbf{q}}^o(t), \ddot{\mathbf{q}}^o(t))$. Also, corresponding to the nominal trajectory there is a *nominal generalized force* vector $\tau^o(t)$ which can be computed by using any of the nonlinear dynamic robot models of Chapter V. In the following, for the sake of notational simplicity, we shall drop the time parameter t from all the time dependent terms.

Local linearization is based on the Taylor series expansion of the nonlinear dynamic equations about a nominal trajectory. The Taylor series expansion is applicable to these equations since they are analytic functions of their arguments. The approach is concep-

† A diffeomorphic transformation is a one-to-one onto C^∞ transformation between two manifolds such that its inverse exists and is also C^∞.

tually simple. We assume that the perturbations are small and we consider the first order approximation to this expansion. To this effect, the nominal portion of this expansion is cancelled algebraically and the terms which are *linear* in the perturbation quantities are retained and define the linearized dynamic equations. Following this procedure, we can express the linearized dynamic equations of a robot manipulator by a closed-form linear vector equation or by a recursive algorithm which has the same structure as the recursive algorithms which solves the IDP.

A closed-form linearized dynamic robot model, which can be derived by applying the Taylor series expansion to one of the recursive dynamic robot models of Chapter V, is written in the form

$$\delta\tau = \mathbf{D}^o\,\delta\ddot{\mathbf{q}} + \mathbf{V}^o\,\delta\dot{\mathbf{q}} + \mathbf{P}^o\,\delta\mathbf{q} \tag{7.2.1}$$

where $\delta\tau, \delta\mathbf{q}, \delta\dot{\mathbf{q}}, \delta\ddot{\mathbf{q}} \in \mathbb{R}^n$ are small deviations about some nominal torque and nominal joint space trajectory. The coefficients in equation (7.2.1) are known as the *(coefficient) sensitivity matrices* of the linearized model. These, are functions of the nominal trajectory $(\mathbf{q}^o, \dot{\mathbf{q}}^o, \ddot{\mathbf{q}}^o)$ and are independent of the perturbations. Here, following the established terminology, we call the coefficients in equation (7.2.1) "matrices", although actually they are 2*nd* order tensors. The tensor character of these quantities is obvious from their definitions which are presented next :

$\mathbf{D}^o$ The *inertial force-acceleration* sensitivity matrix is the Jacobian of the generalized force vector τ with respect to $\ddot{\mathbf{q}}$ evaluated about the nominal trajectory i.e., $\mathbf{D}^o \equiv \nabla_{\ddot{\mathbf{q}}}\tau\,|_{(\mathbf{q}^o,\dot{\mathbf{q}}^o,\ddot{\mathbf{q}}^o)}$.

$\mathbf{V}^o$ The *centrifugal and Coriolis force-velocity* sensitivity matrix is the Jacobian of τ with respect to $\dot{\mathbf{q}}$ evaluated about the nominal trajectory i.e., $\mathbf{V}^o \equiv \nabla_{\dot{\mathbf{q}}}\tau\,|_{(\mathbf{q}^o,\dot{\mathbf{q}}^o,\ddot{\mathbf{q}}^o)}$.

$\mathbf{P}^o$ The *force-position* sensitivity matrix is the Jacobian of τ with respect to $\mathbf{q}$ evaluated about the nominal trajectory i.e., $\mathbf{P}^o \equiv \nabla_{\mathbf{q}}\tau\,|_{(\mathbf{q}^o,\dot{\mathbf{q}}^o,\ddot{\mathbf{q}}^o)}$.

To derive a recursive linearized dynamic robot model we again apply the Taylor series expansion to a nonlinear recursive dynamic robot model. But in this case, the joint space perturbations $\delta\tau$ of the generalized force vector τ are computed componentwise by a recursive algorithm which has the same structure as the recursive algorithm which computes the nonlinear manipulator dynamics. The only difference is that now instead of propagating the actual velocities, accelerations, forces, etc. from link to link we propagate the perturbations of these quantities about the operating point of a nominal trajectory. Thus it can be shown [9] that recursive linearized dynamic robot models retain the $O(n)$ computational complexity of a recursive nonlinear dynamic robot model. However, recursive linearization has limited applications in manipulator control, since it is difficult to derive the *state-space representation* (which is used for the time domain analysis and control) of the linearized manipulator dynamics directly from a recursive model. On the other hand, the state-space representation of the linearized manipulator dynamics can be derived easily from a closed-form linearized dynamic robot model.

Thus, for example, since the force acceleration matrix $\mathbf{D}^o \in \mathbb{R}^{n \times n}$ is positive definite [14] its inverse always exists and this allow us to write equation (7.2.1) as

$$\begin{bmatrix} \delta\dot{\mathbf{q}} \\ \delta\ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_n & \mathbf{1}_n \\ -(\mathbf{D}^o)^{-1}\mathbf{P}^o & -(\mathbf{D}^o)^{-1}\mathbf{V}^o \end{bmatrix} \begin{bmatrix} \delta\mathbf{q} \\ \delta\dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} 0 \\ (\mathbf{D}^o)^{-1} \end{bmatrix} \delta\tau \tag{7.2.2}$$

where $\mathbf{0}_n$ and $\mathbf{1}_n$ are the zero and unity $n \times n$ matrices, respectively. Equation (7.2.2) is in the standard state-space form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \tag{7.2.3}$$

where $\mathbf{x}(t)\ (= [\delta\mathbf{q}^T(t)\ \delta\dot{\mathbf{q}}^T(t)]) \in \mathbb{R}^{2n}$, and $\mathbf{u}(t)\ (= \delta\tau(t)) \in \mathbb{R}^n$.

To compute the coefficient sensitivity matrices of a closed-form linearized dynamic robot model we can use one of the following methods.

a) *parameter identification techniques* : In this method, a discrete-time version of the linearized model is considered first and then an iterative scheme, such as the least-

squares parameter identification algorithm, can be used to evaluate the unknown parameters in the sensitivity matrices. This approach has been used in [4] to calculate directly the coefficient matrices **A** and **B** of the state space linearized robot model (7.2.3). However, it should be noticed that identification schemes can be used effectively only when the parameters of the system are *slowly time-varying*. Also, even in slowly time-varying systems the convergence of the iterative algorithm may present a problem.

b) *analytic or recursive formulations* : In this approach we first obtain *analytic* or *recursive* expressions for the elements of the Jacobians, and then devise algorithms which evaluate them numerically or symbolically along points on the nominal trajectory. In this approach, we do not face the aforementioned restrictions of the parameter identification techniques, but this method can lead us to computationally expensive algorithms. Linearized robot models based on this approach have been proposed in [8,15]. In both cases, the 4x4 Lagrangian formulation of robot dynamics has been used, and this results in linearized robot models which are computationally inefficient since they inherit the computational complexity associated with the 4x4 formulation of the Lagrangian dynamic robot model. Also, in [14] the linearization of the Newton-Euler formulation of robot dynamic models has been derived in parallel with the nonlinear dynamic equations. This approach is also computationally inefficient since, as has been estimated in [14], one has to evaluate $n^3 + 2n^2$ terms in order to obtain the coefficient sensitivity matrices.

In the following, using recursive formulations, we propose a new method for deriving the Jacobians which define the coefficient sensitivity matrices of a linearized dynamic robot model, and devise algorithms for evaluating these Jacobians in a computational efficient manner.

7.3 Joint Space Linearized Dynamic Robot Models

The application of the Taylor series expansion to the joint space nonlinear dynamic

equations of a robot manipulator, at least in principle, does not present any problem. However, the complexity of these nonlinear equations makes that task a challenging problem, specially if one attempts to derive efficient computational algorithms for determining the coefficient sensitivity matrices of the closed-form linearized dynamic robot model such as that of equation (7.2.1). In this section we shall address this problem and demonstrate how the various coefficient sensitivity matrices in equation (7.2.1) can be evaluated in a computationally efficient manner.

7.3.1 Joint Space Coefficient Sensitivity Matrices

By definition, the coefficient sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$ of a *joint space linearized dynamic robot model* are the Jacobians of the generalized force vector τ with respect to the generalized coordinates $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$ and $\mathbf{q}$, respectively, evaluated at a point of a nominal trajectory $(\mathbf{q}^o, \dot{\mathbf{q}}^o, \ddot{\mathbf{q}}^o)$. Thus, for example, if the generalized force vector τ is defined component-wise by equation (5.3.8c) of Algorithm 5.4 in Chapter V, we can compute the elements of the sensitivity matrix $\mathbf{D}^o$ by using the following equation,

$$\begin{aligned} d_{ij}^o &= \frac{\partial \tau_i}{\partial \ddot{q}_j}\Big|_{(\mathbf{q}^o, \dot{\mathbf{q}}^o, \ddot{\mathbf{q}}^o)} \\ &= \sigma_i \left(\frac{\partial \eta_i^i}{\partial \ddot{q}_j} \cdot \mathbf{z}_i^i \right)\Big|_{(\mathbf{q}^o, \dot{\mathbf{q}}^o, \ddot{\mathbf{q}}^o)} + (1-\sigma_i) \left(\frac{\partial \mathbf{f}_i^i}{\partial \ddot{q}_j} \cdot \mathbf{z}_i^i \right)\Big|_{(\mathbf{q}^o, \dot{\mathbf{q}}^o, \ddot{\mathbf{q}}^o)} \end{aligned} \tag{7.3.1}$$

where σ_i is equal to one when the i-th joint is revolute and zero when the i-th joint is prismatic. The elements of the sensitivity matrices $\mathbf{V}^o$ and $\mathbf{P}^o$ are defined in an analogous manner. Therefore, it is obvious that if we have available the various partial derivatives of the vector functions η_i^i and $\mathbf{f}_i^i$, we can easily compute the joint space sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$ for robot manipulators with revolute and /or prismatic joints. In this section, in order to derive a procedure for computing the coefficient sensitivity matrices of joint space linearized robot models, we shall consider manipulators with revolute joints only. In the case of manipulators with some prismatic joints, the equations are valid with minor modifications. Moreover, for the sake of

continuity, only the main results will be presented in this section. Salient steps (such as partial differentiation of various vector functions) for arriving at these results are outlined in Appendix C.

In the following, we shall assume that the generalized force vector τ is defined by equation (5.3.46c) of Algorithm 5.5, and that this algorithm is applied to manipulators with revolute joints only. As we have shown in Chapter V, the tensor formulation of Algorithm 5.5 makes its implementation computationally very efficient. Therefore, Cartesian tensor analysis will by used here as well and the basic equations will be stated in a tensor formulation.

Based on the definition of the appropriate Jacobians, the expressions for the elements of the sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$ may be determined as given below :

i) Inertial Force-Acceleration Sensitivity Matrix $\mathbf{D}^o$:

By definition, we can write the (i,j)-th element of $\mathbf{D}^o$ as

$$d^o_{ij} = \frac{\partial \tau_i}{\partial \ddot{q}_j}\Big|_{(\mathbf{q}^o,\dot{\mathbf{q}}^o,\ddot{\mathbf{q}}^o)}$$

$$= \left(\mathbf{z}^i_i \cdot \frac{\partial \eta^i_i}{\partial \ddot{q}_j}\right)\Big|_{(\mathbf{q}^o,\dot{\mathbf{q}}^o,\ddot{\mathbf{q}}^o)}$$

which may be simplified (see Appendix C) to

$$d_{ij} = \begin{cases} \mathbf{z}^i_i \cdot \left(\mathbf{E}^i_{O_i} - \tilde{\mathbf{U}}^i_{O_i}\, \tilde{\mathbf{s}}^i_{j,i}\right) \cdot \mathbf{z}^i_j & j \leq i \\ \mathbf{z}^i_i \cdot {}^i\mathbf{W}_j \left(\left[\mathbf{E}^j_{O_j} - \tilde{\mathbf{s}}^j_{i,j} \tilde{\mathbf{U}}^j_{O_j}\right] \cdot \mathbf{z}^j_j\right) & j > i \end{cases}$$

$$= \begin{cases} \mathbf{z}^i_i \cdot \left(\mathbf{E}^i_{O_i} - \tilde{\mathbf{U}}^i_{O_i}\, \tilde{\mathbf{s}}^i_{j,i}\right) \cdot \mathbf{z}^i_j & j \leq i \\ \mathbf{z}^j_i \cdot \left(\mathbf{E}^j_{O_j} - \tilde{\mathbf{s}}^j_{i,j} \tilde{\mathbf{U}}^j_{O_j}\right) \cdot \mathbf{z}^j_j & j > i \end{cases} \qquad (7.3.2)$$

where the last step follows from the fact that the dot product is invariant under coordinate transformations.

ii) Centrifugal and Coriolis Force-Velocity Sensitivity Matrix $\mathbf{V}^o$:

The elements of the sensitivity matrix $\mathbf{V}^o$ are defined by considering the following Jacobian,

$$v_{ij}^{\circ} = \frac{\partial \tau_i}{\partial \dot{q}_j}\Big|_{(\mathbf{q}^\circ, \dot{\mathbf{q}}^\circ, \ddot{\mathbf{q}}^\circ)} = \left[\mathbf{z}_i^i \cdot \frac{\partial \eta_i^i}{\partial \dot{q}_j} \right]\Big|_{(\mathbf{q}^\circ, \dot{\mathbf{q}}^\circ, \ddot{\mathbf{q}}^\circ)}$$

which, on simplification (see Appendix C), gives

$$v_{ij} = 2 \begin{cases} \mathbf{z}_i^i \cdot \left(\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i \, \tilde{\mathbf{s}}_{j,i}^i \right) \cdot \dot{\mathbf{z}}_j^i - \mathbf{z}_i^i \cdot \left(\mathbf{L}_i^i + \tilde{\mathbf{U}}_{O_i}^i \, \dot{\tilde{\mathbf{s}}}_{j,i}^i \right) \cdot \mathbf{z}_j^i & j \leq i \\ \mathbf{z}_i^i \cdot {}^i\mathbf{W}_j \left([\mathbf{E}_{O_j}^j - \tilde{\mathbf{s}}_{i,j}^j \tilde{\mathbf{U}}_{O_j}^j] \cdot \dot{\mathbf{z}}_j^j - \mathbf{z}_i^j \cdot [\mathbf{L}_j^j + \tilde{\mathbf{s}}_{i,j}^j \dot{\tilde{\mathbf{U}}}_{O_j}^j] \cdot \mathbf{z}_j^j \right) & j > i \end{cases}$$

$$= 2 \begin{cases} \mathbf{z}_i^i \cdot \left(\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i \, \tilde{\mathbf{s}}_{j,i}^i \right) \cdot \dot{\mathbf{z}}_j^i - \mathbf{z}_i^i \cdot \left(\mathbf{L}_i^i + \tilde{\mathbf{U}}_{O_i}^i \, \dot{\tilde{\mathbf{s}}}_{j,i}^i \right) \cdot \mathbf{z}_j^i & j \leq i \\ \mathbf{z}_i^j \cdot \left(\mathbf{E}_{O_j}^j - \tilde{\mathbf{s}}_{i,j}^j \tilde{\mathbf{U}}_{O_j}^j \right) \cdot \dot{\mathbf{z}}_j^j - \mathbf{z}_i^j \cdot \left(\mathbf{L}_j^j + \tilde{\mathbf{s}}_{i,j}^j \dot{\tilde{\mathbf{U}}}_{O_j}^j \right) \cdot \mathbf{z}_j^j & j > i \end{cases} \tag{7.3.3}$$

where

$$\mathbf{L}_i^i = \bar{\omega}_i^i \, \bar{\mathbf{K}}_{O_i}^i + [\tilde{\mathbf{s}}_{i,i+1}^i \dot{\tilde{\mathbf{U}}}_{O_{i+1}}^i + \tilde{\mathbf{U}}_{O_{i+1}}^i \dot{\tilde{\mathbf{s}}}_{i,i+1}^i] + \mathbf{A}_{i+1} \mathbf{L}_{i+1}^{i+1} \mathbf{A}_{i+1}^T,$$

and $\bar{\mathbf{K}}_{O_i}^i$ is the pseudo-inertia tensor of the i-th augmented link which is defined by equation (5.3.44d).

iii) Force-Position Sensitivity Matrix $\mathbf{P}^o$:

As in the case of the elements of $\mathbf{D}^o$ and $\mathbf{V}^o$, we can write

$$p_{ij}^{\circ} = \frac{\partial \tau_i}{\partial q_j}\Big|_{(\mathbf{q}^\circ, \dot{\mathbf{q}}^\circ, \ddot{\mathbf{q}}^\circ)} = \left[\mathbf{z}_i^i \cdot \frac{\partial \eta_i^i}{\partial q_j} \right]\Big|_{(\mathbf{q}^\circ, \dot{\mathbf{q}}^\circ, \ddot{\mathbf{q}}^\circ)}$$

which may be simplified (see Appendix C) to

$$p_{ij} = \begin{cases} \mathbf{z}_i^i \cdot \left[[\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i \, \tilde{\mathbf{s}}_{j,i}^i] \cdot \ddot{\mathbf{z}}_j^i - 2[\mathbf{L}_i^i + \tilde{\mathbf{U}}_{O_i}^i \dot{\tilde{\mathbf{s}}}_{j,i}^i] \cdot \dot{\mathbf{z}}_j^i + [\tilde{\mathbf{U}}_{O_i}^i \, \ddot{\tilde{\mathbf{s}}}_{0,j}^i] \cdot \mathbf{z}_j^i \right] & j \leq i \\ \mathbf{z}_i^i \cdot {}^i\mathbf{W}_j \left([\mathbf{E}_{O_j}^j - \tilde{\mathbf{s}}_{i,j}^j \tilde{\mathbf{U}}_{O_j}^j] \cdot \ddot{\mathbf{z}}_j^j - 2[\mathbf{L}_j^j + \tilde{\mathbf{s}}_{i,j}^j \dot{\tilde{\mathbf{U}}}_{O_j}^j] \cdot \dot{\mathbf{z}}_j^j + [\tilde{\mathbf{U}}_{O_j}^j \, \ddot{\tilde{\mathbf{s}}}_{0,j}^j - \tilde{\mathbf{s}}_{i,j}^j \ddot{\tilde{\mathbf{U}}}_{O_j}^j - \bar{\eta}_j^j] \cdot \mathbf{z}_j^j \right) & j > i \end{cases}$$

$$= \begin{cases} \mathbf{z}_i^i \cdot [\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i \tilde{\mathbf{s}}_{j,i}^i] \cdot \ddot{\mathbf{z}}_j^i - 2\mathbf{z}_i^i \cdot [\mathbf{L}_i^i + \tilde{\mathbf{U}}_{O_i}^i \dot{\tilde{\mathbf{s}}}_{j,i}^i] \cdot \dot{\mathbf{z}}_j^i + \mathbf{z}_i^i \cdot [\tilde{\mathbf{U}}_{O_i}^i \ddot{\tilde{\mathbf{s}}}_{0,j}^i] \cdot \mathbf{z}_j^i & j \leq i \\ \mathbf{z}_i^j \cdot [\mathbf{E}_{O_j}^j - \tilde{\mathbf{s}}_{i,j}^j \tilde{\mathbf{U}}_{O_j}^j] \cdot \ddot{\mathbf{z}}_j^j - 2\mathbf{z}_i^j \cdot [\mathbf{L}_j^j + \tilde{\mathbf{s}}_{i,j}^j \dot{\tilde{\mathbf{U}}}_{O_j}^j] \cdot \dot{\mathbf{z}}_j^j + \mathbf{z}_i^j \cdot [\tilde{\mathbf{U}}_{O_j}^j \ddot{\tilde{\mathbf{s}}}_{0,j}^j - \tilde{\mathbf{s}}_{i,j}^j \ddot{\tilde{\mathbf{U}}}_{O_j}^j - \tilde{\boldsymbol{\eta}}_j^j] \cdot \mathbf{z}_j^j & j > i \end{cases}$$

(7.3.4)

Note that for notational convenience, the explicit dependence on the nominal variables has not been shown in equations (7.3.2), (7.3.3) and (7.3.4).

At a first glance, these equations look very complex. However, a closer examination shows that most of the terms are common. This reduces considerably the computational burden in their implementation. Also, in these equations we can identify the first moment $\mathbf{U}_{O_i}^i$ and the second moment $\mathbf{E}_{O_i}^i$ of the i-th generalized link which, as we mentioned in Chapter VI, can be computed recursively by using equations (6.3.3) and (6.3.6), respectively. Also, as we shall see later, terms such as $\mathbf{z}_j^i$, $\dot{\mathbf{z}}_j^i$ $\mathbf{s}_{j,i}^i$, etc. can be computed recursively. Moreover, the formulation of these equations gives us a valuable insight into the structural characteristics and properties of the coefficient sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$.

It is worth noticing that at a point on the nominal trajectory, equation (7.3.2) is equivalent to equation (6.3.45) which defines the joint space generalized inertia tensor of a robot manipulator. This is to be expected, since in a closed form representation of a nonlinear dynamic robot model the generalized force vector τ is linear in the joint acceleration $\ddot{\mathbf{q}}$, with the generalized inertia tensor $\mathbf{D}$ as coefficient of linearity. Therefore, the force-acceleration sensitivity matrix $\mathbf{D}^o$ exhibits the properties of the generalized inertia tensor $\mathbf{D}$, i.e., it is symmetric (which can also be seen from equation (7.3.2)) and positive definite. The former property of $\mathbf{D}^o$ is obviously important in computational considerations. The latter is important in controller design since it implies that $\mathbf{D}^o$ is nonsingular and thus its inverse exists for all the points along a nominal trajectory.

Unfortunately, symmetry is not preserved in the coefficient sensitivity matrices $\mathbf{V}^o$ and $\mathbf{P}^o$. However, they have other important characteristics which may be used to

simplify their evaluation at points along a nominal trajectory. For example, in [8] it has been shown (following a different analysis) that the element $v_{n,n}$ of $\mathbf{V}^o$ is zero for *any* configuration of the manipulator. This also follows from equation (7.3.3), if we notice the following : When $i = j = n$, we have,

$$\mathbf{E}_{O_n}^n = \mathbf{K}_{O_n}^n$$
$$\dot{\mathbf{z}}_n^n = \tilde{\omega}_n^n \mathbf{z}_n^n$$
$$\mathbf{s}_{n,n}^n = \dot{\mathbf{s}}_{n,n}^n = 0$$

and

$$\mathbf{L}_n^n = \tilde{\omega}_n^n \hat{\mathbf{K}}_{O_n}^n$$
$$= \frac{1}{2} tr\,[\mathbf{K}_{O_n}^n]\tilde{\omega}_n^n - \tilde{\omega}_n^n \mathbf{K}_{O_n}^n .$$

Therefore, for $i = j = n$ equation (7.3.3) implies that

$$v_{n,n} = \mathbf{z}_n^n \cdot \left(\mathbf{K}_{O_n}^n \tilde{\omega}_n^n + \tilde{\omega}_n^n \mathbf{K}_{O_n}^n - \frac{1}{2} tr\,[\mathbf{K}_{O_n}^n]\tilde{\omega}_n^n \right) \cdot \mathbf{z}_n^n$$

which, by using the tensor equation (3.4.25), can be simplified into

$$v_{n,n} = -\,\mathbf{z}_n^n \cdot dual\left(\mathbf{K}_{O_n}^n \tilde{\omega}_n^n\right) \cdot \mathbf{z}_n^n$$
$$= \mathbf{z}_n^n \cdot \tilde{\mathbf{z}}_n^n \cdot \left(\mathbf{K}_{O_n}^n \tilde{\omega}_n^n\right)$$
$$= 0 \qquad (7.3.5)$$

where the first step follows from equation (3.4.4) and the last has been derived by applying equation (3.4.7).

Another important observation, concerning the formulation of the elements of the *first column of the sensitivity matrices* $\mathbf{V}^o$ and $\mathbf{P}^o$, is the following : Since the angular velocity vector is always parallel to the first revolute joint of the manipulator, i.e., the vector ω_1^1 is parallel to $\mathbf{z}_1^1$ (we assume here, as is usually the case, that the first joint is revolute), then according to Remark 3.8 of Chapter III, we can write the following equation

$$\tilde{\omega}_1^1 \cdot \mathbf{z}_1^1 = 0 \qquad (7.3.6)$$

which implies that

$$\dot{\mathbf{z}}_1^1 = \mathbf{0}$$

Furthermore, since equation (7.3.6) is invariant under coordinate transformations, we can write

$$\begin{aligned}\dot{\mathbf{z}}_1^i &= \tilde{\omega}_1^i \cdot \mathbf{z}_1^i \\ &= \mathbf{0}\end{aligned} \tag{7.3.7}$$

for all i. Also, using the same arguments, it can be shown that the vector $\ddot{\mathbf{z}}_1^i$ is zero for all i, i.e., we have

$$\begin{aligned}\ddot{\mathbf{z}}_1^i &= \Omega_1^i \cdot \mathbf{z}_1^i \equiv \left(\dot{\tilde{\omega}}_1^i + \tilde{\omega}_1^i \tilde{\omega}_1^i\right) \cdot \mathbf{z}_1^i \\ &= \mathbf{0}\end{aligned} \tag{7.3.8}$$

Therefore, by using equations (7.3.7) and (7.3.8) in equations (7.3.3) and (7.3.4), we can simplify the formulations for the elements of the first column of $\mathbf{V}^o$ and $\mathbf{P}^o$. Moreover, in the case where the first joint of the manipulator rotates about an axis which is parallel to the gravity field the first column of the sensitivity matrix $\mathbf{P}^o$ becomes equal to zero. This is true since, by assuming that the vector $\mathbf{z}_1^1$ is parallel to the gravity vector $\mathbf{g} = \ddot{\mathbf{s}}_{0,1}^1$, we have

$$\ddot{\tilde{\mathbf{s}}}_{0,1}^1 \cdot \mathbf{z}_1^1 = 0 \tag{7.3.9}$$

which follows from equation (3.4.8) (see Remark 3.8). Now, as with equation (7.3.6), in the i-th coordinate system we have

$$\ddot{\tilde{\mathbf{s}}}_{0,1}^i \cdot \mathbf{z}_1^i = 0 \tag{7.3.10}$$

and this, together with equation (7.3.6), implies that the first column of the coefficient sensitivity matrix $\mathbf{P}^o$ is zero in the case where the first revolute joint of the manipulator is parallel to the gravity field. An alternative proof of this result can be found in [8].

With these observations on the structural characteristics of the sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$ we proceed to analyze the numerical implementation of equations (7.3.2)-(7.3.4).

7.3.2 Implementation and Computational Considerations

As with the numerical implementation of various tensor equations in Chapters V and VI , the same observations and terminology will be applied in this section for the numerical implementation of equations (7.3.2)-(7.3.4). Moreover, for the implementation of these equations, we shall assume that the nonlinear dynamic robot model which is described by Algorithm 5.5 is available. This assumption is justified since in control applications the generalized force vector τ is an integral part of most control laws. Thus, quantities such as : $\mathbf{u}_{O_i}^i$, $\tilde{\omega}_i^i$, Ω_j^j, $\ddot{\mathbf{s}}_{0,j}^j$, $\mathbf{K}_{O_i}^i$ and $\hat{\mathbf{K}}_{O_i}^i$ are assumed to be available from Algorithm 5.5. From the foregoing, only the quantities which are computed in the following two algorithms are needed to be known for the evaluation of equations (7.3.2), (7.3.3) and (7.3.4).

ALGORITHM 7.1

Step 0 : Initialization :- $j=1,\ n-1$

$$z_j^j = [\,0\ 0\ 1\,]^T,\ \dot{\mathbf{z}}_j^j = \tilde{\omega}_j^j \mathbf{z}_j^j,\ \ddot{\mathbf{z}}_j^j = \Omega_j^j \mathbf{z}_j^j,\ \mathbf{s}_{j,j}^j = \dot{\mathbf{s}}_{j,j}^j = 0 \tag{7.3.11}$$

Step 1 : Forward Recursion :- $i=j+1,n$

$$\mathbf{z}_j^i = \mathbf{A}_i^T z_j^{i-1} \tag{7.3.12a}$$

$$\dot{\mathbf{z}}_j^i = \mathbf{A}_i^T \dot{z}_j^{i-1} \tag{7.3.12b}$$

$$\ddot{\mathbf{z}}_j^i = \mathbf{A}_i^T \ddot{z}_j^{i-1} \tag{7.3.12c}$$

$$\ddot{\mathbf{s}}_{0,j}^i = \mathbf{A}_i^T \ddot{s}_{0,j}^{i-1} \tag{7.3.12d}$$

$$s_{j,i}^i = \mathbf{A}_i^T \left[s_{j,i-1}^{i-1} + s_{i-1,i}^{i-1} \right] \tag{7.3.12e}$$

$$\dot{s}_{j,i}^i = \mathbf{A}_i^T \left[\dot{s}_{j,i-1}^{i-1} + \tilde{\omega}_{i-1}^{i-1} s_{i-1,i}^{i-1} \right] \tag{7.3.12f}$$

end.

ALGORITHM 7.2

Step 0 : Initialization :- $i=n$

$$\mathbf{U}_{O_n}^n = \mathbf{u}_{O_n}^n,\ \dot{\mathbf{U}}_{O_n}^n = \tilde{\omega}_n^n \mathbf{u}_{O_n}^n,\ \mathbf{E}_{O_n}^n = \mathbf{K}_{O_n}^n,\ \mathbf{L}_n^n = \tilde{\omega}_n^n \hat{\mathbf{K}}_{O_n}^n \tag{7.3.13}$$

Step 1 : Backward Recursion :- $i=n-1,\ 1$

$$\mathbf{U}_{O_i}^i = \mathbf{u}_{O_i}^i + \mathbf{A}_{i+1} \mathbf{U}_{O_{i+1}}^{i+1} \tag{7.3.14a}$$

$$\dot{\mathbf{U}}_{O_i}^{i} = \bar{\omega}_i^i \mathbf{u}_{O_i}^{i} + \mathbf{A}_{i+1}\dot{\mathbf{U}}_{O_{i+1}}^{i+1} \tag{7.3.14b}$$

$$\mathbf{E}_{O_i}^{i} = \mathbf{K}_{O_i}^{i} + \mathbf{A}_{i+1}\left\{\mathbf{E}_{O_{i+1}}^{i+1} - \left[\tilde{\mathbf{s}}_{i,i+1}^{i+1}\tilde{\mathbf{U}}_{O_{i+1}}^{i+1} + (\tilde{\mathbf{s}}_{i,i+1}^{i+1}\tilde{\mathbf{U}}_{O_{i+1}}^{i+1})^T\right]\right\}\mathbf{A}_{i+1}^T \tag{7.3.14c}$$

$$\mathbf{L}_i^i = \bar{\omega}_i^i \tilde{\mathbf{K}}_{O_i}^{i} + \mathbf{A}_{i+1}\left\{\mathbf{L}_{i+1}^{i+1} + \left[\tilde{\mathbf{s}}_{i,i+1}^{i+1}\dot{\tilde{\mathbf{U}}}_{O_{i+1}}^{i+1} + \tilde{\mathbf{U}}_{O_{i+1}}^{i+1}\tilde{\mathbf{s}}_{i,i+1}^{i+1}\right]\right\}\mathbf{A}_{i+1}^T \tag{7.3.14d}$$

end.

Let us consider first the numerical implementation of Algorithms 7.1 and 7.2. As in Chapters V and VI, we shall consider two classes of robot manipulators, namely, robot manipulators with a general geometric structure and robot manipulators for which the twist angles are, by design, either 0 or 90 degrees. In the following, when we refer to multiplications or additions we mean scalar multiplications or scalar additions.

As we can see, the structure of the equations in Algorithm 7.2 is the same as that of the equations in Algorithm 6.2. In particular, equations (7.3.14a), and (7.3.14c) are exactly the same as equations (6.3.29a) and (6.3.29c), respectively. Therefore, observations made in Section 6.4 can also be used here. For example, to implement equation (7.3.14c) in the general case (i.e., when the twist angle is different from 0 and 90 degrees), we need 30 multiplications and 39 additions. Also, using the trigonometric identities of Section 6.4, it can be shown that the coordinate transformation $\mathbf{A}_{i+1}\mathbf{L}_{i+1}^{i+1}\mathbf{A}_{i+1}^T$ can be implemented with 36 (18) multiplications and 32 (20) additions when the twist angle is different from (equal to) 0 or 90 degrees. Moreover, since we need 9 multiplications and 3 additions for the multiplication of two skew-symmetric matrices, we require 18 multiplications and 15 additions to compute the matrix $\left(\tilde{\mathbf{s}}_{i,i+1}^{i+1}\dot{\tilde{\mathbf{U}}}_{O_{i+1}}^{i+1} + \tilde{\mathbf{U}}_{O_{i+1}}^{i+1}\tilde{\mathbf{s}}_{i,i+1}^{i+1}\right)$. Finally, to implement the term $\bar{\omega}_i^i\tilde{\mathbf{K}}_{O_i}^{i}$ we need another 15 multiplications and 9 additions. Therefore, to implement equation (7.3.14d), we need 69 scalar multiplications and 74 scalar additions in the general case.

The implementation of Algorithm 7.1 is straightforward. Each equation needs to be evaluated $n(n-1)/2$ times. However, if equations (7.3.7) and (7.3.8) are taken into

account, the terms $\dot{\mathbf{z}}_j^i$ and $\ddot{\mathbf{z}}_j^i$ need to be evaluated $(n-1)(n-2)/2$ times. Moreover, when the first joint is parallel to the gravity field, the vector $\ddot{\mathbf{s}}_{0,1}^i$ is a scalar multiple (by $g = 9.81$) of $\mathbf{z}_1^i$ and this simplifies the computations. Also, as we mentioned in Section 6.4, some saving in computations can be made in computing the vectors $\mathbf{s}_{j,j+1}^{j+1} = \mathbf{A}_{j+1}^T \mathbf{s}_{j,j+1}^j$. The same is obviously true for the vectors $\mathbf{z}_j^{j+1}$.

Based on these observations, a breakdown of the number of scalar multiplications and additions required for the implementation of each equation in Algorithms 7.1 and 7.2 is given in Table 7.1. The total figure represents the operations count for computing these equations when all the joints of a robot manipulator are of revolute type.

	Manipulator with general twist angles		Manipulator with twist angles equal to 0^o or 90^o	
Equation	Multiplications	Additions	Multiplications	Additions
7.3.12a	$4n^2 - 10n + 6$	$2n^2 - 6n + 4$	$2n^2 - 6n + 4$	$n^2 - 3n + 2$
7.3.12b	$4n^2 - 12n + 8$	$2n^2 - 6n + 4$	$2n^2 - 6n + 4$	$n^2 - 3n + 2$
7.3.12c	$4n^2 - 12n + 8$	$2n^2 - 6n + 4$	$2n^2 - 6n + 4$	$n^2 - 3n + 2$
7.3.12d	$4n^2 - 9n + 5$	$2n^2 - 6n + 4$	$2n^2 - 3n + 1$	$n^2 - 3n + 2$
7.3.12e	$4n^2 - 10n + 6$	$3.5n^2 - 11.5n + 7$	$2n^2 - 6n + 4$	$2.5n^2 - 7.5n + 5$
7.3.12f	$7n^2 - 7n$	$5n^2 - 5n$	$5n^2 - 5n$	$4n^2 - 4n$
7.3.14a	$8n - 8$	$7n - 7$	$4n - 4$	$5n - 5$
7.3.14b	$14n - 14$	$10n - 10$	$10n - 10$	$8n - 8$
7.3.14c	$30n - 30$	$39n - 39$	$20n - 20$	$32n - 32$
7.3.14d	$69n - 69$	$74n - 74$	$51n - 51$	$63n - 63$
Total	$27n^2 + 61n - 88$	$16.5n^2 + 100.5n - 107$	$15n^2 + 53n - 68$	$10.5n^2 + 84.5n - 95$
$n = 6$	1250	1090	790	790

Table 7.1 : Operations Count for Implementing Algorithms 7.1 and 7.2.

Now, for an efficient implementation of equations (7.3.2)-(7.3.4) we can make a number of interesting observations. For example, in implementing equation (7.3.2), since the matrix $\mathbf{D}^o$ is symmetric, we need to compute only its upper (or lower) triangular part. Also, some terms are common to all three matrices. Moreover, in most dot products one of the vectors involved, namely the vector $\mathbf{z}_i^i$, is a unit vector and thus we do not actually need to perform any computation for implementing such operations. In

particular, based on these observations, we can compute the diagonal elements of these matrices with almost no computational cost since we have $\mathbf{s}_{i,i}^{i} = \dot{\mathbf{s}}_{j,j}^{j} = 0$. Also, in computing the first sub-diagonal (or super-diagonal) of these matrices we can assume that terms such as $\tilde{\mathbf{s}}_{i,i+1}^{i+1}\tilde{\mathbf{U}}_{O_{i+1}}^{i+1}$, $\tilde{\mathbf{s}}_{i,i+1}^{i+1}\dot{\mathbf{U}}_{O_{i+1}}^{i+1}$ and $\tilde{\mathbf{U}}_{O_{i+1}}^{i+1}\tilde{\mathbf{s}}_{i,i+1}^{i+1}$ are known from Algorithm 7.2. Finally, equations (7.3.7), (7.3.8) and (7.3.10) may be taken into consideration for a more efficient implementation of equations (7.3.3) and (7.3.4). Based on these observations, an estimate of the number of scalar operations involved in the numerical implementation of equations (7.3.2), (7.3.3) and (7.3.4) is given in Table 7.2.

	Manipulator with general twist angles	
Equation	Multiplications	Additions
7.3.2	$9n^2 - 18n + 8$	$9n^2 - 12n + 3$
7.3.3	$15n^2 - 30n + 34$	$14n^2 - 15n + 25$
7.3.4	$16n^2 - 53n + 60$	$13n^2 - 43n + 46$
Total	$40n^2 - 101n + 102$	$36n^2 - 70n + 74$
$n = 6$	936	950

Table 7.2 : Operations Count for Implementing Equations (7.3.2), (7.3.3) and (7.3.4)

As we can see from Tables 7.1 and 7.2, we need approximately 2186 scalar multiplications and 2040 scalar additions to compute the joint space sensitivity coefficient matrices of a linearized robot model at a point of the nominal trajectory. This estimate is valid for a manipulator with all joints of revolute type and arbitrary twist angles. As is well known, this case is computationally the most expensive one since when prismatic joints are present some rotational transformations are not needed. For manipulators with simpler geometric structure, i.e., with twist angles 0 or 90 degrees, the computational cost for the same operations is reduced to 1726 scalar multiplications and 1740 scalar additions.

For the sake of comparison, we mention here that to compute the closed-form linearized robot model in joint space, with the procedure which is proposed in [15], will

require 7400 scalar operations. Also, in [8] it has reported that to generate, symbolically, the joint space linearized robot model will require approximately $\sqrt{n}$ times more scalar operations than the symbolic generation of the associated nonlinear dynamic robot model. However, the same authors in [17] have been estimated that this nonlinear dynamic robot model will require 267 (4×4) matrix-matrix multiplications and 196 trace operations and vector-matrix-vector multiplications (for a 6 degrees-of-freedom manipulator) which results in more than 15000 scalar multiplications. Therefore, the procedure which is proposed here for obtaining the coefficient sensitivity matrices of a joint space linearized dynamic robot model has significantly higher computationally efficiency when compared with other approaches available today.

7.4 Cartesian Space Dynamic Robot Models and Their Linearization

As is well known [18], it is possible to describe the dynamics of a robot manipulator by using other sets of variables besides joint space variables. These variables are known as *operational space variables*, and among them, the *Cartesian configuration space variables* or simply *Cartesian space variables* are probably the most important. For example, in many cases, such as for end-effector motion and force control it may be desirable to express the dynamics of a manipulator in terms of "external" variables for direct, and thus better, measurements. In these cases, Cartesian space variables are obviously appropriate. Briefly, dynamic robot models described in terms of Cartesian space variables -henceforth called *Cartesian space dynamic robot models* can be introduced as follows.

7.4.1 Cartesian Space Dynamic Robot Models

As we mentioned in Chapter II, for nonredundant manipulators, the Cartesian space variables χ are defined to be the independent configuration parameters which specify the position and orientation of the end-effector relative to the inertia coordinate

system. These Cartesian variables are functions of the joint space coordinates and this is usually expressed by a "geometric" equation of the form

$$\chi = \mathbf{h}(\mathbf{q}). \tag{7.4.1}$$

In general, the vector function $\mathbf{h}$ is not one-to-one. However, in a restricted domain of the joint space, $\mathbf{h}$ can be assumed to be one-to-one. In this case the Cartesian space dynamic model of a robot manipulator can be defined [1,18] by the following equation

$$\mathbf{f} = \mathbf{D}_\chi(\mathbf{q})\ddot{\chi} + \mathbf{C}_\chi(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}_\chi(\mathbf{q}) \tag{7.4.2}$$

where $\mathbf{f}$ is an $n \times 1$ force-torque vector acting on the end-effector of the robot, and χ is a Cartesian space vector which describes the position and the orientation of the end-effector. The other terms in equation (7.4.2) are defined as follows : $\mathbf{D}_\chi(\mathbf{q})$ is the $n \times n$ Cartesian space generalized inertia tensor, $\mathbf{C}_\chi(\mathbf{q}, \dot{\mathbf{q}})$ is the $n \times 1$ Cartesian space vector of centrifugal and Coriolis terms, and $\mathbf{G}_\chi(\mathbf{q})$ is the $n \times 1$ Cartesian space vector of gravity terms. Obviously, all these Cartesian terms are implicit functions of the joint space coordinates. Actually, it can be shown [18] that if a closed-form joint space dynamic robot model is given by the equation

$$\tau = \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) \tag{7.4.3}$$

where $\mathbf{D}(\mathbf{q})$ is the $n \times n$ joint space generalized inertia tensor of the manipulator, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the $n \times 1$ joint space vector of centrifugal and Coriolis terms, and $\mathbf{G}(\mathbf{q})$ is the $n \times 1$ joint space vector of gravity terms, then the aforementioned Cartesian space quantities are related to their joint space counterparts by the following equations :

$$\mathbf{D}_\chi(\mathbf{q}) = \mathbf{J}^{-T}(\mathbf{q})\mathbf{D}(\mathbf{q})\mathbf{J}^{-1}(\mathbf{q}) \tag{7.4.4a}$$

$$\mathbf{C}_\chi(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}^{-T}(\mathbf{q})\left[\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{D}(\mathbf{q})\mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}\right] \tag{7.4.4b}$$

$$\mathbf{G}_\chi(\mathbf{q}) = \mathbf{J}^{-T}(\mathbf{q})\mathbf{G}(\mathbf{q}) \tag{7.4.4c}$$

$$\mathbf{f} = \mathbf{J}^{-T}(\mathbf{q})\tau \tag{7.4.4d}$$

where $\mathbf{J}(\mathbf{q})$ is the manipulator Jacobian which has been assumed here to be nonsingular. When the Jacobian is locally singular, it is still possible [18] to define equation (7.4.2) by

considering the manipulator to be a redundant manipulator locally. However, since in this thesis we are dealing with nonredundant manipulators, we shall assume that the manipulator Jacobian is nonsingular.

Now, since in practical applications we cannot actually cause a Cartesian force to be applied to the end-effector of a manipulator, we use equation (7.4.4d) to transfer the Cartesian force vector $\mathbf{f}$ to an equivalent joint torque vector τ which effectively will cause the end-effector to follow the required motion. Therefore, instead of computing first the force vector $\mathbf{f}$ and then transferring it to τ, we may compute directly the joint torque vector τ. To achieve this, we combine equations (7.4.2) and (7.4.4d) and write the following *Cartesian configuration space torque equation.*

$$\tau = \hat{\mathbf{D}}_{\chi}(\mathbf{q})\ddot{\chi} + \hat{\mathbf{C}}_{\chi}(\mathbf{q}, \dot{\mathbf{q}}) + \hat{\mathbf{G}}_{\chi}(\mathbf{q}) \tag{7.4.5}$$

which defines directly the vector of the joint torques τ when the dynamics of a robot are expressed in terms of the Cartesian space variables χ.

From the foregoing, as we can see, it is possible to define the generalized force vector τ in terms of either joint or Cartesian space variables. Therefore, it may be required in practice (e.g., for Cartesian based control applications) to be able to define the perturbations $\delta\tau$ *of* τ *in* terms of perturbations of the Cartesian space variables, i.e., to be able to define *Cartesian (configuration) space linearized dynamic robot models.*

7.4.2 Cartesian Space Linearized Dynamic Robot Models

Direct application of the Taylor series expansion to the nonlinear equations of a robot manipulator written in terms of the Cartesian space variables, as in equations (7.4.2) or (7.4.5), is rather difficult because it involves implicit differentiation in terms of the joint space variables. To avoid this complex differentiation, we can follow a similar approach as that used to derive Cartesian space nonlinear dynamic models from the joint space ones. In particular, in this approach, to define Cartesian space linearized dynamic robot models we define first a joint space linearized dynamic robot model and

then, by expressing the joint space perturbations in terms of the Cartesian space perturbations, we manipulate algebraically the joint space linearized dynamic robot model to a Cartesian space one. Therefore, in order to derive a Cartesian space linearized dynamic robot model we shall assume that the joint space linearized dynamic robot model of equation (7.2.1) is available.

As we mentioned in Section 7.4.1, the end-effector Cartesian space coordinates χ of a nonredundant manipulator can be considered as (Cartesian space) generalized coordinates which are related to the joint space generalized coordinates $\mathbf{q}$ by the nonlinear equation (7.4.1). As is well known, the time derivatives of these two sets of generalized coordinates are related by the equation

$$\dot{\chi} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{7.4.6}$$

where $\mathbf{J}(\mathbf{q})$ is the manipulator Jacobian. For general operational spaces the manipulator Jacobian is defined by the equation

$$\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{h}(\mathbf{q})}{\partial \mathbf{q}} \tag{7.4.7}$$

where $\mathbf{h}$ is defined by equation (7.4.2). Unfortunately, this is not true for the Cartesian space variables since there is no 3×1 orientation vector whose derivative is the vector of the angular velocity. However, in the case of Cartesian space variables the manipulator Jacobian can be easily extracted from the equations which define the linear and angular velocity of the end-effector. Based on these equations, several methods for defining the manipulator Jacobian have been proposed in [19].

Equation (7.4.6) implies that infinitesimal Cartesian displacements or small perturbations of the end-effector Cartesian vector χ are related to the joint space perturbations $\delta\mathbf{q}$ by the equation

$$\delta\chi = \mathbf{J}(\mathbf{q})\delta\mathbf{q} \,. \tag{7.4.8}$$

Furthermore, by differentiating equation (7.4.8) we get

$$\delta\dot{\chi} = \dot{\mathbf{J}}(\mathbf{q})\delta\mathbf{q} + \mathbf{J}(\mathbf{q})\delta\dot{\mathbf{q}} \tag{7.4.9}$$

and

$$\delta\ddot{\chi} = \ddot{\mathbf{J}}(\mathbf{q})\delta\mathbf{q} + 2\dot{\mathbf{J}}(\mathbf{q})\delta\dot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\delta\ddot{\mathbf{q}}\,. \tag{7.4.10}$$

Now, since in the definition of the Cartesian space nonlinear dynamic equations we have assumed $\mathbf{J}(\mathbf{q})$ to be nonsingular, $\mathbf{J}^{-1}$ exists and therefore we can solve equations (7.4.8)-(7.4.10) for $\delta\mathbf{q}$, $\delta\dot{\mathbf{q}}$ and $\delta\ddot{\mathbf{q}}$ to get

$$\delta\mathbf{q} = \mathbf{J}^{-1}\delta\chi \tag{7.4.11}$$

$$\delta\dot{\mathbf{q}} = \mathbf{J}^{-1}\delta\dot{\chi} - \mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\delta\chi \tag{7.4.12}$$

$$\delta\ddot{\mathbf{q}} = \mathbf{J}^{-1}\delta\ddot{\chi} - 2\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\delta\dot{\chi} - \left[\mathbf{J}^{-1}\ddot{\mathbf{J}}\,\mathbf{J}^{-1} - 2\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\right]\delta\chi. \tag{7.4.13}$$

Expressions (7.4.11)-(7.4.13) can be used in (7.2.1) to yield

$$\begin{aligned}\delta\tau = &\left[\mathbf{D}^{o}\,\mathbf{J}^{-1}\right]\delta\ddot{\chi} + \left[\mathbf{V}^{o}\,\mathbf{J}^{-1} - 2\mathbf{D}^{o}\,\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\right]\delta\dot{\chi} \\ &+ \left[\mathbf{P}^{o}\,\mathbf{J}^{-1} - \mathbf{V}^{o}\,\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1} - \mathbf{D}^{o}\left(\mathbf{J}^{-1}\ddot{\mathbf{J}}\,\mathbf{J}^{-1} - 2\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\dot{\mathbf{J}}\,\mathbf{J}^{-1}\right)\right]\delta\chi.\end{aligned} \tag{7.4.14}$$

or, if we define

$$\hat{\mathbf{D}}_{\chi}^{o} = \mathbf{D}^{o}\,\mathbf{J}^{-1} \tag{7.4.15}$$

$$\hat{\mathbf{V}}_{\chi}^{o} = \left(\mathbf{V}^{o} - 2\hat{\mathbf{D}}_{\chi}^{o}\dot{\mathbf{J}}\right)\mathbf{J}^{-1} \tag{7.4.16}$$

and

$$\hat{\mathbf{P}}_{\chi}^{o} = \left(\mathbf{P}^{o} - \hat{\mathbf{V}}_{\chi}^{o}\dot{\mathbf{J}} - \hat{\mathbf{D}}_{\chi}^{o}\ddot{\mathbf{J}}\right)\mathbf{J}^{-1}, \tag{7.4.17}$$

equation (7.4.14) can be written in a compact form as

$$\delta\tau = \hat{\mathbf{D}}_{\chi}^{o}\delta\ddot{\chi} + \hat{\mathbf{V}}_{\chi}^{o}\delta\dot{\chi} + \hat{\mathbf{P}}_{\chi}^{o}\delta\chi. \tag{7.4.18}$$

Equation (7.4.18) defines the perturbation in the vector of joint torques τ as a result of perturbations in the vectors of Cartesian space positions, velocities and accelerations, i.e., it defines a *Cartesian space linearized dynamic robot model.* Moreover, by analogy with joint space linearized dynamic robot models, we may refer to the matrix coefficients of equations (7.4.18) as the *Cartesian space coefficient sensitivity matrices.*

Now, as we can see from equations (7.4.15)-(7.4.17), most of the quantities which are involved in the definitions of the Cartesian space coefficient sensitivity matrices $\hat{\mathbf{D}}_X^o$, $\hat{\mathbf{V}}_X^o$ and $\hat{\mathbf{P}}_X^o$ may be considered to be known, since they are available either from the joint space linearized dynamic robot model or from the Cartesian space nonlinear dynamic equations (e.g., see equation (7.4.4)). Therefore, the implementation of equation (7.4.18) requires only a few extra computations. Thus, following this approach, we can derive both joint and Cartesian space linearized dynamic robot models with almost the same computational cost.

7.5 Concluding Remarks

In this chapter, the linearization of the dynamic equations for rigid-link serial-type robot manipulators has been considered. Based on the Taylor series expansion and using Cartesian tensor analysis, we have proposed a new procedure for obtaining the elements of the joint space coefficient sensitivity matrices. Also, we have shown that this procedure leads to algorithms which can be implemented numerically more efficiently than other similar algorithms existing in the literature.

The problem of obtaining Cartesian space linearized dynamic robot models has also been addressed in this chapter and, to the best of our knowledge, this is the first time where Cartesian space linearized dynamic robot models have been considered. To simplify our analysis, we have assumed that the manipulator is operating in a region of the work space where the Jacobian is nonsingular and we have shown that in these singularity free Cartesian configuration spaces, linearized dynamic robot models can be readily obtained from the joint space ones and the manipulator Jacobian.

7.6 REFERENCES

[1] J. J. Craig, *Introduction to Robotics : Mechanics & Control*, Addison-Wesley, Reading. MA 1986.

[2] P. Misra, R. V. Patel, and C. A. Balafoutis, "Robust Control of Linearized Dynamic Robot Models", in *'Robot Manipulators: Modeling, Control and Education'*, M. Jamshidi, J.Y.S. Luh and M. Shahinpur, *Eds.*, 1986.

[3] P. Misra, R. V. Patel and C. A. Balafoutis, in "Robust Control of Robot Manipulators in Cartesian Space", *Proc. American Control Conf.*, pp. 1351-1356, Atlanta, Georgia, June 15-17, 1988.

[4] C. S. G. Lee and M. J. Chung, "An Adaptive Control Strategy for Mechanical Manipulators", in *Tutorial on Robotics*, C. S. G. Lee, R. C. Gonzales and K. S. Fu, *Eds.*, IEEE Computer Society, 1983.

[5] J. Y. S. Luh, M. W. Walker and R. P. Paul, "Resolved-Acceleration Control for Mechanical Manipulators", *J. Dyn. Sys., Meas., & Cont.*, Vol. 102, pp. 69-76, 1980.

[6] E. Freund, "Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators", *Int. J. Robotics Research*, pp. 65-78, Vol. 1, 1982.

[7] P. M. Frank, *Introduction to System Sensitivity Theory*, Academic press, New York, 1978.

[8] C. P. Neuman and J. J. Murray, "Linearization and Sensitivity Functions of Dynamic Robot Models", *IEEE Trans. Syst. Man and Cyber.*, SMC-14, pp. 805-818, 1984.

[9] J. J. Murray and C. P. Neuman, "Linearization and Sensitivity Models of the Newton-Euler Dynamic Robot Models", *J. Dyn. Sys. Meas. & Contr.*, Vol. 108, pp. 272-276, 1986.

[10] C. P. Neuman, and P. K. Khosla, "Identification of Robot Dynamics : An Application of Recursive Estimation," in *Adaptive and Learning Systems : Theory and Applications*, K. S. Narendra, Eds., Plenum Publishing Corporation, New York, 1986.

[11] R. Su, "On the Linear Equivalents of Nonlinear Systems" in *Systems and Control Letters*, Vol. 2, No. 1, pp. 48-52, 1982.

[12] L. R. Hunt, R. Su, and G. Meyer, "Global Transformations of Nonlinear Systems", in *IEEE Trans. on Automatic Control*, Vol. AC-28, No. 1, 1983.

[13] Y. Chen, *Nonlinear Feedback and Computer Control of Robot Arms*, Ph. D. Thesis, Washington University, St. Louis, MO, 1984.

[14] M. Vukobratovic, and N. Kircanski, *Scientific Fundamentals of Robotics 4 : Real-Time Dynamics of Manipulation Robots*, Springer-Verlag, Berlin, 1985.

[15] C. A. Balafoutis, P. Misra, and R. V. Patel, "Recursive Evaluation of Linearized Dynamic Robot Models", in *IEEE J. Robotics and Automation*, RA-2, pp. 146-155, 1986.

[16] C. A. Balafoutis and R. V. Patel, "Linearized Robot Models in Joint and Cartesian Spaces", in *Proc. 9th Symposium on Engineering Applications of Mechanics : Current and Emerging Technologies*, London, Ontario, May 29-31, pp. 587-594, 1988.

[17] J. J. Murray and C. P. Neuman, "ARM : An Algebraic Robot Dynamic Modeling Program", in *Proc. IEEE Conf. on Robotics*, pp. 103-114, Atlanta, CA, Mar. 13-15, 1984.

[18] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators : The Operational Space Formulation", in *IEEE Journal of Robotics and Automation*, Vol. 3, No. 1, pp. 43-53, 1987.

[19] D. E. Orin and W. W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators", in *Int. J. Robotics Research,* Vol. 3, pp. 66-75, 1984.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

This thesis does not represent the start of a new field nor the culmination of an existing one. It is a contribution to the significant effort that has been made and is being made by a number of researchers to make flexible automation an every day reality. Our goal in this thesis has been to derive easy-to-use and computationally efficient algorithms for solving some basic problems of robot dynamics. In particular, we have considered the following problems of rigid-link open-chain manipulator dynamics :

i) the problem of inverse dynamics,

ii) the problem of forward dynamics, and

iii) the linearization of the equations of motion for the above mentioned class of manipulators.

These problems are chosen from practical considerations since, as we have discussed in this thesis, their solution in a computationally efficient manner is a prerequisite for real-time robotic applications, which in turn is necessary for flexible automation in a dynamically changing environment. The main drawback in solving these problems with many of the existing efficient algorithms in the robotics literature is that they are based on customization which restricts their applicability to manipulators with specific geometry. The main feature of the algorithms presented in this thesis is that they are computationally very efficient without requiring customization, and therefore they can be applied to almost all general purpose industrial robots.

The computational efficiency of these algorithms has been demonstrated through comparison with other algorithms in the literature and has been achieved mainly

through a more efficient formulation of the dynamic equations of motion. A deeper understanding of the mathematical representations used to describe basic physical quantities of motion, has guided us in developing a new methodology for the analysis of the dynamic equations of rigid body motion and has resulted in an efficient formulation of these equations. Also, the use of generalized and augmented links has enabled us to devise underlying modeling schemes which are very much suited for the dynamic analysis of rigid-link open-chain robot manipulators, since they allow us to compute as many as possible manipulator's configuration independent dynamic parameters off-line. Thus, by using Cartesian tensor analysis and the ideas of generalized and augmented links, we have proposed in this thesis algorithms which are computationally the most efficient non-customized algorithms presently available for solving the three problems mentioned above.

However, during the course of this research, contributions have been made not only in the field of robotics (which was our main goal) but also in the fields of Cartesian tensor analysis and multi-body dynamics. Briefly the contributions that this thesis has made in the latter fields are as follows :

In Cartesian tensor analysis, by exploring the relationships between second order Cartesian tensors and their vector invariants, new simple coordinate-free proofs are given for some well known tensor-vector equations and a number of new propositions have been stated and proved. These propositions define important tensor identities which allow us to manipulate second order Cartesian tensors very efficiently as abstract objects without the need to resort to coordinate bases. Also, a geometric interpretation for the second order skew-symmetric Cartesian tensors has been given which allows for an easy description of (relatively) oriented plane areas and a simple mathematical analysis of the linear space of planes in three dimensional Euclidean spaces. This geometric characterization and analysis of skew-symmetric tensors and planes gives a concrete physical interpretation and a deeper insight into tensor algebraic equations

which may result in more efficient solutions to practical problems.

Based on this new understanding of Cartesian tensor analysis, a new methodology for studying classical rigid body dynamics has been proposed in this thesis which is conceptually simple, easy to implement, and computationally efficient. In particular, by introducing the angular acceleration tensor, a new tensor formulation has been given to one of the basic equations of rigid body motion, namely, the Euler equation. This tensor formulation of Euler's equation, retains the simple structure of its classical vector formulation but is computationally far more efficient. Moreover, it has been demonstrated that the use of the angular acceleration tensor also simplifies the computation for the linear acceleration of various position vectors which are important in studying rigid body motion. Thus, based on this new tensor methodology we have provided a new approach to the study of classical Newtonian mechanics.

8.2 FUTURE WORK

It is in the nature of research that the solution of one problem often gives rise to many new questions or problems. In the case of the research which is presented in this thesis, the following questions surface naturally.

i) *Parallel computations :* The execution of the proposed algorithms is assumed to be done in a sequential fashion using a single processor. Therefore, in order to reduce further the time required to carry out the computations involved, it would be interesting to examine possible scheduling schemes in which many of the 'primitive' operations (matrix/vector arithmetic operations) can be executed in parallel using a number of processors. Moreover, it would be useful to develop schemes for implementing these computations on a commercially available parallel computer architecture such as the Hypercube [1-2].

ii) *Kinematic and dynamic analysis of closed-chain mechanisms (e.g., manipulators, dextrous hands, multirobots, etc.) and flexible link manipulators* . As we have mentioned

in this thesis, the articulated part in most industrial robots is an open-chain manipulator with rigid links. However, in recent years some effort has been directed towards alternative manipulator designs which require the study of closed kinematic chain mechanisms or structural flexibility. Closed kinematic chain rigid-link manipulators have potential applications where the demand on workspace and maneuverability is low but the dynamic loading is severe and high speed and high precision motions are of primary concern. On the other hand, the use of large-scale light-weight manipulators, with inherent structural flexibility is very attractive in space applications where the size of the workspace and the amount of power which is required to operate the robot are critical factors.

The kinematic and dynamic analysis of rigid-link robot manipulators with closed-chain geometry or those with flexible links is more difficult than that of open-chain rigid-link robot manipulators and has not been as extensively studied. For example, present formulations for the dynamic equations of motion of robot manipulators with closed kinematic chains pertain to the particular method (Newton-Euler, Lagrange, Kane's, etc.) which is used to derive them and, in general, different methods result in different forms of the equations of motion. Also, it is not clear if recursive formulations, like those proposed for open-chain manipulators, can be found for manipulators with closed-chain geometry. These and other problems in the kinematic and dynamic analysis of closed-chain manipulators may benefit from the physical insight that dynamic analysis of rigid body motion based on Cartesian tensor analysis can provide. Also, techniques from rigid body dynamic analysis can be used for the dynamic analysis of a flexible link manipulator since the latter can be modeled, for example, as a chain of rigid sublinks interconnected by elastic joints.

iii) Manipulator control : This is a very active field of research in robotics and many problems in position and/or force control have yet to be solved with adequate degree of robustness so that real-time flexible automation can be achieved. Towards this goal, the

computational algorithms which solve the problem of inverse dynamics, presented in this thesis, provide new efficient ways for implementing "computed-torque" type control methods. In particular, since the computational cost of these inverse dynamics algorithms is relatively small, it may be possible now to design real-time manipulator controllers based on more advanced techniques of linear control theory (e.g., robust servomechanism techniques [3] and robust control [4]) instead of using simple PD or PID controllers which are usually employed with computed-torque control methods. If such robust controllers can be implemented in real-time, then the computed-torque based controllers, may become effective means of controlling rigid link robotic manipulators. Also computed-torque based controllers augmented with simple adaptive controllers (as proposed by Craig et al. [5]) which will compensate for unmodeled dynamics or varying pay-loads may then also provide an efficient and economic solution to manipulator control problems. Finally, the joint or Cartesian space linearized robot models which have been proposed in this thesis provide us with an alternative way of using linear control theory in designing simple manipulator controllers.

iv) Geometric characterization of rotation tensors and Cartesian tensor analysis : The problem of describing uniquely, with the minimum possible number of independent variables, a rotation tensor defined in a 3-D Euclidean space has been of interest since the middle of the 18-th century, when Euler first showed that a rotation tensor possesses three degrees-of-freedom. The extensive literature on the subject indicates the importance of this problem. Over the years, a number of solutions have been proposed for this problem. Among them the most familiar are those which are based on concepts involving *a set of three linearly independent scalar variables* (usually three successive angles), *quarternions, special unitary* 2×2 *and* 3×3 *matrices, Pauli spin matrices, etc.* Although sufficient for practical purposes, all these solutions are not free from limitations or other problems. For example, solutions which are based on three linearly independent scalar variables are not free from singularities while the so-called "quarternion methods" are

1-2 ways descriptions. On the other hand, it is well known [6] that topologically it is impossible to have a global 3-dimensional parametrization for the rotation group without singular points and that five is the minimum number of scalar parameters which suffices to represent the rotation group in a 1-1 global manner. At first glance, this appears to be a paradox since, as we have mentioned, a rotation tensor possesses three degrees-of-freedom, so that it is natural to expect that three linearly independent variables can be associated with them globally.

Examination of all existing descriptions of a rotation tensor reveals that to a degree-of-freedom of a rotation tensor, we associate a 1-dimensional scalar variable (i.e., a real number). Thus, for example, when we describe a rotation tensor by using three successive angles, conventionally we consider an angle to be described by a (1-dimensional scalar) variable which measures the length of an arc and is associated with one direction, the axis of rotation. However, an angle is more than a scalar. Actually, an angle expresses a relation between two directions and this implies that an angle is associated with two directions, i.e., an angle is associated with a plane area. Therefore, the "correct" description of an angle is not by a 1-dimensional variable but by a 2-dimensional variable which defines a plane area. This approach of describing the degrees-of-freedom of a rotation tensor is also supported from the following considerations :

First, as is well known [7], an orthogonal tensor $\mathbf{R}$ defined in the n-dimensional Euclidean space can be written in the form

$$\mathbf{R} = (1 + \mathbf{N})(1 - \mathbf{N})^{-1}$$

where $\mathbf{N}$ is a skew-symmetric tensor. Now, since a skew-symmetric tensor defined in an n-dimensional (oriented) Euclidean space has $\frac{1}{2}n(n-1)$ linearly independent variables it follows that the tensor $\mathbf{R}$ can be described in terms of $\frac{1}{2}n(n-1)$ linearly independent variables. On the other hand, in an n-dimensional (oriented) Euclidean space the

number of ways of choosing 2 different directions (i.e., oriented plane areas or oriented angles) out of n is given by

$$\binom{n}{2} = \frac{n!}{2!(n-2)!}$$
$$= \frac{1}{2}n(n-1)$$

which is equal to the number of linearly independent variables which an orthogonal tensor (and thus a rotation tensor) possesses. Secondly it can be shown [8] that we can define the components of the angular velocity vector by the following integrals

$$\omega_i = \frac{1}{4\pi}\int \dot{\theta}_i \, d\sigma \qquad (i = 1, 2, 3)$$

where, $d\sigma$ is the area of a differential element and θ_i are angles introduced in a certain manner (see [8]). This fact, and the close relationship which exists between a rotation tensor and its angular velocity tensor indicates that the degrees-of-freedom of a rotation tensor have a close relationship with plane areas.

From these considerations, it seems that the primitive and thus "proper" description of a degree-of-freedom of a rotation tensor is provided by a 2-dimensional variable which may define a plane area. If this is true, it implies that we may be able to describe a rotation tensor by three linearly independent variables in an 1-1 global manner by using three linearly independent planes or three linearly independent 2-dimensional variables. To see if this is possible and how it can be done, we need to have a deeper understanding of Cartesian tensor analysis in general and, in particular, we need to be able to work effectively in the linear space of planes which has been introduced in Chapter III of this thesis.

However, besides applications to this "unconventional" description of rotation tensors, Cartesian tensor analysis can be used in other ways too. For example, as we have shown in Chapter III of this thesis, Cartesian tensor analysis provides a geometric explanation of why the transpose of a rotation tensor is equal to its inverse. Furthermore,

relatively oriented skew-symmetric tensors provide a mathematical representation of expressions in the nature language such as "right-handed" or "left-handed" rotations. Moreover, Cartesian tensor analysis enables us to extract the correct angle θ, for $0 \leq \theta \leq 2\pi$, from a rotation tensor when its axis of rotation is known. This follows from the fact that for a rotation $\mathbf{R}$, with axis of rotation $\mathbf{r}$, its angle θ satisfies the following equations

$$\cos(\theta) = \frac{1}{2}\left(tr[\mathbf{R}] - 1\right)$$

and

$$\sin(\theta) = \frac{1}{2\|\mathbf{r}\|}\mathbf{R} : \tilde{\mathbf{r}}^{T}$$

the derivation of which is simple and will be reported elsewhere [9]. Therefore, Cartesian tensor analysis is shown to be a powerful mathematical language for the analysis of rotation tensors. This may help us to derive unique solutions for problems where, by traditional methods, we end-up with multiple solutions (e.g., inverse manipulator kinematics).

8.3 REFERENCES

[1] A. J. Laub and J. D. Gardiner, "Hypercube Implementation of Some Parallel Algorithms in Control", in *Advanced Computing Concepts and Techniques in Control Engineering*, Edited by M. J. Denham and A. J. Laub, Springer-Verlag, NY, 1987.

[2] O. A. McBryan and E. F. Van de Velde, "Hypercube Algorithms and Implementations", *SIAM J. Sci. Stat. Comput.*, Vol. 8, pp. 227-287, 1987.

[3] R. V. Patel and N. Munro, *Multivariable System Theory and Design*, Pergamon Press, Oxford, 1982.

[4] M. W. Spong and M. Vidyasagar, "Robust Linear Compensator Design for Nonlinear Robotic Control", *IEEE J. Robotics and Automation*, Vol. RA-3, No. 4, pp. 345-351, 1987.

[5] J. J. Craig, P. Hsu and S. S. Sastry, "Adaptive Control of Mechanical Manipulators", *Int. J. Robotics Research*, Vol. 6, No. 2, pp. 16-28, 1987.

[6] J. Stuelpnagel, "On the Parametrization of the Three-Dimensional Rotation Group", *SIAM REVIEW*, pp. 422-430, Vol. 6, No. 4, 1964.

[7] O. Bottema and B. Roth, *Theoretical Kinematics*, North-Holland Publishing Co., Amsterdam, 1978.

[8] T. R. Kane, P. W. Likins, and D. A. Levinson, *Spacecraft Dynamics*, McGraw-Hill, New York, 1983.

[9] C. A. Balafoutis, and R. V. Patel, "On the Orientation and Representation of Spatial Rotations About a Fixed Point", in preparation.

APPENDIX A

RECURSIVE LAGRANGIAN FORMULATION

In a dynamical system, the Lagrangian $\mathbf{L}$ is defined as the difference between the total kinetic energy of the system, $\mathbf{K}$, and the total potential energy of the system, $\mathbf{H}$, assuming no dissipation of energy, that is,

$$\mathbf{L} = \mathbf{K} - \mathbf{H} \tag{A.1}$$

and the corresponding Euler-Lagrange equations are written as

$$\tau_i = \frac{d}{dt}\left(\frac{\partial \mathrm{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathrm{L}}{\partial q_i}, \qquad i = 1, \cdots, n \ , \tag{A.2}$$

where τ_i are the generalized forces, q_i are the generalized coordinates, and $\dot{q}_i$ is the generalized velocity.

Now, following an analysis similar to that used by Hollerbach in [14], we can show that the total kinetic energy of the manipulator is given by the equation

$$\mathbf{K} = \frac{1}{2}\sum_{j=1}^{n} tr\left\{ m_j \dot{\mathbf{s}}_{0,j}\dot{\mathbf{s}}_{0,j}^{T} + 2\dot{\mathbf{W}}_j \mathbf{n}_j^j \dot{\mathbf{s}}_{0,j}^{T} + \dot{\mathbf{W}}_j \mathbf{J}_{\mathrm{o}_j}^j \dot{\mathbf{W}}_j^{T} \right\} \tag{A.3}$$

where m_j is the mass of the j-th link, $\dot{\mathbf{s}}_{0,j}$ is the absolute velocity of the origin o_j of the j-th link coordinate system, $\mathbf{n}_j^j = m_j \mathbf{r}_{j,j}^j$ is the first moment of the j-th link about o_j expressed in link coordinate system orientation, $\dot{\mathbf{W}}_j$ is the absolute derivative of the orientation tensor of the j-th link coordinate system and $\mathbf{J}_{\mathrm{o}_j}^j$ is the pseudo-inertia tensor of the j-th link about o_j, expressed in link coordinate system orientation.

The total potential energy $\mathbf{H}$ is equal to the sum of the work required to transport the mass center of each link from a reference plane, i.e.,

$$\begin{aligned} \mathbf{H} &= \text{constant} - \sum_{j=1}^{n} m_j \mathbf{g}^{T} \mathbf{r}_{0,j} \\ &= \text{constant} - \sum_{j=1}^{n} m_j \mathbf{g}^{T} \left(\mathbf{r}_{0,j} + \mathbf{W}_j \mathbf{r}_{j,j}^j \right), \end{aligned} \tag{A.4}$$

where **g** is the acceleration due to gravity, with reference to the base coordinate system. This form for the potential energy is different from equation (A6) in Ref. [14] and leads to the modified analysis which is presented in this Appendix. Since the potential energy is position dependent only, equation (A.2) can be written as

$$\tau_i = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i}\right) - \frac{\partial K}{\partial q_i} + \frac{\partial H}{\partial q_i}, \qquad i = 1, \cdots, n . \tag{A.5}$$

Moreover, as in [14], we can write

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_i}\right) - \frac{\partial K}{\partial q_i} = \sum_{j=1}^{n} tr\left[m_j \frac{\partial s_{0,j}}{\partial q_i}\ddot{s}_{0,j}^T + \frac{\partial s_{0,j}}{\partial q_i}(n_j^j)^T \ddot{W}_j^T + \frac{\partial W_j}{\partial q_i} n_j^j \ddot{s}_{0,j}^T + \frac{\partial W_j}{\partial q_i} J_{0_j}^j \ddot{W}_j^T\right]. \tag{A.6}$$

For the partial derivative of the potential energy we have from equation (A.4)

$$\frac{\partial H}{\partial q_i} = -\sum_{j=1}^{n} m_j g^T \frac{\partial r_{0,j}}{\partial q_i} = -\sum_{j=1}^{n} m_j g^T \left(\frac{\partial s_{0,j}}{\partial q_i} + \frac{\partial W_j}{\partial q_i} r_{j,j}^j\right). \tag{A.7}$$

Moreover, since for $j \geq i$ we have

$$\frac{\partial s_{0,j}}{\partial q_i} = \frac{\partial W_i}{\partial q_i} s_{i,j}^i \tag{A.8}$$

and

$$\frac{\partial W_j}{\partial q_i} = \frac{\partial W_i}{\partial q_i} {}^iW_j , \tag{A.9}$$

equation (A.7) can be written as

$$\frac{\partial H}{\partial q_i} = -g^T \frac{\partial W_i}{\partial q_i} \sum_{j=1}^{n} m_j (s_{i,j}^i + {}^iW_j r_{j,j}^j) = -g^T \frac{\partial W_i}{\partial q_i} \sum_{j=1}^{n} m_j r_{i,j}^i . \tag{A.10}$$

Now, using (A.6) and (A.10) we can write equation (A.5) as follows :

$$\tau_i = \sum_{j=1}^{n} \left\{ tr\left[m_j \frac{\partial s_{0,j}}{\partial q_i}\ddot{s}_{0,j}^T + \frac{\partial s_{0,j}}{\partial q_i}(n_j^j)^T \ddot{W}_j^T + \frac{\partial W_j}{\partial q_i} n_j^j \ddot{s}_{0,j}^T \right.\right.$$

$$+ \frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \Big] - m_j \mathbf{g}^T \frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{r}_{i,j}^i \Big\}, \qquad i = 1, \cdots, n \ . \quad \text{(A.11)}$$

Moreover, if we use equations (A.8) and (A.9), we can simplify equation (A.11) as follows

$$\tau_i = tr\Big\{ \frac{\partial \mathbf{W}_i}{\partial q_i} \sum_{j=1}^{n} \Big[m_j \mathbf{s}_{i,j}^i \ddot{\mathbf{s}}_{0,j}^T + \mathbf{s}_{i,j}^i (\mathbf{n}_j^j)^T \ddot{\mathbf{W}}_j^T + {}^i\mathbf{W}_j \mathbf{n}_j^j \ddot{\mathbf{s}}_{0,j}^T + {}^i\mathbf{W}_j \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \Big] \Big\} - \mathbf{g}^T \frac{\partial \mathbf{W}_i}{\partial q_i} \sum_{j=1}^{n} m_j \mathbf{r}_{i,j}^i, \qquad i = 1, \cdots, n \ . \quad \text{(A.12)}$$

Now, following Hollerbach's approach [14], the first summation term on the right-hand side of (A.12) can be computed by the recurrence relations

$$\begin{aligned} \mathbf{D}_i &= \sum_{j=1}^{n} \Big[m_j \mathbf{s}_{i,j}^i \ddot{\mathbf{s}}_{0,j}^T + \mathbf{s}_{i,j}^i (\mathbf{n}_j^j)^T \ddot{\mathbf{W}}_j^T + {}^i\mathbf{W}_j \mathbf{n}_j^j \ddot{\mathbf{s}}_{0,j}^T + {}^i\mathbf{W}_j \mathbf{J}_{O_j}^j \ddot{\mathbf{W}}_j^T \Big] \\ &= \mathbf{A}_{i+1} \mathbf{D}_{i+1} + \mathbf{s}_{i,i+1}^i \mathbf{e}_{i+1} + \mathbf{n}_i^i \ddot{\mathbf{s}}_{0,i}^T + \mathbf{J}_{O_i}^i \ddot{\mathbf{W}}_i^T \end{aligned} \quad \text{(A.13)}$$

where

$$\begin{aligned} \mathbf{e}_i &= \sum_{j=1}^{n} \Big[m_j \ddot{\mathbf{s}}_{0,j}^T + (\mathbf{n}_j^j)^T \ddot{\mathbf{W}}_j^T \Big] \\ &= m_i \ddot{\mathbf{s}}_{0,i}^T + (\mathbf{n}_i^i)^T \ddot{\mathbf{W}}_i^T + \mathbf{e}_{i+1}. \end{aligned} \quad \text{(A.14)}$$

Similarly, the second summation term can also be computed by a recurrence relation. However, the recurrence relation presented here is different from the equation (13) derived by Hollerbach in Ref. [14]. We proceed as follows :

$$\begin{aligned} \mathbf{c}_i^i &= \sum_{j=1}^{n} m_j \mathbf{r}_{i,j}^i \\ &= m_i \mathbf{r}_{i,i}^i + \sum_{j=1+1}^{n} m_j \left(\mathbf{s}_{i,i+1}^i + \mathbf{A}_{i+1} \mathbf{r}_{i+1,j}^{i+1} \right) \\ &= m_i \mathbf{r}_{i,i}^i + \mathbf{s}_{i,i+1}^i \sum_{j=1+1}^{n} m_j + \mathbf{A}_{i+1} \sum_{j=1+1}^{n} m_j \mathbf{r}_{i+1,j}^{i+1} \\ &= m_i \mathbf{r}_{i,i}^i + \overline{m}_{i+1} \mathbf{s}_{i,i+1}^i + \mathbf{A}_{i+1} \mathbf{c}_{i+1}^{i+1} \end{aligned} \quad \text{(A.15)}$$

Substituting equations (A.13) and (A.15) into equation (A.12) we finally get Hollerbach's recurrence equation

$$\tau_i = tr\left(\frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{D}_i \right) - \mathbf{g}^T \frac{\partial \mathbf{W}_i}{\partial q_i} \mathbf{c}_i^i, \qquad i = 1, \cdots, n \ . \quad \text{(A.16)}$$

APPENDIX B

ON THE CONTRIBUTION OF MOMENT VECTORS TO GENERALIZED FORCES

As is well known, due to the rotational motion of a manipulator link, say the j-th one, forces and moments will be developed at all joints i for $i \leq j$. In this appendix we analyze the contribution of these moment vectors on the generalized force vector τ.

From the Newton-Euler formulation of the equations of motion of a robot manipulator it is known that when the j-th joint is of revolute type, then as a result of the rotation of the j-th link, a moment vector $\mathbf{M}_{o_j}$ is developed with respect to the center of rotation which is defined by the equation

$$\mathbf{M}_{o_j} = \mathbf{I}_{o_j} \cdot \dot{\omega}_j + \bar{\omega}_j \mathbf{I}_{o_j} \cdot \omega_j \tag{B.1}$$

where $\mathbf{I}_{o_j}$ is the inertia tensor of the j-th link with respect to the origin o_j, expressed in base frame orientation, and ω_j ($\dot{\omega}_j$) is the angular velocity (acceleration) of the j-th link. The contribution of this vector on the j-th component of the generalized force vector is denoted and defined by the equation

$$\hat{\tau}_j = \mathbf{z}_j \cdot \mathbf{M}_{o_j} \tag{B.2}$$

where $\mathbf{z}_j$ is the unit vector which is parallel to the j-th axis of rotation. Moreover, from the structure of the recurrence equation (5.3.8b) of Algorithm 5.4 (or equation (5.3.46b) of Algorithm 5.5), the moment vector $\mathbf{M}_{o_j}$ will also contribute to all components τ_i, for $i \leq j$, of the generalized force vector τ and this contribution is given by

$$\hat{\tau}_i = \mathbf{z}_i \cdot \mathbf{M}_{o_j} \tag{B.3}$$

From the foregoing, in the Newton-Euler formulation, the contribution of the moment vector $\mathbf{M}_{o_j}$ on the generalized force vector τ is explicitly defined by equations (B.2) and (B.3). However, this is not explicit or obvious, in the Lagrangian formulation of the dynamic equations of motion of a robot manipulator. In the following we shall

show that

$$\mathbf{M}_{o_j} \cdot \mathbf{z}_i = tr\left[\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}_{o_j}^{j}\ddot{\mathbf{W}}_j^T\right] \tag{B.4}$$

where $\mathbf{W}_j$ is the rotation tensor which specifies the orientation of the j-th link with reference to the base frame and $\mathbf{J}_{o_j}^{j}$ is the pseudo-inertia tensor of the j-th link with reference to o_j. Equation (B.4) is important since, by using the physical interpretation of the left-hand side of equation (B.4), we can gain a valuable insight into the structure of the Lagrangian formulation. To derive equation (B.4) we proceed as follows.

First we notice that, from equation (4.2.11), the rotation tensor $\mathbf{W}_j$ satisfies the equation

$$\begin{aligned}\ddot{\mathbf{W}}_j &= \boldsymbol{\Omega}_j \mathbf{W}_j \\ &= \left(\tilde{\dot{\omega}}_j + \tilde{\omega}_j\tilde{\omega}_j\right)\mathbf{W}_j .\end{aligned} \tag{B.5}$$

Moreover, for the partial derivative of the rotation tensor $\mathbf{W}_j$, we can write

$$\frac{\partial \mathbf{W}_j}{\partial q_i} = \tilde{\mathbf{z}}_i \mathbf{W}_j \tag{B.6}$$

Now, using equations (B.5) and (B.6) we can manipulate the right-hand side of equation (B.4) as follows

$$\begin{aligned} tr\left[\frac{\partial \mathbf{W}_j}{\partial q_i}\mathbf{J}_{o_j}^{j}\ddot{\mathbf{W}}_j^T\right] &= tr\left[\ddot{\mathbf{W}}_j \mathbf{J}_{o_j}^{j}\frac{\partial \mathbf{W}_j^T}{\partial q_i}\right] \\ &= tr\left[\left(\tilde{\dot{\omega}}_j + \tilde{\omega}_j\tilde{\omega}_j\right)\mathbf{W}_j \mathbf{J}_{o_j}^{j}\mathbf{W}_j^T\tilde{\mathbf{z}}_i^T\right] \\ &= tr\left[\tilde{\dot{\omega}}_j \mathbf{J}_{o_j}\tilde{\mathbf{z}}_i^T\right] + tr\left[\tilde{\omega}_j\tilde{\omega}_j \mathbf{J}_{o_j}\tilde{\mathbf{z}}_i^T\right]. \end{aligned} \tag{B.7}$$

Further, since the inertia and the pseudo-inertia tensors of a link satisfy equation (4.3.6), i.e., they satisfy the equation

$$\mathbf{J}_{o_j} = -\mathbf{I}_{o_j} + \frac{1}{2}tr\,[\mathbf{I}_{o_j}]\mathbf{1}$$

we can use Proposition 3.14 of Chapter III to write

$$tr\left[\tilde{\dot{\omega}}_j \mathbf{J}_{o_j} \tilde{\mathbf{z}}_i^T\right] = \dot{\omega}_j \cdot \mathbf{I}_{o_j} \cdot \mathbf{z}_i \tag{B.8}$$

Also, since from equation (4.3.19), we can write

$$\tilde{\omega}_j \mathbf{J}_{o_j} = \dot{\mathbf{J}}_{o_j} + \mathbf{J}_{o_j} \tilde{\omega}_j,$$

the second term on the right-hand side becomes

$$tr\left[\tilde{\omega}_j \tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\mathbf{z}}_i^T\right] = tr\left[\tilde{\omega}_j \dot{\mathbf{J}}_{o_j} \tilde{\mathbf{z}}_i^T\right] + tr\left[\tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\omega}_j \tilde{\mathbf{z}}_i^T\right].$$

But, from Proposition 3.12, the tensor $\tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\omega}_j$ is a symmetric tensor and therefore, by Proposition 3.13 we have

$$tr\left[\tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\omega}_j \tilde{\mathbf{z}}_i^T\right] = 0.$$

Thus, we can write

$$tr\left[\tilde{\omega}_j \tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\mathbf{z}}_i^T\right] = tr\left[\tilde{\omega}_j \dot{\mathbf{J}}_{o_j} \tilde{\mathbf{z}}_i^T\right]$$

which, on using Proposition 3.14, may be simplified to yield

$$tr\left[\tilde{\omega}_j \tilde{\omega}_j \mathbf{J}_{o_j} \tilde{\mathbf{z}}_i^T\right] = \omega_j \cdot \dot{\mathbf{I}}_{o_j} \cdot \mathbf{z}_i. \tag{B.9}$$

Therefore, by substituting equations (B.8) and (B.9) into (B.7), we get

$$tr\left[\frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{o_j}^j \ddot{\mathbf{W}}_j^T\right] = \left[\dot{\omega}_j \cdot \mathbf{I}_{o_j} + \omega_j \cdot \dot{\mathbf{I}}_{o_j}\right] \cdot \mathbf{z}_i \tag{B.10}$$

Moreover, since $\mathbf{I}_{o_j}$ and $\dot{\mathbf{I}}_{o_j}$ are symmetric we can write equation (B.10) as follows

$$\begin{aligned} tr\left[\frac{\partial \mathbf{W}_j}{\partial q_i} \mathbf{J}_{o_j}^j \ddot{\mathbf{W}}_j^T\right] &= \left[\mathbf{I}_{o_j} \cdot \dot{\omega}_j + \dot{\mathbf{I}}_{o_j} \cdot \omega_j\right] \cdot \mathbf{z}_i \\ &= \left[\mathbf{I}_{o_j} \cdot \dot{\omega}_j + \left(\tilde{\omega}_j \mathbf{I}_{o_j} - \mathbf{I}_{o_j} \tilde{\omega}_j\right) \cdot \omega_j\right] \cdot \mathbf{z}_i \quad \text{(by (4.3.17))} \\ &= \left[\mathbf{I}_{o_j} \cdot \dot{\omega}_j + \tilde{\omega}_j \mathbf{I}_{o_j} \cdot \omega_j\right] \cdot \mathbf{z}_i \qquad \text{(by (3.4.7))} \\ &= \mathbf{M}_{o_j} \cdot \mathbf{z}_i. \end{aligned}$$

which is equation (B.4).

APPENDIX C

ON PARTIAL DIFFERENTIATION

To derive the Jacobians which define the joint space coefficient sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$ of a linearized robot model, we need to compute the partial derivatives for a number of tensor and vector functions involved in the definition of the non-linear robot model. In this appendix, the partial derivatives of tensor and vector functions appearing in Algorithm 5.5 are defined and some important lemmas are proved.

First to compute the partial derivatives of the angular velocity and angular acceleration tensors $\tilde{\omega}_i^j$ and Ω_i^j, we can use either the recursive equations (5.3.45a)-(5.3.45c) or the following equations

$$\tilde{\omega}_i^j = \mathbf{W}_i^T \dot{\mathbf{W}}_i \tag{C.1}$$

and

$$\Omega_i^j = \mathbf{W}_i^T \ddot{\mathbf{W}}_i \tag{C.2}$$

which define these tensors (see Chapter IV) in terms of the orientation (transformation) tensor $\mathbf{W}_i = \mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_i$. Here we shall use equations (C.1) and (C.2). Before we proceed to define these partial derivatives we need to state the following simple facts. First, for $j \leq i$, the following equations are obviously true :

$$\mathbf{W}_i = \mathbf{W}_j {}^j\mathbf{W}_i \tag{C.3}$$

$$\dot{\mathbf{W}}_i = \dot{\mathbf{W}}_j {}^j\mathbf{W}_i + \mathbf{W}_j {}^j\dot{\mathbf{W}}_i \tag{C.4}$$

$$\ddot{\mathbf{W}}_i = \ddot{\mathbf{W}}_j {}^j\mathbf{W}_i + 2\dot{\mathbf{W}}_j {}^j\dot{\mathbf{W}}_i + \mathbf{W}_j {}^j\ddot{\mathbf{W}}_i \tag{C.5}$$

where ${}^j\mathbf{W}_i = \mathbf{A}_{j+1}\mathbf{A}_{j+2} \cdots \mathbf{A}_i$. Furthermore, from (C.1), (C.3) and (C.4) we have

$$\begin{aligned}
\tilde{\omega}_i^j &= \mathbf{W}_i^T(\dot{\mathbf{W}}_j {}^j\mathbf{W}_i + \mathbf{W}_j {}^j\dot{\mathbf{W}}_i) \\
&= {}^j\mathbf{W}_i^T(\mathbf{W}_j^T\dot{\mathbf{W}}_j){}^j\mathbf{W}_i + {}^j\mathbf{W}_i^T {}^j\dot{\mathbf{W}}_i \\
&= {}^j\mathbf{W}_i^T\tilde{\omega}_j {}^j\mathbf{W}_i + {}^j\mathbf{W}_i^T {}^j\dot{\mathbf{W}}_i \\
&= \tilde{\omega}_j^i + {}^j\mathbf{W}_i^T {}^j\dot{\mathbf{W}}_i
\end{aligned}$$

which implies that for $j \leq i$ we can write

$$ {}^{j}\mathbf{W}_i^{Tj}\dot{\mathbf{W}}_i = \tilde{\omega}_i^j - \tilde{\omega}_j^j. \tag{C.6} $$

Similarly, from (C.2), (C.3) and (C.5), for $j \leq i$, we have

$$ \begin{aligned} \Omega_i^j &= \mathbf{W}_i^T(\ddot{\mathbf{W}}_j\, {}^{j}\mathbf{W}_i + 2\dot{\mathbf{W}}_j\, {}^{j}\dot{\mathbf{W}}_i + \mathbf{W}_j\, {}^{j}\ddot{\mathbf{W}}_i) \\ &= {}^{j}\mathbf{W}_i^T(\mathbf{W}_j^T\ddot{\mathbf{W}}_j)\,{}^{j}\mathbf{W}_i + 2\,{}^{j}\mathbf{W}_i^T(\mathbf{W}_j^T\dot{\mathbf{W}}_j)\,{}^{j}\mathbf{W}_i({}^{j}\mathbf{W}_i^{Tj}\dot{\mathbf{W}}_i) + {}^{j}\mathbf{W}_i^{Tj}\ddot{\mathbf{W}}_i \\ &= \tilde{\dot{\omega}}_j^j + \tilde{\omega}_j^j\tilde{\omega}_j^j + 2\tilde{\omega}_j^j(\tilde{\omega}_i^j - \tilde{\omega}_j^j) + {}^{j}\mathbf{W}_i^{Tj}\ddot{\mathbf{W}}_i \\ &= \tilde{\dot{\omega}}_j^j - \tilde{\omega}_j^j\tilde{\omega}_j^j + 2\tilde{\omega}_j^j\tilde{\omega}_i^j + {}^{j}\mathbf{W}_i^{Tj}\ddot{\mathbf{W}}_i \\ &= -\,\Omega_j^{jT} + 2\tilde{\omega}_j^j\tilde{\omega}_i^j + {}^{j}\mathbf{W}_i^{Tj}\ddot{\mathbf{W}}_i, \end{aligned} $$

which finally gives,

$$ {}^{j}\mathbf{W}_i^{Tj}\ddot{\mathbf{W}}_i = \Omega_i^j + \Omega_j^{jT} - 2\tilde{\omega}_j^j\tilde{\omega}_i^j. \tag{C.7} $$

Furthermore, since $\mathbf{z}_j^j$ is constant relative to the j-th frame, its absolute derivative satisfies the equation

$$ \dot{\mathbf{z}}_j^j = \tilde{\omega}_j^j\mathbf{z}_j^j, $$

which in the i-th frame orientation is simply written as

$$ \dot{\mathbf{z}}_j^i = \tilde{\omega}_j^i\mathbf{z}_j^i. \tag{C.8} $$

Similarly the absolute acceleration $\ddot{\mathbf{z}}_j^j$ of $\mathbf{z}_j^j$, written in the i-th frame orientation, satisfies the following equation

$$ \begin{aligned} \ddot{\mathbf{z}}_j^i &= \Omega_j^i\mathbf{z}_j^i \\ &= \tilde{\dot{\omega}}_j^i\mathbf{z}_j^i + \tilde{\omega}_j^i\tilde{\omega}_j^i\mathbf{z}_j^i. \end{aligned} $$

Moreover, the dual tensors $\tilde{\dot{\mathbf{z}}}_j^i$ and $\tilde{\ddot{\mathbf{z}}}_j^i$ can be computed as follows

$$ \begin{aligned} \tilde{\dot{\mathbf{z}}}_j^i &= dual(\tilde{\omega}_j^i\mathbf{z}_j^i) \\ &= \tilde{\omega}_j^i\tilde{\mathbf{z}}_j^i - \tilde{\mathbf{z}}_j^i\tilde{\omega}_j^i \end{aligned} \tag{C.9} $$

and

$$ \tilde{\ddot{\mathbf{z}}}_j^i = dual(\tilde{\dot{\omega}}_j^i\mathbf{z}_j^i) + dual(\tilde{\omega}_j^i\tilde{\omega}_j^i\mathbf{z}_j^i). \tag{C.10} $$

Now, using equation (3.4.15) and (3.4.25) and some algebraic manipulations, equation

(C.10) can be simplified as follows :

$$\begin{aligned}\ddot{\tilde{\mathbf{z}}}_j^i &= [\dot{\tilde{\omega}}_j^i - \tilde{\omega}_j^i\tilde{\omega}_j^i]\tilde{\mathbf{z}}_j^i - \tilde{\mathbf{z}}_j^i[\dot{\tilde{\omega}}_j^i + \tilde{\omega}_j^i\tilde{\omega}_j^i] + tr\,[\tilde{\omega}_j^i\tilde{\omega}_j^i]\tilde{\mathbf{z}}_j^i \\ &= -\,\Omega_j^{i\,T}\tilde{\mathbf{z}}_j^i - \tilde{\mathbf{z}}_j^i\Omega_j^i + tr\,[\Omega_j^i]\tilde{\mathbf{z}}_j^i \qquad \text{(C.11)}\end{aligned}$$

since $tr\,[\Omega_j^i] = tr\,[\tilde{\omega}_j^i\tilde{\omega}_j^i]$.

After these preliminaries, we are in a position to prove the following lemmas. First, due to the nature of the transformation tensors $\mathbf{A}_i$ and $\mathbf{W}_i$, the following two results can be easily shown :

Lemma C-1 : The partial derivatives of $\mathbf{A}_i$, $\dot{\mathbf{A}}_i$ and $\ddot{\mathbf{A}}_i$, with respect to the generalized coordinate q_i and its time derivatives $\dot{q}_i$, $\ddot{q}_i$ satisfy the following equations :

$$i)\ \frac{\partial \mathbf{A}_i}{\partial q_i} = \frac{\partial \dot{\mathbf{A}}_i}{\partial \dot{q}_i} = \frac{\partial \ddot{\mathbf{A}}_i}{\partial \ddot{q}_i} = \mathbf{A}_i\tilde{\mathbf{z}}_i^i$$

$$ii)\ \frac{\partial \dot{\mathbf{A}}_i}{\partial q_i} = \dot{\mathbf{A}}_i\tilde{\mathbf{z}}_i^i, \qquad \frac{\partial \ddot{\mathbf{A}}_i}{\partial q_i} = \ddot{\mathbf{A}}_i\tilde{\mathbf{z}}_i^i$$

$$iii)\ \frac{\partial \ddot{\mathbf{A}}_i}{\partial \dot{q}_i} = 2\frac{\partial \dot{\mathbf{A}}_i}{\partial q_i}.$$

Lemma C-2 : The partial derivatives of $\mathbf{W}_i$ and its time derivatives $\dot{\mathbf{W}}_i$ and $\ddot{\mathbf{W}}_i$, with respect to the generalized coordinate q_j and its time derivatives $\dot{q}_j$, $\ddot{q}_j$ satisfy the following equations :

$$i)\ \frac{\partial \mathbf{W}_i}{\partial \dot{q}_j} = \frac{\partial \mathbf{W}_i}{\partial \ddot{q}_j} = \frac{\partial \dot{\mathbf{W}}_i}{\partial \ddot{q}_j} = 0$$

$$ii)\ \frac{\partial \mathbf{W}_i}{\partial q_j} = \frac{\partial \dot{\mathbf{W}}_i}{\partial \dot{q}_j} = \frac{\partial \ddot{\mathbf{W}}_i}{\partial \ddot{q}_j}$$

$$iii)\ \frac{\partial \ddot{\mathbf{W}}_i}{\partial \dot{q}_j} = 2\frac{\partial \dot{\mathbf{W}}_i}{\partial q_j}.$$

The proofs of these two lemmas are straightforward and are therefore omitted.

Lemma C-3 The partial derivatives of the transformation matrix $\mathbf{W}_i$ and its time

derivatives $\dot{W}_i$ and $\ddot{W}_i$, with respect to the generalized coordinate q_j, satisfy the following tensor equations :

$$i)\ \frac{\partial W_i}{\partial q_j} = \begin{cases} W_i \tilde{z}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \dot{W}_i}{\partial q_j} = \begin{cases} W_i \left[\dot{\tilde{z}}_j^i + \tilde{z}_j^i \tilde{\omega}_i^i \right] & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \ddot{W}_i}{\partial q_j} = \begin{cases} W_i \left[\ddot{\tilde{z}}_j^i + 2\dot{\tilde{z}}_j^i \tilde{\omega}_i^i + \tilde{z}_j^i \tilde{\Omega}_i^i \right] & j \leq i \\ 0 & j > i \end{cases}$$

Proof (outline) :

i) For $j \leq i$; since $W_i = W_j {}^jW_i$ and ${}^jW_i$ is independent of q_j, we have

$$\begin{aligned} \frac{\partial W_i}{\partial q_j} &= \frac{\partial W_j}{\partial q_j} {}^jW_i \\ &= W_{j-1} \frac{\partial A_j}{\partial q_j} {}^jW_i \\ &= W_j \tilde{z}_j^{jj} W_i \\ &= W_i \tilde{z}_j^i . \end{aligned}$$

ii) For $j \leq i$; from $\dot{W}_i = \dot{W}_j {}^jW_i + W_j {}^j\dot{W}_i$, and since

$$\begin{aligned} \frac{\partial \dot{W}_j}{\partial q_j} &= \dot{W}_{j-1} \frac{\partial A_j}{\partial q_j} + W_{j-1} \frac{\partial \dot{A}_j}{\partial q_j} \\ &= \dot{W}_j \tilde{z}_j^j , \end{aligned}$$

we have

$$\begin{aligned} \frac{\partial \dot{W}_i}{\partial q_j} &= \dot{W}_j \tilde{z}_j^{jj} W_i + W_j \tilde{z}_j^{jj} \dot{W}_i \\ &= \dot{W}_j {}^jW_i \tilde{z}_j^i + W_i \tilde{z}_j^{ij} W_i^{Tj} \dot{W}_i . \end{aligned}$$

Now since $\dot{W}_j = W_j \tilde{\omega}_j^j$, using (C.6), we can write

$$\frac{\partial \dot{W}_i}{\partial q_j} = W_j \tilde{\omega}_j^{jj} W_i \tilde{z}_j^j + W_i \tilde{z}_j^j \left(\tilde{\omega}_i^i - \tilde{\omega}_j^j\right)$$

$$= W_i \left[\tilde{\omega}_j^j \tilde{z}_j^j - \tilde{z}_j^j \tilde{\omega}_j^j + \tilde{z}_j^j \tilde{\omega}_i^i\right]$$

$$= W_i \left[dual\,(\tilde{\omega}_j^j z_j^j) + \tilde{z}_j^j \tilde{\omega}_i^i\right]$$

$$= W_i \left[\tilde{\dot{z}}_j^j + \tilde{z}_j^j \tilde{\omega}_i^i\right].$$

iii) For $j \leq i$; from $\ddot{W}_i = \ddot{W}_j{}^j W_i + 2\dot{W}_j{}^j \dot{W}_i + W_j{}^j \ddot{W}_i$, and since $\frac{\partial \ddot{W}_j}{\partial q_j} = \ddot{W}_j \tilde{z}_j^j$, we have

$$\frac{\partial \ddot{W}_i}{\partial q_j} = \ddot{W}_j \tilde{z}_j^{jj} W_i + 2\dot{W}_j \tilde{z}_j^{jj} \dot{W}_i + W_j \tilde{z}_j^{jj} \ddot{W}_i .$$

Also, since $\ddot{W}_j = W_j \Omega_j^j$ and $\dot{W}_j = W_j \tilde{\omega}_j^j$, the above expression becomes

$$\frac{\partial \ddot{W}_i}{\partial q_j} = W_j \left[\Omega_j^j \tilde{z}_j^{jj} W_i + 2\tilde{\omega}_j^j \tilde{z}_j^{jj} \dot{W}_i + \tilde{z}_j^{jj} \ddot{W}_i\right]$$

$$= W_i \left[\Omega_j^j \tilde{z}_j^j + 2\tilde{\omega}_j^j \tilde{z}_j^{jj} W_i^{Tj} \dot{W}_i + \tilde{z}_j^j{}^j W_i^{Tj} \ddot{W}_i\right].$$

Moreover, using (C.6) , (C.7), we have

$$\frac{\partial \ddot{W}_i}{\partial q_j} = W_i \left[\Omega_j^j \tilde{z}_j^j + 2\tilde{\omega}_j^j \tilde{z}_j^j \left(\tilde{\omega}_i^i - \tilde{\omega}_j^j\right) + \tilde{z}_j^j [\Omega_i^i + \Omega_j^{jT} - 2\tilde{\omega}_j^j \tilde{\omega}_i^i]\right]$$

$$= W_i \left[\Omega_j^j \tilde{z}_j^j + \tilde{z}_j^j \Omega_j^{jT} - 2\tilde{\omega}_j^j \tilde{z}_j^j \tilde{\omega}_j^j + 2[\tilde{\omega}_j^j \tilde{z}_j^j - \tilde{z}_j^j \tilde{\omega}_j^j]\tilde{\omega}_i^i + \tilde{z}_j^j \Omega_i^i\right]. \quad \text{(C.12)}$$

Now, the term $\Omega_j^j \tilde{z}_j^j + \tilde{z}_j^j \Omega_j^{jT} - 2\tilde{\omega}_j^j \tilde{z}_j^j \tilde{\omega}_j^j$ may be simplified as follows : Using the relation $\Omega_j^j = \tilde{\dot{\omega}}_j^j + \tilde{\omega}_j^j \tilde{\omega}_j^j$ and equation (3.4.22) we have

$$\Omega_j^j \tilde{z}_j^j + \tilde{z}_j^j \Omega_j^{jT} - 2\tilde{\omega}_j^j \tilde{z}_j^j \tilde{\omega}_j^j = \tilde{\dot{\omega}}_j^j \tilde{z}_j^j - \tilde{\omega}_j^j \tilde{\omega}_j^j \tilde{z}_j^j - \tilde{z}_j^j \tilde{\dot{\omega}}_j^j - \tilde{z}_j^j \tilde{\omega}_j^j \tilde{\omega}_i^i + tr\,[\tilde{\omega}_j^j \tilde{\omega}_j^j] \tilde{z}_j^j$$

$$= -\Omega_j^{jT} \tilde{z}_j^j - \tilde{z}_j^j \Omega_j^j + tr\,[\Omega_j^j] \tilde{z}_j^j$$

$$= \tilde{\ddot{z}}_j^j \quad \text{(C.13)}$$

where the last step follows from (C.11). Therefore, using (C.9) and (C.13), equation (C.12) can be written as

$$\frac{\partial \ddot{\mathbf{W}}_i}{\partial q_j} = \mathbf{W}_i [\ddot{\tilde{\mathbf{z}}}_j^i + 2\dot{\tilde{\mathbf{z}}}_j^i \tilde{\omega}_i^i + \tilde{\mathbf{z}}_j^i \Omega_i^i]$$

and this completes the proof □.

Lemma C-4

$$i)\ \mathbf{W}_i^T \frac{\partial \mathbf{W}_i}{\partial q_j} = \begin{cases} \tilde{\mathbf{z}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \mathbf{W}_i^T \frac{\partial \dot{\mathbf{W}}_i}{\partial q_j} = \begin{cases} \dot{\tilde{\mathbf{z}}}_j^i + \tilde{\mathbf{z}}_j^i \tilde{\omega}_i^i & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \mathbf{W}_i^T \frac{\partial \ddot{\mathbf{W}}_i}{\partial q_j} = \begin{cases} \ddot{\tilde{\mathbf{z}}}_j^i + 2\dot{\tilde{\mathbf{z}}}_j^i \tilde{\omega}_i^i + \tilde{\mathbf{z}}_j^i \Omega_i^i & j \leq i \\ 0 & j > i \end{cases}$$

Proof : Follows from Lemma C-3 □.

Now, we can evaluate the partial derivatives of the angular velocity and angular acceleration tensors $\tilde{\omega}_i^i$ and Ω_i^i with respect to the generalized coordinates.

Lemma C-5 The partial derivatives of the angular velocity tensor $\tilde{\omega}_i^i$, with respect to the generalized coordinates q_j, $\dot{q}_j$ and $\ddot{q}_j$ are given by,

$$i)\ \frac{\partial \tilde{\omega}_i^i}{\partial q_j} = \begin{cases} \dot{\tilde{\mathbf{z}}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \tilde{\omega}_i^i}{\partial \dot{q}_j} = \begin{cases} \tilde{\mathbf{z}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \tilde{\omega}_i^i}{\partial \ddot{q}_j} = 0 \qquad \text{for } \textit{all } j$$

Proof (outline) :

i) For $j \leq i$; since $\tilde{\omega}_i^i = \mathbf{W}_i^T \dot{\mathbf{W}}_i$, we have

$$\begin{aligned}\frac{\partial \tilde{\omega}_i^i}{\partial q_j} &= (\frac{\partial \mathbf{W}_i}{\partial q_j})^T \dot{\mathbf{W}}_i + \mathbf{W}_i^T \frac{\partial \dot{\mathbf{W}}_i}{\partial q_j} \\ &= (\mathbf{W}_i^T \frac{\partial \mathbf{W}_i}{\partial q_j})^T \mathbf{W}_i^T \dot{\mathbf{W}}_i + \mathbf{W}_i^T \frac{\partial \dot{\mathbf{W}}_i}{\partial q_j} \\ &= - \tilde{\mathbf{z}}_j^i \tilde{\omega}_i^i + \dot{\tilde{\mathbf{z}}}_j^i + \tilde{\mathbf{z}}_j^i \tilde{\omega}_i^i \\ &= \dot{\tilde{\mathbf{z}}}_j^i\end{aligned}$$

ii) For $j \leq i$; we have

$$\begin{aligned}\frac{\partial \tilde{\omega}_i^i}{\partial \dot{q}_j} &= \mathbf{W}_i^T \frac{\partial \dot{\mathbf{W}}_i}{\partial \dot{q}_j} \\ &= \mathbf{W}_i^T \frac{\partial \mathbf{W}_i}{\partial q_j} \\ &= \tilde{\mathbf{z}}_j^i \qquad \square.\end{aligned}$$

Lemma C-6 : The partial derivatives of the angular acceleration tensor Ω_i^i, with respect to the generalized coordinates q_j , $\dot{q}_j$ and $\ddot{q}_j$ are given by,

$$i)\ \frac{\partial \Omega_i^i}{\partial q_j} = \begin{cases} \ddot{\tilde{\mathbf{z}}}_j^i + 2\dot{\tilde{\mathbf{z}}}_j^i \tilde{\omega}_i^i & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \Omega_i^i}{\partial \dot{q}_j} = \begin{cases} 2[\dot{\tilde{\mathbf{z}}}_j^i + \tilde{\mathbf{z}}_j^i \tilde{\omega}_i^i] & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \Omega_i^i}{\partial \ddot{q}_j} = \begin{cases} \tilde{\mathbf{z}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

Proof (outline) :

i) for $j \leq i$; since $\Omega_i^i = \mathbf{W}_i^T \ddot{\mathbf{W}}_i$, we have

$$\begin{aligned}\frac{\partial \Omega_i^i}{\partial q_j} &= (\frac{\partial \mathbf{W}_i}{\partial q_j})^T \ddot{\mathbf{W}}_i + \mathbf{W}_i^T \frac{\partial \ddot{\mathbf{W}}_i}{\partial q_j} \\ &= (\mathbf{W}_i^T \frac{\partial \mathbf{W}_i}{\partial q_j})^T \mathbf{W}_i^T \ddot{\mathbf{W}}_i + \mathbf{W}_i^T \frac{\partial \ddot{\mathbf{W}}_i}{\partial q_j}\end{aligned}$$

$$= - \tilde{\mathbf{z}}_j^i \Omega_i^i + \ddot{\tilde{\mathbf{z}}}_j^i + 2\dot{\tilde{\mathbf{z}}}_j^i \tilde{\omega}_i^i + \tilde{\mathbf{z}}_j^i \Omega_i^i$$
$$= \ddot{\tilde{\mathbf{z}}}_j^i + 2\dot{\tilde{\mathbf{z}}}_j^i \tilde{\omega}_i^i.$$

ii) for $j \leq i$; we have

$$\frac{\partial \Omega_i^i}{\partial \dot{q}_j} = \mathbf{W}_i^T \frac{\partial \ddot{\mathbf{W}}_i}{\partial \dot{q}_j}$$
$$= 2\mathbf{W}_i^T \frac{\partial \dot{\mathbf{W}}_i}{\partial q_j}$$
$$= 2[\dot{\tilde{\mathbf{z}}}_j^i + \tilde{\mathbf{z}}_j^i \tilde{\omega}_i^i].$$

iii) for $j \leq i$; we have

$$\frac{\partial \Omega_i^i}{\partial \ddot{q}_j} = \mathbf{W}_i^T \frac{\partial \ddot{\mathbf{W}}_i}{\partial \ddot{q}_j}$$
$$= \mathbf{W}_i^T \frac{\partial \mathbf{W}_i}{\partial q_j}$$
$$= \tilde{\mathbf{z}}_j^i \qquad \square.$$

Furthemore, by using Lemma C-6, we can derive formulas for computing the partial derivatives ot the dual tensor of the angular acceleration vector $\dot{\omega}_i^i$ with respect to the generalized coordinates.

Lemma C-7

i) $$\frac{\partial \tilde{\dot{\omega}}_i^i}{\partial q_j} = \begin{cases} \ddot{\tilde{\mathbf{z}}}_j^i + dual(\dot{\tilde{\mathbf{z}}}_j^i \omega_i^i) & j \leq i \\ 0 & j > i \end{cases}$$

ii) $$\frac{\partial \tilde{\dot{\omega}}_i^i}{\partial \dot{q}_j} = \begin{cases} 2\dot{\tilde{\mathbf{z}}}_j^i + dual(\tilde{\mathbf{z}}_j^i \omega_i^i) & j \leq i \\ 0 & j > i \end{cases}$$

iii) $$\frac{\partial \tilde{\dot{\omega}}_i^i}{\partial \ddot{q}_j} = \begin{cases} \tilde{\mathbf{z}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

Proof (outline) : Follows from the equation

$$\tilde{\omega}_i^i = \frac{\Omega_i^i - \Omega_i^{i^T}}{2}$$

and Lemma C-6 □.

Note that the partial derivatives, with respect to the generalized coordinates, of the angular velocity and angular acceleration vectors with respect to the generalized coordinates are readily available from Lemmas C-5 and C-7.

Now, from Algorithm 5.5, it is obvious that to compute the Jacobians which define the coefficient sensitivity matrices $\mathbf{D}^o$, $\mathbf{V}^o$ and $\mathbf{P}^o$, we need to compute the partial derivatives (with respect to the generalized coordinates $\ddot{q}_j$, $\dot{q}_j$ and q_j) of the vectors η_i^i (for revolute joints) which are defined by (5.3.46b). Equation (5.3.46b), for $i \leq j \leq N$, can also be written as

$$\eta_i^i = \sum_{k=i}^{j-1} {}^i\mathbf{W}_k \left[\mu_k^k + \tilde{\mathbf{u}}_{O_k}^k \ddot{\mathbf{s}}_{0,k}^k + \tilde{\mathbf{s}}_{k,k+1}^k \ddot{\mathbf{U}}_{O_{k+1}}^k\right] + {}^i\mathbf{W}_j \eta_j^j. \tag{C.14}$$

Therefore, to compute the partial derivatives of the vectors η_i^i, we need to compute the partial derivatives of various vector functions which appear in equation (C.14). These partial derivatives are derived as follows.

Lemma C-8 : The partial derivatives, with respect to the generalized coordinates, of the vector function μ_i^i are given by :

$$i)\ \frac{\partial \mu_i^i}{\partial \ddot{q}_j} = \begin{cases} \mathbf{K}_{O_i}^i \mathbf{z}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \mu_i^i}{\partial \dot{q}_j} = \begin{cases} 2[\mathbf{K}_{O_i}^i \dot{\mathbf{z}}_j^i - \hat{\mathbf{L}}_i^i \mathbf{z}_j^i] & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \mu_i^i}{\partial q_j} = \begin{cases} \mathbf{K}_{O_i}^i \ddot{\mathbf{z}}_j^i - 2\hat{\mathbf{L}}_i^i \dot{\mathbf{z}}_j^i & j \leq i \\ 0 & j > i \end{cases}$$

where $\mathbf{K}_{O_i}^i$ is the inertia tensor of the i-th augmented link, $\hat{\mathbf{L}}_i^i = \tilde{\omega}_i^i \mathbf{K}_{O_i}^i$ and

$$\hat{K}_{O_i}^i = \frac{1}{2} tr\,[K_{O_i}^i]1 - K_{O_i}^i .$$

Proof (outline) : As we have shown in Chapter IV, the vector function μ_i^i may be defined by the equation

$$\mu_i^i = K_{O_i}^i \dot{\omega}_i^i + \tilde{\omega}_i^i K_{O_i}^i \omega_i^i$$

Now, since the inertia tensor of the i-th augmented link $K_{O_i}^i$ is independent of the generalized coordinates, (note that we assumed revolute joints only), we have the following :

i) follows from Lemmas (C-5) and (C-7).

ii) is obvious for $j > i$. For $j \leq i$, by using Lemmas (C-5), (C-7) and the tensor equations (3.4.4) and (3.4.25), we get

$$\begin{aligned}
\frac{\partial \mu_i^i}{\partial \dot{q}_j} &= K_{O_i}^i [2\dot{z}_j^i + (\tilde{z}_j^i \omega_i^i)] + \tilde{z}_j^i (K_{O_i}^i \omega_i^i) + \tilde{\omega}_i^i K_{O_i}^i z_j^i \\
&= 2K_{O_i}^i \dot{z}_j^i - K_{O_i}^i \tilde{\omega}_i^i z_j^i - dual\,(K_{O_i}^i \omega_i^i) z_j^i + \tilde{\omega}_i^i K_{O_i}^i z_j^i \qquad \text{by (3.4.4)} \\
&= 2K_{O_i}^i \dot{z}_j^i + [-K_{O_i}^i \tilde{\omega}_i^i - dual\,(K_{O_i}^i \omega_i^i) + \tilde{\omega}_i^i K_{O_i}^i] z_j^i \\
&= 2K_{O_i}^i \dot{z}_j^i + [2\tilde{\omega}_i^i K_{O_i}^i - tr\,(K_{O_i}^i)\tilde{\omega}_i^i] z_j^i \qquad \text{by (3.4.25)} \\
&= 2K_{O_i}^i \dot{z}_j^i - 2\tilde{\omega}_i^i \hat{K}_{O_i}^i z_j^i \\
&= 2[K_{O_i}^i \dot{z}_j^i - \hat{L}_i^i z_j^i]
\end{aligned}$$

where, $\hat{L}_i^i = \tilde{\omega}_i^i \hat{K}_{O_i}^i$ and $\hat{K}_{O_i}^i = \frac{1}{2} tr\,(K_{O_i}^i)1 - K_{O_i}^i$.

iii) the proof is similar to that of *ii*) □.

Lemma C-9 : The partial derivatives with respect to the generalized coordinates of the vector function $\ddot{U}_{O_i}^i$ are given by :

$$i)\ \frac{\partial \ddot{U}_{O_i}^i}{\partial q_j} = \begin{cases} \ddot{\tilde{z}}_j^i U_{O_i}^i + 2\dot{\tilde{z}}_j^i \dot{U}_{O_i}^i & j \leq i \\ \ddot{\tilde{z}}_j^i U_{O_j}^i + 2\dot{\tilde{z}}_j^i \dot{U}_{O_j}^i + \tilde{z}_j^i \ddot{U}_{O_j}^i & j > i \end{cases}$$

$$ii)\ \frac{\partial \ddot{U}_{O_i}^i}{\partial \dot{q}_j} = \begin{cases} 2[\dot{\tilde{z}}_j^i U_{O_i}^i + \tilde{z}_j^i \dot{U}_{O_i}^i] & j \leq i \\ 2[\dot{\tilde{z}}_j^i U_{O_j}^i + \tilde{z}_j^i \dot{U}_{O_j}^i] & j > i \end{cases}$$

$$iii)\ \frac{\partial \ddot{\mathbf{U}}^i_{O_i}}{\partial \ddot{q}_j} = \begin{cases} \tilde{\mathbf{z}}^i_j \mathbf{U}^i_{O_i} & j \leq i \\ \tilde{\mathbf{z}}^i_j \mathbf{U}^i_{O_j} & j > i \end{cases}$$

Proof (outline) : From equation (5.3.46a), we have $\ddot{\mathbf{U}}^i_{O_i} = \sum_{k=i}^{N} {}^i\mathbf{W}_k \boldsymbol{\Omega}^k_k \mathbf{u}^k_{O_k}$, which, for $i < j$, can be written as $\ddot{\mathbf{U}}^i_{O_i} = \sum_{k=i}^{j-1} {}^i\mathbf{W}_k \boldsymbol{\Omega}^k_k \mathbf{u}^k_{O_k} + {}^i\mathbf{W}_j \ddot{\mathbf{U}}^j_{O_j}$. Then,

i) from Lemma C-6 and the fact that $\mathbf{u}^i_{O_i}$ is independent of the generalized coordinates (for revolute joints), we have

$$\frac{\partial \ddot{\mathbf{U}}^i_{O_i}}{\partial q_j} = \begin{cases} \sum_{k=i}^{N} {}^i\mathbf{W}_k \dfrac{\partial \boldsymbol{\Omega}^k_k}{\partial q_j} \mathbf{u}^k_{O_k} & j \leq i \\ \dfrac{\partial {}^i\mathbf{W}_j}{\partial q_j} \ddot{\mathbf{U}}^j_{O_j} + {}^i\mathbf{W}_j \dfrac{\partial \ddot{\mathbf{U}}^j_{O_j}}{\partial q_j} & j > i \end{cases}$$

$$\frac{\partial \ddot{\mathbf{U}}^i_{O_i}}{\partial \dot{q}_j} = \begin{cases} \sum_{k=i}^{N} {}^i\mathbf{W}_k [\ddot{\tilde{\mathbf{z}}}^k_j + 2\dot{\tilde{\mathbf{z}}}^k_j \tilde{\omega}^k_k] \mathbf{u}^k_{O_k} = \ddot{\tilde{\mathbf{z}}}^i_j \mathbf{U}^i_{O_i} + 2\dot{\tilde{\mathbf{z}}}^i_j \dot{\mathbf{U}}^i_{O_i} & j \leq i \\ {}^i\mathbf{W}_j \tilde{\mathbf{z}}^i_j \ddot{\mathbf{U}}^j_{O_j} + {}^i\mathbf{W}_j [\ddot{\tilde{\mathbf{z}}}^i_j \mathbf{U}^j_{O_j} + 2\dot{\tilde{\mathbf{z}}}^i_j \dot{\mathbf{U}}^j_{O_j}] = \ddot{\tilde{\mathbf{z}}}^i_j \mathbf{U}^i_{O_j} + 2\dot{\tilde{\mathbf{z}}}^i_j \dot{\mathbf{U}}^i_{O_j} + \tilde{\mathbf{z}}^i_j \ddot{\mathbf{U}}^i_{O_j} & j > i \end{cases}$$

ii) and *iii*) can be proved in a similar manner □.

To compute the partial derivatives of $\ddot{\mathbf{s}}^i_{0,i}$, we need the following results.

Lemma C-10 :

$$i)\ \frac{\partial \ddot{\mathbf{s}}^i_{i,i+1}}{\partial q_j} = \begin{cases} \ddot{\tilde{\mathbf{z}}}^i_j \mathbf{s}^i_{i,i+1} + 2\dot{\tilde{\mathbf{z}}}^i_j \dot{\mathbf{s}}^i_{i,i+1} & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \ddot{\mathbf{s}}^i_{i,i+1}}{\partial \dot{q}_j} = \begin{cases} 2[\dot{\tilde{\mathbf{z}}}^i_j \mathbf{s}^i_{i,i+1} + \tilde{\mathbf{z}}^i_j \dot{\mathbf{s}}^i_{i,i+1}] & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \ddot{\mathbf{s}}^i_{i,i+1}}{\partial \ddot{q}_j} = \begin{cases} \tilde{\mathbf{z}}^i_j \mathbf{s}^i_{i,i+1} & j \leq i \\ 0 & j > i \end{cases}$$

Proof : The results follow from the relation $\ddot{\mathbf{s}}^i_{i,i+1} = \boldsymbol{\Omega}^i_i \mathbf{s}^i_{i,i+1}$ and the partial derivatives of $\boldsymbol{\Omega}^i_i$ □.

Lemma C-11 : : The partial derivatives, with respect to the generalized coordinates of the vector function $\ddot{\mathbf{s}}^i_{0,i}$ are given by :

$$i)\ \frac{\partial \ddot{\mathbf{s}}^i_{0,i}}{\partial q_j} = \begin{cases} \ddot{\tilde{\mathbf{z}}}^j_j \mathbf{s}^j_{j,i} + 2\dot{\tilde{\mathbf{z}}}^j_j \dot{\mathbf{s}}^j_{j,i} - \tilde{\mathbf{z}}^j_j \ddot{\mathbf{s}}^i_{0,j} & j \leq i \\ 0 & j > i \end{cases}$$

$$ii)\ \frac{\partial \ddot{\mathbf{s}}^i_{0,i}}{\partial \dot{q}_j} = \begin{cases} 2[\dot{\tilde{\mathbf{z}}}^j_j \mathbf{s}^j_{j,i} + \tilde{\mathbf{z}}^j_j \dot{\mathbf{s}}^j_{j,i}] & j \leq i \\ 0 & j > i \end{cases}$$

$$iii)\ \frac{\partial \ddot{\mathbf{s}}^i_{0,i}}{\partial \ddot{q}_j} = \begin{cases} \tilde{\mathbf{z}}^j_j \mathbf{s}^j_{j,i} & j \leq i \\ 0 & j > i \end{cases}$$

Proof (outline) : For $j \leq i$ we can write

$$\ddot{\mathbf{s}}^i_{0,i} = \ddot{\mathbf{s}}^i_{0,j} + \ddot{\mathbf{s}}^i_{j,i} = {}^{(j-1)}\mathbf{W}_i{}^T \ddot{\mathbf{s}}^{j-1}_{0,j} + \sum_{k=j}^{i-1} {}^k\mathbf{W}_i{}^T \ddot{\mathbf{s}}^k_{k,k+1}$$

i) From Lemmas C-1 and C-8, we get

$$\begin{aligned} \frac{\partial \ddot{\mathbf{s}}^i_{0,i}}{\partial q_j} &= {}^j\mathbf{W}_i{}^T \tilde{\mathbf{z}}^j_j{}^T \mathbf{A}^T_j \ddot{\mathbf{s}}^{j-1}_{0,j} + \sum_{k=j}^{i-1} {}^k\mathbf{W}_i{}^T [\ddot{\tilde{\mathbf{z}}}^k_j \mathbf{s}^k_{k,k+1} + 2\dot{\tilde{\mathbf{z}}}^k_j \dot{\mathbf{s}}^k_{k,k+1}] \\ &= -\tilde{\mathbf{z}}^j_j \ddot{\mathbf{s}}^i_{0,j} + 2\dot{\tilde{\mathbf{z}}}^j_j \dot{\mathbf{s}}^j_{j,i} + \ddot{\tilde{\mathbf{z}}}^j_j \mathbf{s}^j_{j,i} \end{aligned}$$

ii) and *iii*) can be proved similarly []

Now, we are in a position to derive the partial derivatives, with respect to the generalized coordinates, of the vector η^i_i defined by equation (C.14).

Lemma C-12 : : The partial derivatives, with respect to the generalized coordinates, of the vector function η^i_i are given by :

$$i)\ \frac{\partial \eta^i_i}{\partial \ddot{q}_j} = \begin{cases} [\mathbf{E}^i_{O_i} - \tilde{\mathbf{U}}^i_{O_i} \tilde{\mathbf{s}}^i_{j,i}]\mathbf{z}^i_j & j \leq i \\ {}^i\mathbf{W}_j [\mathbf{E}^j_{O_j} - \tilde{\mathbf{s}}^j_{i,j} \tilde{\mathbf{U}}^j_{O_j}]\mathbf{z}^j_j & j > i \end{cases}$$

$$ii)\ \frac{\partial \eta^i_i}{\partial \dot{q}_j} = 2 \begin{cases} [\mathbf{E}^i_{O_i} - \tilde{\mathbf{U}}^i_{O_i} \tilde{\mathbf{s}}^i_{j,i}]\dot{\mathbf{z}}^i_j - [\mathbf{L}^i_i + \tilde{\mathbf{U}}^i_{O_i} \dot{\tilde{\mathbf{s}}}^i_{j,i}]\mathbf{z}^i_j & j \leq i \\ {}^i\mathbf{W}_j \left([\mathbf{E}^j_{O_j} - \tilde{\mathbf{s}}^j_{i,j} \tilde{\mathbf{U}}^j_{O_j}]\dot{\mathbf{z}}^j_j - [\mathbf{L}^j_j + \tilde{\mathbf{s}}^j_{i,j} \dot{\tilde{\mathbf{U}}}^j_{O_j}]\mathbf{z}^j_j \right) & j > i \end{cases}$$

$$iii)\ \frac{\partial \eta_i^i}{\partial q_j} = \begin{cases} [\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i\, \tilde{\mathbf{s}}_{j,i}^j]\ddot{\mathbf{z}}_j^j - 2[\mathbf{L}_i^j + \tilde{\mathbf{U}}_{O_i}^i\, \dot{\tilde{\mathbf{s}}}_{j,i}^j]\dot{\mathbf{z}}_j^j + \tilde{\mathbf{U}}_{O_i}^i\, \ddot{\tilde{\mathbf{s}}}_{0,j}^i \mathbf{z}_j^j & j \le i \\ {}^iW_j([\mathbf{E}_{O_j}^j - \tilde{\mathbf{s}}_{i,j}^j \tilde{\mathbf{U}}_{O_j}^j]\ddot{\mathbf{z}}_j^j - 2[\mathbf{L}_j^j + \tilde{\mathbf{s}}_{i,j}^j \dot{\tilde{\mathbf{U}}}_{O_j}^j]\dot{\mathbf{z}}_j^j + [\tilde{\mathbf{U}}_{O_j}^j \ddot{\tilde{\mathbf{s}}}_{0,j}^j - \tilde{\mathbf{s}}_{i,j}^j \ddot{\tilde{\mathbf{U}}}_{O_j}^j - \tilde{\eta}_j^j]\mathbf{z}_j^j) & j > i \end{cases}$$

where $\mathbf{E}_{O_i}^i$ ($\mathbf{U}_{O_i}^i$) is the second (first) moment of the i-th generalized link, and the tensor $\mathbf{L}_i^j$ satisfies the equation

$$\mathbf{L}_i^j = \hat{\mathbf{L}}_i^j + [\tilde{\mathbf{s}}_{i,i+1}^j \dot{\tilde{\mathbf{U}}}_{O_{i+1}}^i + \tilde{\mathbf{U}}_{O_{i+1}}^i \dot{\tilde{\mathbf{s}}}_{i,i+1}^i] + \mathbf{A}_{i+1}\mathbf{L}_{i+1}^{i+1}\mathbf{A}_{i+1}^T \tag{C.15}$$

Proof (outline) : Using Lemmas C-8, C-9 and C-11, the partial derivarives of η_i^i are derived as follows :

) There are two cases :

a) For $j \le i$, we have

$$\frac{\partial \eta_i^i}{\partial \ddot{q}_j} = \sum_{k=i}^{N} {}^iW_k \left[\frac{\partial \mu_k^k}{\partial \ddot{q}_j} + \tilde{\mathbf{u}}_{O_k}^k \frac{\partial \ddot{\mathbf{s}}_{0,k}^k}{\partial \ddot{q}_j} + \tilde{\mathbf{s}}_{k,k+1}^k \frac{\partial \ddot{\mathbf{U}}_{O_{k+1}}^k}{\partial \ddot{q}_j}\right]$$

$$= \sum_{k=i}^{N} {}^iW_k \left[\mathbf{K}_{O_k}^k \mathbf{z}_j^k + \tilde{\mathbf{u}}_{O_k}^k\, \tilde{\mathbf{z}}_j^k \mathbf{s}_{j,k}^k + \tilde{\mathbf{s}}_{k,k+1}^k \tilde{\mathbf{z}}_j^k \mathbf{U}_{O_{k+1}}^k\right]$$

$$= \sum_{k=i}^{N} {}^iW_k \left[\mathbf{K}_{O_k}^k - \tilde{\mathbf{u}}_{O_k}^k\, \tilde{\mathbf{s}}_{j,k}^k - \tilde{\mathbf{s}}_{k,k+1}^k \tilde{\mathbf{U}}_{O_{k+1}}^k\right] {}^iW_k^T \mathbf{z}_j^i \qquad \text{by (C-3).}$$

Now, for $j \le i \le k$ $\tilde{\mathbf{s}}_{j,k}^k = \tilde{\mathbf{s}}_{j,i}^k + \tilde{\mathbf{s}}_{i,k}^k$. Therefore, we have

$$\frac{\partial \eta_i^i}{\partial \ddot{q}_j} = \sum_{k=i}^{N} {}^iW_k \left[\mathbf{K}_{O_k}^k - \tilde{\mathbf{u}}_{O_k}^k\, \tilde{\mathbf{s}}_{i,k}^k - \tilde{\mathbf{s}}_{k,k+1}^k \tilde{\mathbf{U}}_{O_{k+1}}^k\right] {}^iW_k^T \mathbf{z}_j^i - \sum_{k=i}^{N} {}^iW_k \left[\tilde{\mathbf{u}}_{O_k}^k \tilde{\mathbf{s}}_{j,i}^k\right] {}^iW_k^T \mathbf{z}_j^i$$

$$= [\mathbf{E}_{O_i}^i - \tilde{\mathbf{U}}_{O_i}^i\, \tilde{\mathbf{s}}_{j,i}^i]\mathbf{z}_j^i$$

where,

$$\mathbf{E}_{O_i}^i = \sum_{k=i}^{N} {}^iW_k \left[\mathbf{K}_{O_k}^k - \tilde{\mathbf{u}}_{O_k}^k\, \tilde{\mathbf{s}}_{i,k}^k - \tilde{\mathbf{s}}_{k,k+1}^k \tilde{\mathbf{U}}_{O_{k+1}}^k\right] {}^iW_k^T \tag{C.16}$$

is the inertia tensor of the i-th generalized link, and

$$\tilde{\mathbf{U}}_{O_i}^i = \sum_{k=i}^{N} {}^iW_k\, \tilde{\mathbf{u}}_{O_k}^k\, {}^iW_k^T \tag{C.17}$$

is the dual tensor of the first moment of the i-th generalized link. To see that $\mathbf{E}_{O_i}^i$ is the inertia tensor of the i-th generalized link, we shall show that equation (C.16) is

equivalent to equation (6.3.6). Since $\tilde{\mathbf{s}}^i_{i,i} = 0$, we have

$$\begin{aligned}
\mathbf{E}^i_{O_i} &= \mathbf{K}^i_{O_i} - \tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} \\
&\quad + \sum_{k=i+1}^{N} {}^i\mathbf{W}_k \left[\mathbf{K}^k_{O_i} - \tilde{\mathbf{u}}^k_{O_i}(\tilde{\mathbf{s}}^k_{i,i+1} + \tilde{\mathbf{s}}^k_{i+1,k}) - \tilde{\mathbf{s}}^k_{k,k+1}\tilde{\mathbf{U}}^k_{O_{i+1}}\right] {}^i\mathbf{W}_k^T \\
&= \mathbf{K}^i_{O_i} - \tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} - \sum_{k=i+1}^{N} {}^i\mathbf{W}_k\, \tilde{\mathbf{u}}^k_{O_i}\tilde{\mathbf{s}}^k_{i,i+1}\, {}^i\mathbf{W}_k^T + \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{O_{i+1}}\mathbf{A}^T_{i+1} \\
&= \mathbf{K}^i_{O_i} - \tilde{\mathbf{s}}^i_{i,i+1}\tilde{\mathbf{U}}^i_{O_{i+1}} - \tilde{\mathbf{U}}^i_{O_{i+1}}\tilde{\mathbf{s}}^i_{i,i+1} + \mathbf{A}_{i+1}\mathbf{E}^{i+1}_{O_{i+1}}\mathbf{A}^T_{i+1}.
\end{aligned}$$

which implies that indeed equation (C.16) is equivalent to equation (6.3.6). Thus, case a) of part *i*) has been proved.

b) For $j > i$, we have

$$\begin{aligned}
\frac{\partial \eta^i_i}{\partial \ddot{q}_j} &= \sum_{k=i}^{j-1} {}^i\mathbf{W}_k \left[\frac{\partial \mu^k_k}{\partial \ddot{q}_j} + \tilde{\mathbf{u}}^k_{O_i}\frac{\partial \ddot{\tilde{\mathbf{s}}}^k_{0,k}}{\partial \ddot{q}_j} + \tilde{\mathbf{s}}^k_{k,k+1}\frac{\partial \ddot{\tilde{\mathbf{U}}}^k_{O_{i+1}}}{\partial \ddot{q}_j}\right] + {}^i\mathbf{W}_j \frac{\partial \eta^j_j}{\partial \ddot{q}_j} \\
&= \sum_{k=i}^{j-1} {}^i\mathbf{W}_k \left[\tilde{\mathbf{s}}^k_{k,k+1}\tilde{\mathbf{z}}^k_j \mathbf{U}^k_{O_j}\right] + {}^i\mathbf{W}_j \frac{\partial \eta^j_j}{\partial \ddot{q}_j} \\
&= {}^i\mathbf{W}_j \left[-\tilde{\mathbf{s}}^j_{i,j}\tilde{\mathbf{U}}^j_{O_j}\mathbf{z}^j_j\right] + {}^i\mathbf{W}_j \left[\mathbf{E}^j_{O_j} - \tilde{\mathbf{U}}^j_{O_j}\tilde{\mathbf{s}}^j_{j,j}\right]\mathbf{z}^j_j \qquad \text{(by part } a\text{)} \\
&= {}^i\mathbf{W}_j \left[\mathbf{E}^j_{O_j} - \tilde{\mathbf{s}}^j_{i,j}\tilde{\mathbf{U}}^j_{O_j}\right]\mathbf{z}^j_j, \qquad \text{since } \tilde{\mathbf{s}}^j_{j,j} = 0
\end{aligned}$$

ii) As in part *i*), we have two cases.

a) For $j \leq i$, from Lemmas C-8, C-9 and C-11, we get

$$\begin{aligned}
\frac{\partial \eta^i_i}{\partial \dot{q}_j} &= \sum_{k=i}^{N} {}^i\mathbf{W}_k \Big\{ 2\left[\mathbf{K}^k_{O_i}\dot{\mathbf{z}}^k_j - \hat{\mathbf{L}}^k_k\mathbf{z}^k_j\right] \\
&\quad + 2\tilde{\mathbf{u}}^k_{O_i}\left[\dot{\tilde{\mathbf{z}}}^k_j\mathbf{s}^k_{j,k} + \tilde{\mathbf{z}}^k_j\dot{\mathbf{s}}^k_{j,k}\right] + 2\tilde{\mathbf{s}}^k_{k,k+1}\left[\dot{\tilde{\mathbf{z}}}^k_j\mathbf{U}^k_{O_{i+1}} + \tilde{\mathbf{z}}^k_j\dot{\mathbf{U}}^k_{O_{i+1}}\right]\Big\} \\
&= 2\sum_{k=i}^{N} {}^i\mathbf{W}_k \left[\mathbf{K}^k_{O_i} - \tilde{\mathbf{u}}^k_{O_i}\,\tilde{\mathbf{s}}^k_{j,k} - \tilde{\mathbf{s}}^k_{k,k+1}\tilde{\mathbf{U}}^k_{O_{i+1}}\right] {}^i\mathbf{W}_k^T\dot{\mathbf{z}}^i_j \\
&\quad - 2\sum_{k=i}^{N} {}^i\mathbf{W}_k \left[\hat{\mathbf{L}}^k_k + \tilde{\mathbf{u}}^k_{O_i}\dot{\tilde{\mathbf{s}}}^k_{i,k} + \tilde{\mathbf{s}}^k_{k,k+1}\dot{\bar{\mathbf{U}}}^k_{O_{i+1}}\right] {}^i\mathbf{W}_k^T\mathbf{z}^i_j \\
&\quad - 2\sum_{k=i}^{N} {}^i\mathbf{W}_k \left[\tilde{\mathbf{u}}^k_{O_i}\dot{\tilde{\mathbf{s}}}^k_{j,i}\right] {}^i\mathbf{W}_k^T\mathbf{z}^i_j \\
&= 2\left(\left[\mathbf{E}^i_{O_i} - \tilde{\mathbf{U}}^i_{O_i}\,\tilde{\mathbf{s}}^i_{j,i}\right]\dot{\mathbf{z}}^i_j - \left[\mathbf{L}^i_i + \dot{\tilde{\mathbf{U}}}^i_{O_i}\,\tilde{\mathbf{s}}^i_{j,i}\right]\mathbf{z}^i_j\right)
\end{aligned}$$

where $L_i^i = \sum_{k=i}^{N} {}^iW_k [\hat{L}_k^k + \tilde{u}_{O_k}^k \tilde{\dot{s}}_{i,k}^k + \tilde{s}_{k,k+1}^k \dot{\tilde{U}}_{O_{k+1}}^k] {}^iW_k^T$. To see that L_i^i satisfies (C.15) we proceed as follows :

Since $\tilde{\dot{s}}_{i,i}^i = 0$, we have

$$\begin{aligned} L_i^i &= \hat{L}_i^i + \tilde{s}_{i,i+1}^i \dot{\tilde{U}}_{O_{i+1}}^i \\ &\quad + \sum_{k=i+1}^{N} {}^iW_k [\hat{L}_k^k + \tilde{u}_{O_k}^k (\tilde{\dot{s}}_{i,i+1}^k + \tilde{\dot{s}}_{i+1,k}^k) + \tilde{s}_{k,k+1}^k \dot{\tilde{U}}_{O_{k+1}}^k] {}^iW_k^T \\ &= \hat{L}_i^i + \tilde{s}_{i,i+1}^i \dot{\tilde{U}}_{O_{i+1}}^i + \sum_{k=i+1}^{N} {}^iW_k \tilde{u}_{O_k}^k \tilde{\dot{s}}_{i,i+1}^k {}^iW_k^T + A_{i+1} L_{i+1}^{i+1} A_{i+1}^T \\ &= \hat{L}_i^i + [\tilde{s}_{i,i+1}^i \dot{\tilde{U}}_{O_{i+1}}^i + \tilde{U}_{O_{i+1}}^i \tilde{\dot{s}}_{i,i+1}^i] + A_{i+1} L_{i+1}^{i+1} A_{i+1}^T \end{aligned}$$

Therefore, case a) of part *ii*) has been proved.

b) For $j > i$, using (C.14) we get

$$\begin{aligned} \frac{\partial \eta_i^i}{\partial \dot{q}_j} &= 2 \sum_{k=i}^{j-1} {}^iW_k \tilde{s}_{k,k+1}^k (\tilde{\dot{z}}_j^k U_{O_j}^k + \tilde{z}_j^k \dot{U}_{O_j}^k) + {}^iW_j \frac{\partial \eta_j^j}{\partial \dot{q}_j} \\ &= 2 {}^iW_j [- \tilde{s}_{i,j}^j \tilde{U}_{O_j}^j \dot{z}_j^j - \tilde{s}_{i,j}^j \dot{\tilde{U}}_{O_j}^j z_j^j] + 2 {}^iW_j [E_{O_j}^j \dot{z}_j^j - L_j^j z_j^j] \\ &= 2 {}^iW_j \left([E_{O_j}^j - \tilde{s}_{i,j}^j \tilde{U}_{O_j}^j] \dot{z}_j^j - [L_j^j + \tilde{s}_{i,j}^j \dot{\tilde{U}}_{O_j}^j] z_j^j \right) \end{aligned}$$

which completes the proof of part *ii*).

iii) As before using Lemmas C-8, C-9 and C-11, we have

a) for $j \leq i$

$$\begin{aligned} \frac{\partial \eta_i^i}{\partial q_j} &= \sum_{k=i}^{N} {}^iW_k [\frac{\partial \mu_k^k}{\partial q_j} + \tilde{u}_{O_k}^k \frac{\partial \ddot{\tilde{z}}_{j,k}^k}{\partial q_j} + \tilde{s}_{k,k+1}^k \frac{\partial \ddot{\tilde{U}}_{O_{k+1}}^k}{\partial q_j}] \\ &= \sum_{k=i}^{N} {}^iW_k [K_{O_k}^k - \tilde{u}_{O_k}^k \tilde{s}_{j,k}^k - \tilde{s}_{k,k+1}^k \tilde{U}_{O_{k+1}}^k] {}^iW_k^T \ddot{z}_j^i \\ &\quad - 2 \sum_{k=i}^{N} {}^iW_k [\hat{L}_k^k + \tilde{u}_{O_k}^k \tilde{\dot{s}}_{j,k}^k + \tilde{s}_{k,k+1}^k \dot{\tilde{U}}_{O_{k+1}}^k] {}^iW_k^T \dot{z}_j^i + \sum_{k=i}^{N} {}^iW_k [\tilde{u}_{O_k}^k \tilde{\ddot{s}}_{0,j}^k] {}^iW_k^T z_j^i \\ &= [E_{O_i}^i - \tilde{U}_{O_i}^i \tilde{s}_{j,i}^i] \ddot{z}_j^i - 2[L_i^i + \tilde{U}_{O_i}^i \tilde{\dot{s}}_{j,i}^i] \dot{z}_j^i + \tilde{U}_{O_i}^i \tilde{\ddot{s}}_{0,j}^i z_j^i \end{aligned}$$

b) for $j > i$,

$$\frac{\partial \eta_i^i}{\partial q_j} = \sum_{k=i}^{j-1} {}^iW_k \left[\frac{\partial \mu_k^k}{\partial q_j} + \tilde{u}_{o_k}^k \frac{\partial \ddot{s}_{0,k}^k}{\partial q_j} + \tilde{s}_{k,k+1}^k \frac{\partial \ddot{U}_{o_{k+1}}^k}{\partial q_j}\right] + \frac{\partial {}^iW_j}{\partial q_j}\eta_j^j + {}^iW_j \frac{\partial \eta_j^j}{\partial q_j}$$

$$= {}^iW_j \left[- \tilde{s}_{i,j}^j \tilde{U}_{o_j}^j \ddot{z}_j^j - 2\tilde{s}_{i,j}^j \dot{\tilde{U}}_{o_j}^j \dot{z}_j^j - \tilde{s}_{i,j}^j \ddot{\tilde{U}}_{o_j}^j z_j^j\right]$$

$$+ {}^iW_j \tilde{z}_j^j \eta_j^j + {}^iW_j \left[E_{o_j}^j \ddot{z}_j^j - 2L_j^j \dot{z}_j^j + \tilde{U}_{o_j}^j \ddot{\tilde{s}}_{0,j}^j z_j^j\right]$$

$$= {}^iW_j \left(\left[E_{o_j}^j - \tilde{s}_{i,j}^j \tilde{U}_{o_j}^j\right]\ddot{z}_j^j - 2\left[L_j^j + \tilde{s}_{i,j}^j \dot{\tilde{U}}_{o_j}^j\right]\dot{z}_j^j + \left[\tilde{U}_{o_j}^j \ddot{\tilde{s}}_{0,j}^j - \tilde{s}_{i,j}^j \ddot{\tilde{U}}_{o_j}^j - \tilde{\eta}_j^j\right] z_j^j \right)$$

and this complete the proof of Lemma C-12 □.