DISTRIBUTED DYNAMICS OF SYSTEMS WITH CLOSED KINEMATIC CHAINS

Waseem A. Khan

Department of Mechanical Engineering
McGill University, Montréal

November 2002

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Master of Engineering

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisisitons et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-88363-9
Our file Notre référence
ISBN: 0-612-88363-9

Abstract

The simulation of mathematical models of mechanical systems with closed kinematic chains involves the solution to a system of highly coupled differential-algebraic equations. The *numerical* stiffness of these systems calls for small time steps in order to insure accuracy. *Real-time* and *interactive* forward simulations tend to be difficult to achieve for such systems, especially for large multi-body systems with multiple links and many kinematic loops. One way to overcome the time constraint is to *distribute* the load onto several processors.

The *modular* formulation of mathematical models is attractive because existing models may be assembled to create different topologies, e.g. cooperative robotic systems. Conversely, a given robotic topology may be broken into smaller topologies with simpler dynamics.

Moreover, parallel-kinematics machines bear inherent *spatial* parallelism. This feature is exploited in this thesis, in which we examine the formulation of such *modular* and *distributed* models and evaluate their performance as applied to mechanical systems with closed kinematic chains. Three general *undistributed* formulation methods are specialized to cope with *distribution* and *modularity* and applied to a three-degree-of-freedom planar parallel manipulator to generate distributed dynamics models.

Finally, the results of case studies are reported, and a comparison is made to highlight the salient features of each method.

Résumé

La simulation des modèles mathématiques de systèmes mécaniques avec des boucles cinématiques implique la solution des systèmes d'équations différentielles et algébriques fortement couplés. La raideur *numérique* de ces systèmes demande des pas d'intégration très petits afin d'assurer la précision du calcul. La simulation des modèles dits directs en *temps-réel* et *interactive* s'avère difficile pour de tels systèmes, particulièrement pour de grands systèmes multi-corps à plusieurs boucles fermées. Une possibilité pour surmonter la contrainte de temps est de *distribuer* la charge sur plusieurs processeurs.

La formulation *modulaire* des modèles dynamiques est intéressante parce que des modèles dynamiques existants peuvent être assemblés pour créer des topologies différentes, par exemple les systèmes de robots coopératifs. A l'inverse, une topologie robotique peut être décomposée en topologies simples avec des modèles dynamiques plus simples.

Par ailleurs, les machines, dites à cinématique paralèlle comportent un paralélisme *spatial* inhérent. Ceci est exploité dans ce mémoire, dans lequel nous étudions la formulation de tels modèles *modulaires* et *distribués*, tout en évaluant leur performance vis-à-vis des systèmes mécaniques avec des chaînes cinématiques fermées. Trois méthodes générales *non distribuées* de formulation sont spécialisées pour la *distribution* et la *modularité*, et appliquées à un manipulateur à trois degrés de liberté et à architecture paralèle plane, pour en produire des modèles distribués de dynamique.

Enfin, les résultats de nos simulations sont rapportés, tout en faisant une comparaison de ces méthodes.

Acknowledgements

I would like to express my deep gratitude to my supervisor Professor Venkat Krovi, who gave me a chance to return to graduate studies after six years out of school. His patience, guidance and encouragement made this thesis possible.

I am thankful to Professor Jorge Angeles, who accepted to co-supervise me after Professor Krovi's departure from McGill. He has been my inspiration to make in-depth studies.

I am also grateful to Professor S.K. Saha, who introduced me to the Decoupled Natural Orthogonal Complements and helped me derive simulation models using it.

The support of Quebec's *Fonds pour la formation de chercheurs et l'aide à la recherche* is duly acknowledged.

Provision of a free-license for Rt-Lab by Opal-RT is also acknowledged.

Many thanks go to Messrs. Kourosh Parsa, and Khalid Al-Widyan for their support on LaTeX.

Finally, I would like to express my deepest gratitude to my wife and my family for their never-ending encouragement and support.

TABLE OF CONTENTS

Abstract i
Résumé ii
Acknowledgements iii

CHAPTER 1. Introduction 1
1.1. Literature Survey 3
1.1.1. Non-Recursive Newton-Lagrange Formulations 4
1.1.2. Recursive Newton-Euler Formulations 8
1.1.3. Distributed Forward Dynamic Simulation 10

CHAPTER 2. Distributed Dynamics Formulations for a 3 RRR Planar Parallel Mechanism 17
2.1. Subsystem Dynamics Modelling 17
2.2. Method I: The Penalty-Based Approach 20
2.3. Method II: The Loop-Closure Orthogonal Complement 22
2.4. Method III: The Recursive Decoupled Natural Orthogonal Complement 26

CHAPTER 3. The Recursive Decoupled Natural Orthogonal Complement . . 30
3.1. Forward Kinematics 30
3.2. Inverse Dynamics 40
3.3. Forward Dynamics 45

CHAPTER 4. Simulation Results 51
4.0.1. Computation Environment 51

4.0.2. Parameters and Initial Conditions 53
4.0.3. Inverse Dynamics 54
4.1. Penalty-Based Approach 55
4.2. Loop-Closure Orthogonal Complement 60
4.3. Recursive Decoupled Natural Orthogonal Complement 65

CHAPTER 5. Conclusions and Recommendations for Future Work 68
5.1. Conclusions 68
5.2. Future Work 70

BIBLIOGRAPHY 72

APPENDIX A. Alternative Methods to Obtain an Orthogonal Complement 78
A.1. Singular-Value Decomposition 78
A.2. QR factorization 79
A.3. Gaussian Triangularization 81

CHAPTER 1

Introduction

The principal motivation for this thesis comes from current interest in the cooperation of robot systems, in particular multi-arm or multiple mobile manipulator systems, to transport large common payloads. Instances of these systems are displayed in Fig. 1.1 and 1.2. Such systems bear inherent topological modularity, e.g., different parallel architectures result when different numbers of mobile manipulators are used to carry a common payload, as shown in Fig. 1.2 and 1.3. The development of a flexible and scalable framework for collaboration in such systems imposes corresponding requirements for modularity regarding their kinematic and dynamic analyses as well as their control.

In the last quarter-century, simulation tools have seen manyfold increases in terms of their usage in the design, analysis, parametric refinement and model-based control of a variety of multibody systems such as vehicles, heavy machinery, spacecraft and robots. Numerical simulation methods have taken a primary position in the simulation of such multibody systems because corresponding closed-form solutions are possible only for text-book type of systems. The specialized literature includes a number of books on the subject (Ascher and Petzold, 1998; García de Jalón and Bayo, 1994; Haug, 1989; Schiehlen, 1990b; Shabana, 2001), where a broad variety of formulations and computational methods are discussed.

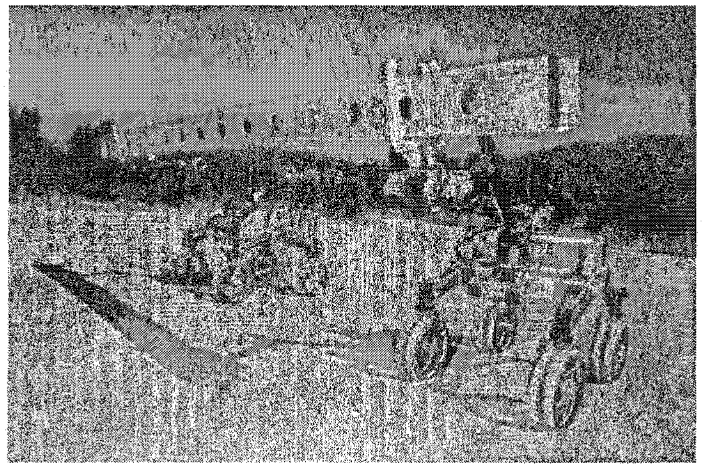
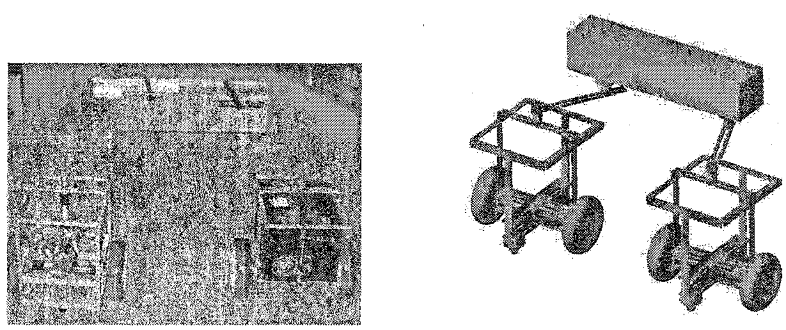


Figure 1.1: Two NASA robots carrying a metal beam (Sachdev, 2002).



(a) (b)

Figure 1.2: ARNOLD, A modular collaborating system of mobile manipulators (Abou-Samah, 2001): (a) physical prototype; (b) its CAD rendering.

While efficient formulations exist for serial-chain and tree-structured multibody systems, the adaptation of these methods to the simulation of closed-chain linkages and parallel manipulators is more difficult. Such systems possess one or more kinematic loops, requiring the introduction of algebraic (typically nonlinear) constraints into the formulation.

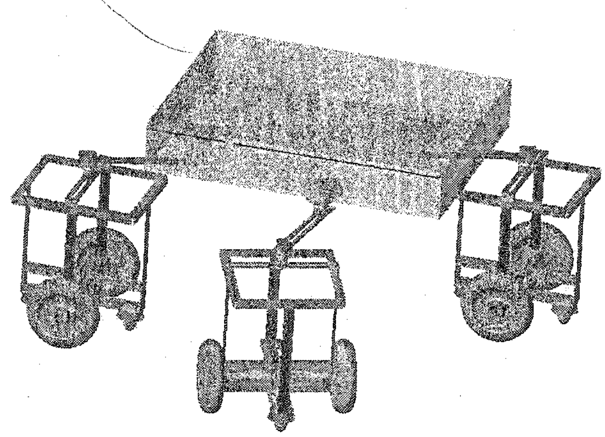


Figure 1.3: CAD rendering of a collaborating system of three mobile manipulators. (Abou-Samah, 2001).

In particular, we wish to focus on methods that permit us to modularly compose the dynamics model and subsequently distribute the computation back to the component subsystems. Such a redistribution of the computation is beneficial to achieve super-real-time simulations as well as model-based distributed control.

In this thesis, we examine both the *development* and *performance-evaluation* of different methods for the *modular* and *distributed* forward dynamics of constrained mechanical systems.

1.1. Literature Survey

The two principal problems associated with the dynamics of mechanical systems are *inverse* and *forward* dynamics. *Inverse dynamics* is defined as: Given the time-histories of all the system degrees-of-freedom, compute the time-histories of the controlling actuated joint torques and forces. The solution process is primarily an algebraic one and typically does not require the use of numerical integration methods, since the position coordinates, velocities and accelerations of the system are known.

Forward dynamics, in turn, is defined as: Given the time-histories of the actuated joint torques and forces, compute their time-histories of the joint coordinates, velocities and accelerations. In this case, the solution is obtained in a two-stage process. In the first stage the equations of motion (EOM) are solved algebraically to determine the accelerations. In the second stage, the underlying ordinary differential equations (ODE) are integrated to obtain all the joint-coordinate time histories. However, since closed-form solutions to such systems of nonlinear ordinary differential equations (ODEs) are out of question, one must resort to numerical integration methods.

Methods for formulation of the EOM fall into two main categories: a) Euler-Lagrange and b) Newton-Euler formulations. Typically, *Euler-Lagrange* formulations use joint-based relative coordinates as configuration-space variables; these formulations are generally not well suited for a recursive formulation. However, they are popular within the robotics community, since they use joint-based relative coordinates, which form a minimal-set for serial manipulators and have a direct meaning in robotics. *Newton-Euler* (NE) approaches, typically use Cartesian variables as configuration-space variables. They admit recursive formulations by first developing EOM for each single body; these equations are then assembled to obtain the model of the entire system.

Considerable work has been reported in the literature on the specialization of the above methods to formulate the EOM of constrained mechanical systems, while including both holonomic and non-holonomic constraints.

Parallel mechanisms and manipulators form a special class of constrained mechanical systems where the multiple kinematic loops give rise to systems of holonomic constraints.

In subsequent discussions we will focus on the development of EOM of constrained mechanical systems with multiple loops, exemplified by manipulators.

1.1.1. Non-Recursive Newton-Lagrange Formulations. The dynamics of constrained mechanical system with *closed loops* using a Newton-Lagrange approach is traditionally obtained by cutting the closed loops to obtain various open loops, also known as reduced systems, and then writing a system of ODEs for the corresponding chains in their corresponding generalized coordinates (Featherstone, 1987). The solution to these are required to satisfy additional algebraic equations which typically are constraint equations required to close the cut-open loops. A *Lagrange multiplier* term is introduced to represent the forces in the direction of the constraint violation. The resulting formulation, often referred to as a *descriptor form*, yields an often simpler, albeit larger, system of index-3 differential algebraic equations (DAEs) as follows[1]:

$$
\begin{aligned}
\mathbf{I}(\mathbf{q})\ddot{\mathbf{q}} &= \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t, \mathbf{u}) - \mathbf{A}(\mathbf{q})^T \boldsymbol{\lambda} && (1.1)\\
\mathbf{c}(\mathbf{q}, t) &= \mathbf{0} && (1.2)
\end{aligned}
$$

where $\mathbf{q}$ and $\dot{\mathbf{q}}$ are, correspondingly, the n-dimensional vectors of generalized coordinates and generalized velocities, $\mathbf{I}(\mathbf{q})$ is the $n \times n$ generalized inertia matrix, $\mathbf{c}(\mathbf{q}, t)$ is the m-dimensional vector of holonomic constraints, $\boldsymbol{\lambda}$ is the m-dimensional vector of Lagrange multipliers, $\mathbf{A}(\mathbf{q})$ is the $m \times n$ constraint Jacobian matrix, $\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t, \mathbf{u})$ is the n-dimensional vector of external forces and velocity-dependent inertial terms, while $\mathbf{u}$ is the vector of actuator forces or torques.

The solution of a system of index-3 DAEs by direct finite difference discretization is not possible using explicit discretization methods (Ascher and Petzold, 1998). Instead, the above system is typically converted to a system of ODEs and expressed in state-space form, which may be integrated using standard numerical code. Typical methods used to achieve this end are discussed below.

1.1.1.1. *Direct Elimination.* The surplus variables are eliminated directly, using the *position-level algebraic constraints* to explicitly reduce index-3 DAE to an ODE in a minimal set of generalized coordinates (conversion into Lagrange's equations of the second kind). This is also referred to as a *closed-form solution* of the constraint equations. The resulting minimal order ODE can then be integrated without concern about stability issues. However, such a reduction cannot be done in general,

[1]The differential index is defined as the number of times the DAE has to be differentiated to obtain a standard set of ODE.

and even when it can, the differential equations obtained, are typically complicated (Kecskemethy *et al.*, 1997).

1.1.1.2. *Lagrange-Multiplier Computation.* All the algebraic position-level and velocity-level constraints are differentiated and represented at the acceleration level, to obtain an augmented index-1 DAE, in terms of both unknown accelerations and unknown multipliers (Ascher and Petzold, 1998; Murray *et al.*, 1994) as:

$$\begin{bmatrix} \mathrm{I}(\mathrm{q}) & \mathrm{A}^T \\ \mathrm{A} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathrm{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ -\dot{\mathrm{A}}(\mathrm{q})\dot{\mathrm{q}} - \ddot{\mathrm{c}} \end{bmatrix} \tag{1.3}$$

which may be solved for $\ddot{\mathrm{q}}$ and λ. By selecting the state of the system to be $\mathbf{x} = \begin{bmatrix} \mathrm{q}^T & \dot{\mathrm{q}}^T \end{bmatrix}^T$ the above set of equations may be converted into the standard state-space form, which may then be integrated using standard code. The advantage is the conceptual simplicity and simultaneous determination of the accelerations and Lagrange multipliers by solving a *linear* system of equations. However, this system needs more initial conditions than the original system to specify a unique solution. The system is also known to entail stability problems.

1.1.1.3. *Lagrange-Multiplier Approximation–Penalty Formulation.* In this approach the loop-closure constraints are *relaxed* and replaced using virtual springs and dampers (Wang *et al.*, 2000). Using such virtual springs can be considered as a form of penalty formulation (García de Jalón and Bayo, 1994), which incorporates the constraint equations as a dynamical system penalized by a large factor. The Lagrange multipliers are *estimated* using a compliance-based force-law. The latter is based on the extent of constraint violation and an assumed spring stiffness; the force is then eliminated from the list of $n+m$ unknowns, leaving behind a system of $2n$ first-order ODEs. While the sole initial drawback may appear to be restricted to the numerical ill-conditioning due to the selection of large penalty factors, it is important to note that penalty approaches only approximate the true constraint forces and can create unanticipated problems.

1.1.1.4. *Dynamic Projection onto the Tangent Space.* These methods seek to take the constraint-reaction-containing dynamics equations into the *orthogonal* and *tangent* subspaces of the vector space of the system generalized velocities. Let $\mathrm{S(q)}$ be a $n \times (n-m)$ full-rank matrix whose column space lies in the nullspace of $\mathbf{A}(\mathrm{q})$, i.e. $\mathbf{A}(\mathrm{q})\mathbf{S}(\mathrm{q}) = \mathbf{0}$. The orthogonal subspace is spanned by the so-called constraint vectors, those forming the rows of the matrix $\mathbf{A}(\mathrm{q})$, while the tangent subspace *complements* this orthogonal subspace in the overall generalized velocity vector space. All *feasible* dependent velocities $\dot{\mathrm{q}}$ of a constrained multibody system necessarily belong to this tangent space, appropriately called the *space of feasible motions*. This space is spanned by the columns of $\mathbf{S}(\mathrm{q})$ and is parameterized by a $n-m$ dimensional vector of independent velocities, $\boldsymbol{v}(t)$ yielding the expression for the feasible dependent velocities as $\dot{\mathrm{q}} = \mathbf{S}(\mathrm{q})\boldsymbol{v}(t)$. A family of choices exist for the selection of dependent and independent velocities, each choice giving rise to a different $\mathbf{S}(\mathrm{q})$.

A popular choice of *Coordinate-Partitioning* (Shabana, 2001) in which the generalized velocity is partitioned into dependent $\dot{\mathrm{q}}_D$ and independent $\dot{\mathrm{q}}_I$ velocities, with $\boldsymbol{v}(t) = \dot{\mathrm{q}}_I$ i.e.

$$\dot{\mathrm{q}} = \begin{bmatrix} \dot{\mathrm{q}}_D \\ \dot{\mathrm{q}}_I \end{bmatrix} \tag{1.4}$$

By selecting $\boldsymbol{v}(t) = \dot{\mathrm{q}}_I$ and solving the linear velocity constraints, a relation between $\dot{\mathrm{q}}_D$ and $\dot{\mathrm{q}}_I$ is then obtained as:

$$\dot{\mathrm{q}}_D = \mathbf{K}\dot{\mathrm{q}}_I \tag{1.5}$$

which leads to a special form of $\mathbf{S}(\mathrm{q})$, denoted by $\mathbf{T}$, namely,

$$\dot{\mathrm{q}} = \begin{bmatrix} \mathrm{K} \\ 1 \end{bmatrix} \dot{\mathrm{q}}_I = \mathbf{T}\dot{\mathrm{q}}_I \tag{1.6}$$

Various methods for the numerical computation of $\mathbf{T}$ exist. Some of these are reviewed by García de Jalón and Bayo (1994); a short review is included in the Appendix. The $n \times (n-m)$ matrix $\mathbf{T}$ lies in the nullspace of $\mathbf{A}$, i.e $\mathbf{AT} = \mathbf{O}$, where $\mathbf{O}$ is the

$m \times (n-m)$ zero matrix. $\mathbf{T}$ is usually called the *loop-closure orthogonal complement.* Pre-multiplying both sides of eq.(1.1) by $\mathbf{T}^T$ we obtain a constraint-free second order ODE as

$$\mathbf{T}^T\mathbf{I}(\mathrm{q})\ddot{\mathrm{q}} = \mathbf{T}^T\mathbf{f}(\mathrm{q}, \dot{\mathrm{q}}, t, \mathbf{u}) \tag{1.7}$$

The same result using virtual displacement $\delta\mathrm{q}$ may also be obtained using variational principles of virtual work. The above system of equation is still underdetermined, but may be successfully used, as is, for inverse dynamics.

For forward dynamics, an approach known as the *Embedding Technique* (Shabana, 2001) is commonly used, where eq.(1.6) is embed in eq.(1.7). Differentiating eq.(1.6) once with respect to time and substituting the expression thus resulting into eq.(1.7) we obtain

$$\mathbf{T}^T\mathbf{I}(\mathrm{q})\mathbf{T}\ddot{\mathrm{q}}_I + \mathbf{T}^T\mathbf{I}(\mathrm{q})\dot{\mathbf{T}}\dot{\mathrm{q}}_I = \mathbf{T}^T\mathbf{f}(\mathrm{q}, \dot{\mathrm{q}}, t, \mathbf{u}) \tag{1.8}$$

which is the minimal-order ODE sought and can be integrated with suitable ODE solvers. The state vector is $\mathbf{x} = [\mathrm{q}^T, \dot{\mathrm{q}}_I^T]^T$, which is of dimension $2n - m$.

1.1.2. Recursive Newton-Euler Formulations. Dynamics equations based on classic Lagrange approaches are of the order $O(N^4)$ (Featherstone, 1987), which means that the number of floating point operations grow with the fourth power of the number of bodies in the system. Many variants of fast and readily-implementable recursive algorithms have been formulated within the last two decades, principally within the robotics community. As with non-recursive algorithms, the development of recursive algorithms started with the development of algorithms for inverse dynamics. The first researchers to develop $O(N)$ algorithms for inverse dynamics for robotics used a Newton-Euler formulation of the problem. Stepanenko and Vukobratovic (1976) developed a recursive NE method for human limb dynamics, and Orin *et al.* (1979) made the recursive method more efficient by referring forces and moments to local link-coordinates for the real-time control of a leg of a walking machine. Luh *et al.* (1980) developed a very efficient recursive Newton-Euler algorithm (RNEA) by referring most quantities to link-coordinates. However, the algorithm reported by Luh *et al.* (1980) is the most often cited. Further gains have been made in the efficiency over the years, as reported, for example, by Balafoutis *et al.* (1988) and Goldenberg and He (1989).

The earliest $O(N)$ algorithm for *forward dynamics* was developed by Vereshchagin (1974) who used a recursive formulation to evaluate the Gibbs-Appel form of the equations of motion and is applicable to un-branched chains with revolute and prismatic joints. Next, Armstrong (1979) developed an $O(N)$ algorithm for mechanisms with spherical joints. Later, Orin and Walker (1982) used RNEA for inverse dynamics as the basis for an efficient recursive forward dynamics algorithm. This method is commonly referred to as the *composite-rigid-body algorithm* (CRBA). This algorithm needed to solve a linear system of equations whose dimension grows with the number of rigid bodies. Since methods to solve a linear system of N equations in the N unknowns are $O(N^3)$, this algorithm is also $O(N^3)$. However, for small N, the first-order terms dominate the computation, so that the algorithm is quite efficient. So far, the composite inertia method is perhaps the most efficient general-purpose algorithm for serial manipulators with $N < 10$, which includes most practical cases. Next, Featherstone (1983) developed what he called the *articulated-body algorithm* (ABA), which was followed by a more elaborate and faster model (Featherstone, 1987). The computational complexity of ABA is $O(N)$ and is more efficient than CRBA for $N > 9$. Further gains have been made in efficiency over the years, as reported by Brandl *et al.* (1986) and McMillan and Orin (1995).

In multi-loop mechanisms the joint variables are no longer independent, since they are subject to loop-closure constraints, which are usually nonlinear. The existing literature on recursive algorithms applied to multi-loop mechanisms almost always uses a non-minimal set of generalized coordinates (Bae and Haug, 1987; Schiehlen, 1990a; Stejskal and Valasek, 1996; Bae and Han, 1999; Featherstone, 1999). The most common method for dealing with kinematics is to cut the loop, introduce Lagrange multipliers to substitute for the cut joints and use a recursive scheme for the open-chain system to obtain a recursive algorithm. However, the inclusion of *Lagrange multipliers*, as previously discussed, raises stability issues.

Recently a global method was proposed by Saha and Schiehlen (2001), which does not cut the kinematic loops. This method uses a minimal set of generalized coordinates. With some modifications, this method will be applied to a three-degree-of-freedon planar parallel manipulator in this thesis.

1.1.3. Distributed Forward Dynamic Simulation. [2]As discussed in the previous sections, the simulation process involves the time-discretized numerical solution of an initial-value problem (IVP), using a variety of numerical time-stepping schemes. In particular, the *numerical stiffness* of the underlying coupled differential-algebraic equations necessitates a large number of small time-steps in order to ensure a prescribed accuracy. Hence, while *real-time* and *interactive* simulations of complex assemblies are desirable from a design view point, they tend to be difficult to achieve for large multi-body systems with multiple links and many kinematic loops using conventional processing paradigms. One method to achieve speed-ups in such computations and to satisfy real-time constraints is to *distribute* the computational load onto several processors running in parallel. Henrich and Höniger (1997) gave a brief review and a preliminary classification of the different levels of distribution that have been explored in the context of robotic applications and noted that distribution at all levels may not be possible. Results obtained by distributed algorithms vary depending on the degree of dependency and coupling among the equations. While image-processing problems (Chaudhry and Aggarwal, 1990) can be broken down quite well by dividing the image into smaller independent blocks, the problems of simulation of constrained mechanical systems is a strongly coupled problem and the task is not trivially distributable (Fijani and Bejczy, 1992; Zoyama, 1993).

[2]For brevity, we use the word "distributed" instead of "parallel" when discussing parallelization of algorithms, in order to differentiate these from algorithms for parallel manipulators. However, we will continue to use the word "parallel" when referring to hardware architecture.

In what follows, we will discuss some aspects of these levels of distribution as applicable to the simulation of robotic systems, and specifically to closed-chain systems.

1.1.3.1. *Distribution Levels for Multi-Body Dynamic Simulation.* At the outset, we note that the nature of the selected computational architecture/infrastructure, such as shared-memory architectures vs. distributed-memory architectures, play a critical role in the implementation of the distribution. Factors such as *latency, throughput, cost* and *modularity* can vary significantly based on the architecture choices and affect the overall implementation of the distributed computation. A detailed discussion of these issues is beyond the scope of this thesis; the reader is referred to number of books on distributed implementations and high-performance computing for further details (Roosta, 2000).

Classifications of distribution methods have been proposed based on a combination of: (i) computational/algorithmic parallelism and (ii) the natural spatial parallelism within multiple rigid bodies and/or articulated sub-chains, within a mechanical system. A gradation of these methods is also possible based on levels of granularity, ranging from fine-grain to extremely coarse-grain. However, we note that most methods for distribution of dynamics simulations usually combine one or more levels of distribution.

1.1.3.2. *Fine-Grain/Link-Level Distribution.* The primary motivation behind the development of such distribution methods, beginning in the mid-eighties, was the desire to speed up the computation of *serial-chain manipulators* to satisfy real-time constraints, not necessarily with modularity in mind.

Lee and Chang (1986) first presented a distributed algorithm for inverse dynamics computation for serial-chain robots by reformulating the recursive Newton-Euler algorithm in a linear homogenous recurrence and utilizing "recursive-doubling" techniques (Kogge, 1974; Kogge and Stone, 1973) to compute the joint torques. Subsequently, Lee and Chang (1988) proposed a distibuted forward dynamics simulation algorithm for *serial-chain* manipulators, where the distribution was applied to the recursive composite-rigid-body algorithm. However, a complex interconnection network or a specialized processor array was required for the implementation. Most of the ensuing work continued to focus on fine-grain parallel algorithms for implementation on special-purpose computational architectures. (Sadayappan *et al.*, 1989; Fijany and Bejczy, 1991; Fijani and Bejczy, 1992).

Fijani and Bejczy (1992) survey many methods developed for distribution of dynamics algorithms for serial-chain manipulators, both at the computation level and at the natural body level. In conclusion, they note that:

- $O(N)$ recursive algorithms, e.g., the one reported in (Featherstone, 1987), are strictly serial and lead to first-order nonlinear recurrences which, regardless of the number of microprocessors employed, can be speeded up only by a constant factor.
- $O(N^3)$ algorithms, such as the one reported in (Orin and Walker, 1982), provide the highest degree of parallelism. Fijani and Bejczy (1992) parallelized the $O(N^3)$ algorithm, using a two-dimensional array of $O(N^2)$ processors to achieve $O(N)$ performance.

Fijani *et al.* (1995) are credited for the first distributed forward dynamics algorithm called the *constraint-force algorithm* (CFA) for *serial/parallel* manipulators with $O(\log(N))$ complexity of computation on $O(N)$ processors. An improved form of this, where all restrictions to type of kinematic chains and classes of joints were removed, appeared in (Featherstone and Fijani, 1999). The algorithm is in full-descriptor form and works by dividing the mechanisms into sub-chains, obtaining a sparse system of linear equations for the unknown inter-body constraint forces. This system is then solved by various iterative parallel methods. The constraint forces are then used to determine state-derivatives that are time-integrated to obtain updated values for the system state. The main disadvantage to this method is the utilization of iterative methods and the use of the full descriptor form, which, as already discussed, is not stable.

In contrast, the *divide-and-conquer articulated-body algorithm* (DCA) by Featherstone (1999) with $O(\log(N))$ time complexity on $O(N)$ processors is the fastest available algorithm for a computer with a large number of processors and low communication cost.

It requires a node parallel-processing architecture and uses a recursive binary assembly of the articulated-body equations of motion of an assembly from those of its constituents, as shown in Fig. 1.4. Each subassembly is assumed to possess two handles and the handles of adjacent subassemblies are joined to each other at each assembly stage. Thus, for unbranched mechanisms, two adjacent handles are removed and replaced by a joint using acceleration-level kinematic constraints and force constraints. The result is a parent assembly with two handles, each handle inherited from its corresponding child assembly with the entire chain assembled in a recursive binary way. The last step of coupling the chain to the base with known kinematic entries creates the final joint and produces a determinate system. Thus, it is now possible to solve for the acceleration of the final joint, and subsequently to recursively disassemble the assembly and determine joint accelerations as the dis-assembly proceeds. The procedure may be extended to parallel mechanisms by cutting the platform into sub-links and defining a rigid joint between them. Simultaneously, the base of the platform is taken as a floating body and is cut into sub-links with rigid joints. Finally the floating base is connected to the ground with a rigid joint.

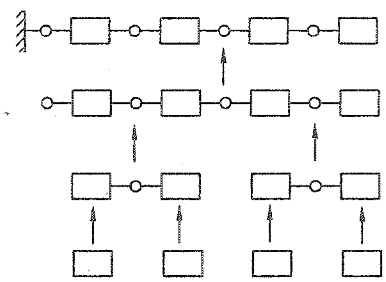


Figure 1.4: Recursive binary assembly of a four-link mechanism. Circles are joints and rectangles are links.

However, the principal drawback of this approach is that the algorithm is implemented with non-minimal coordinates and tends to be mildly unstable. While modifications, including Baumgarte stabilization have been suggested to address this stability issue, they tend to be difficult to apply to systems with kinematic loops.

The Hybrid Direct/Iterative Algorithm (HDIA) proposed by Anderson and Duan (2000) is an iterative algorithm and works by cutting a rigid-body system into just sufficient separate pieces to allow for full use of all the processors on a given parallel computer. The equations of the separate pieces are evaluated in parallel, and the results are loaded into a single system-wide equation to calculate the constraint forces acting between the pieces due to the cut joints. This matrix has dimensions that depend on the number of cut joints, rather than the number of bodies, and is typically sparse, enabling parallel iterative solution techniques to be used effectively. Apart from this one matrix equation, the total cost of the rest of the algorithm is $O(\log N)$. HDIA expresses its equations of motion in minimal coordinates using coordinate-partitioning, which is an advantage. However, again the iterative solution techniques employed are the major draw back.

1.1.3.3. *Integration Level Distribution.* In this approach, the distribution is applied to time stepping processes, instead of the dynamics calculations resulting in *temporal parallelism.* Among many different methods to perform numerical integration, predictor-corrector two-step methods are readily parallelizable (McMillan *et al.*, 1994; Birta and Abou-Rabia, 1987). McMillan *et al.* (1994) proposed a x-point sliding-block method, which they called BxPC5, which utilizes fifth-order predictor-corrector method, where x is the number of processors which must lie between one and four.

1.1.3.4. *Chain-to-Chain Level Distribution.* The two foregoing levels of distribution require special parallel computing architectures, typically with large number of processors and/or high inter-processor communication bandwidth; as well, they require some form of load-balancing.

Hence, many of these approaches are much less efficient when implemented on general-purpose parallel and multiprocessing systems such as distributed-memory cluster computing machines, e.g. Beowulf systems, that are gaining popularity. In

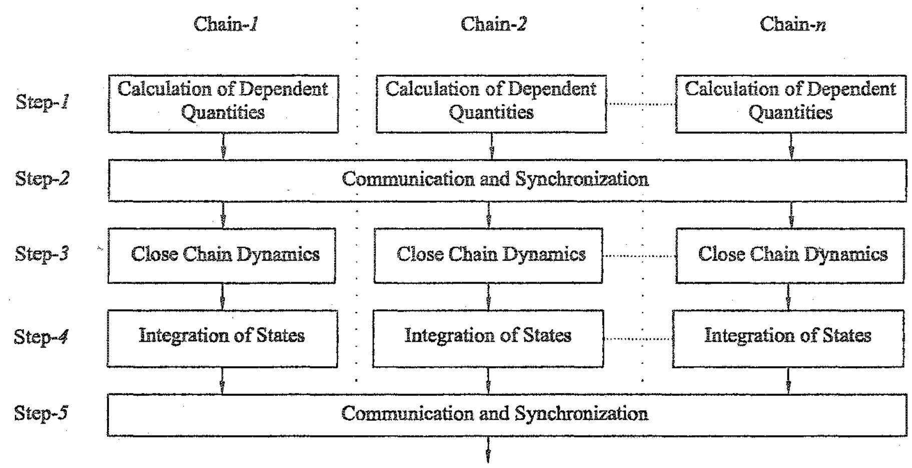


Figure 1.5: Minimal-order distributed forward dynamics model.

particular, such systems typically have a limited number of loosely coupled processors that are not designed to efficiently handle a large amount of inter-processor communication and synchronization.

Hence, for *medium to low-dof multi-loop parallel manipulators simulated on such general purpose distributed-memory systems with limited number of processors*, we look for methods which could efficiently calculate dynamics equations for small groups of links (sub-chains) of the manipulator, rather than individual links or joints. These approaches have distinct advantages over the fine-grained approaches because less time is required to develop the algorithm and no special hardware development is needed (McMillan *et al.*, 1994).

A distributed forward dynamic model based on *minimal coordinates* is shown Fig. 1.5. The communication step (step-2) is required owing to the coupling of the chains. Depending on the modelling technique adopted, more that one such steps may be required. Additionally, steps $1-3$ are repeated, where the number of repetitions depend on the time stepping and on the integration scheme, e.g., a 4th order explicit Runge-Kutta scheme would require four EOM evaluations.

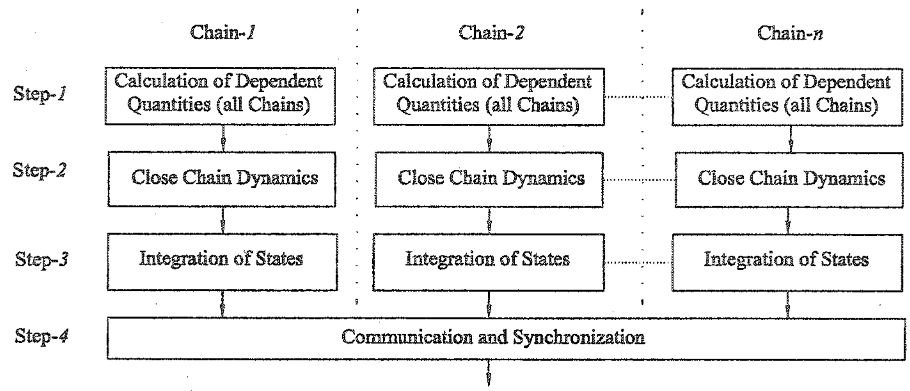


Figure 1.6: Minimal-order distributed forward dynamics model with repeated computation.

We define, a *fully distributed* model as one that would need only one communication step during *each time-step.* i.e. no communication between EOM evaluations. A fully distributed model poses minimum communication load to the distributed system and is thus attractive for general-purpose distributed memory-cluster computing networks.

A variation to the distributed model in Fig. 1.5 is shown in Fig. 1.6, where the dependent calculations of each chain are *embedded* in the EOM evaluation of each chain. Here we achieve full distribution at the expense of extra repeated computations in each chain. However, such variations are not possible for every models and modelling techniques.

CHAPTER 2

Distributed Dynamics Formulations for a 3 <u>R</u>RR Planar Parallel Mechanism

In this chapter we develop alternative formulations for the distributed forward dynamics of a 3 <u>R</u>RR planar parallel mechanism. Many approaches have been proposed in the past for numerically simulating the forward dynamics in non-distributed form. We focus on three non-distributed approaches which show the greatest promise for the re-distribution of the actual computational load back to the individual subsystems. These are:

- The penalty-based approach.
- The loop-closure orthogonal complement approach.
- The recursive decoupled natural orthogonal complement approach

2.1. Subsystem Dynamics Modelling

For the sake of simplicity we restrict ourselves to a 3 <u>R</u>RR planar parallel mechanism that has: *i*) only revolute joints, *ii*) identical legs and *iii*) a moving platform in the shape of an equilateral triangle, as sketched in Fig. 2.1. The three-degree-of-freedom (three-dof) planar manipulator consists of three identical dyads, numbered I, II and III coupling the platform $\mathcal{P}$ with the base $\mathcal{B}$. The three dyads have their fixed pivots O^I, O^{II} and O^{III} on the vertices of an equilateral triangle. Each dyad,

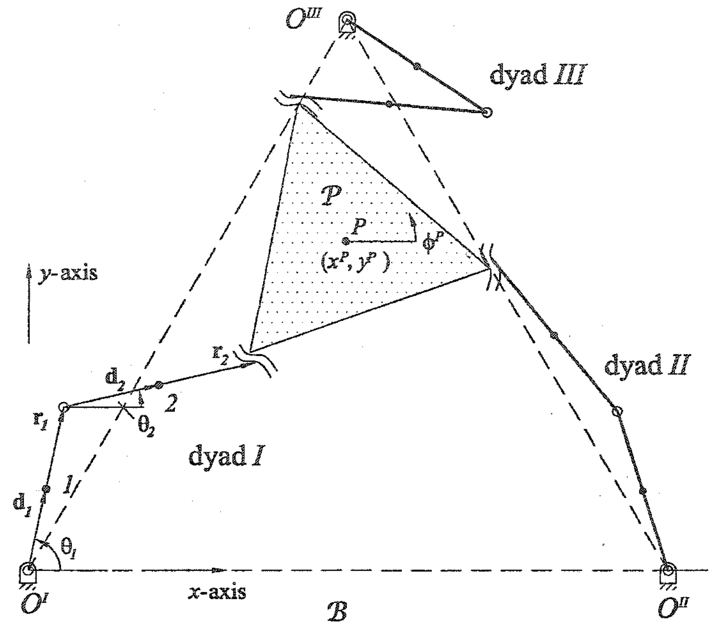


Figure 2.1: 3-dof, planar parallel manipulator.

moreover, has two links of lengths l_1 and l_2, numbered from proximal to distal. The mass of the links are, correspondingly, m_1 and m_2. The centroidal moment of inertia of each link about the axis normal to the xy-plane is I_i, for $i = 1, 2$. The mass of the platform is given by m_P, its mass centre located at P, the centroid of the equilateral triangle, and the centroidal moment of inertia about an axis equally oriented is I_P. We divide the mechanism into four parts i.e. dyad I, dyad II, dyad III and Platform $\mathcal{P}$, as shown in Fig. 2.1. Each dyad can be modelled as an open chain with two degrees of freedom. The dynamics of each dyad can be written for the three chains, for $i = I$, II and III, as follows:

$$\mathbf{I}^i(\mathbf{q}^i)\ddot{\mathbf{q}}^i + \mathbf{b}^i(\mathbf{q}^i, \dot{\mathbf{q}}^i) = \mathbf{u}^i \tag{2.1}$$

where

$$\mathbf{I}^i(\mathbf{q}^i) = \begin{bmatrix} I_1 + m_1 c_1^2 + m_2 l_1^2 & m_2 l_1 c_2 \cos(\theta_2 - \theta_1) \\ m_2 l_1 c_2 \cos(\theta_2 - \theta_1) & I_2 + m_1 c_2^2 \end{bmatrix}_i$$

$$\mathbf{q}^i = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}^i ; \ \mathbf{u}^i = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}^i$$

$$\mathbf{b}^i(\mathbf{q}^i, \dot{\mathbf{q}}^i) = \begin{bmatrix} -m_2 l_1 c_2 \sin(\theta_2 - \theta_1)\dot{\theta}_2^2 - m_1 g c_1 \cos\theta_1 - m_2 g l_1 \cos\theta_1 \\ m_2 l_1 c_2 \sin(\theta_2 - \theta_1)\dot{\theta}_1^{\,2} - m_2 g c_2 \cos\theta_2 \end{bmatrix}_i$$

where g is the acceleration due to gravity and is assumed to act in the $-y$ direction, c_k, for $k = 1, 2$, being the distance of the mass centre from the corresponding joint-axis. The Newton-Euler equations of motion of the free platform can be written as follows:

$$\mathbf{M}^P \dot{\mathbf{t}}^P = \mathbf{w}^P \tag{2.2}$$

where

$$\mathbf{t}^P \equiv \begin{bmatrix} \dot{\phi}^P \\ \dot{x}^P \\ \dot{y}^P \end{bmatrix} : \text{platform twist;} \ \mathbf{w}^P \equiv \begin{bmatrix} f_x^P \\ f_y^P \\ \tau^P \end{bmatrix} : \text{applied wrench}$$

$$\mathbf{M}^P \equiv \begin{bmatrix} I_P & 0 & 0 \\ 0 & m_P & 0 \\ 0 & 0 & m_P \end{bmatrix} : \text{mass matrix of the platform}$$

The constraint equations relating the four members of the cut mechanism are given by

$$\mathbf{c}^i \equiv \mathbf{a}_0^i + \mathbf{a}_1^i + \mathbf{a}_2^i + \mathbf{a}_3^i - \mathbf{p} = \mathbf{0} \tag{2.3}$$

for $i = I, II$ and III. Vector $\mathbf{a}_0^i$ is the position vector of O^i while $\mathbf{a}_1^i$, $\mathbf{a}_2^i$ and $\mathbf{a}_3^i$ are correspondingly the position vectors of joint-2 with respect to joint-1, joint-3 with respect to joint-2 and mass centre P with respect to joint-3 of dyad i. Vector $\mathbf{p}$ is defined as the position vector of point P of the platform $\mathcal{P}$. Letting $\mathbf{q} = [(\mathbf{q}^I)^T, (\mathbf{q}^{II})^T, (\mathbf{q}^{III})^T]^T$ and $\mathbf{c} = [(\mathbf{c}^I)^T, (\mathbf{c}^{II})^T, (\mathbf{c}^{III})^T]^T$, we obtain:

$$\mathbf{c}(\mathbf{q}) = \mathbf{0} \tag{2.4}$$

The Euler-Lagrange equations for the dynamics of a constrained mechanical system with component dynamics constrained by such loop-closure constraints can be written as a system of index-3 DAEs as:

$$\begin{aligned} \mathbf{I}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) &= \mathbf{u} - \mathbf{A}^T\boldsymbol{\lambda} &\quad (2.5)\\ \mathbf{c}(\mathbf{q}) &= \mathbf{0} &\quad (2.6)\end{aligned}$$

with

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^I \\ \mathbf{q}^{II} \\ \mathbf{q}^{III} \\ \mathbf{q}^P \end{bmatrix};\ \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{b}^I \\ \mathbf{b}^{II} \\ \mathbf{b}^{III} \\ \mathbf{0} \end{bmatrix};\ \mathbf{u} = \begin{bmatrix} \mathbf{u}^I \\ \mathbf{u}^{II} \\ \mathbf{u}^{III} \\ \mathbf{w}^P \end{bmatrix};\ \mathbf{A} = \frac{\partial \mathbf{C}(\mathbf{q})}{\partial \mathbf{q}}$$

$$\mathbf{I}(\mathbf{q}) = \begin{bmatrix} \mathbf{I}^I & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}^{II} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{I}^{III} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{M}^P \end{bmatrix}$$

where $\mathbf{q}^P = [\phi^P\ x^P\ y^P]^T$. In particular, the constraints are incorporated by differentiating the position-level (holonomic) constraints to obtain the velocity-level constraint Jacobian $\mathbf{A}$ and adjoined with Lagrange multipliers to obtain the constrained dynamics. The rows of the constraint Jacobian or correspondingly the columns of $\mathbf{A}^T$, span the feasible motion directions, while the Lagrange multipliers, grouped in vector $\boldsymbol{\lambda}$, correspond to the magnitude of the required constraint correction forces.

2.2. Method I: The Penalty-Based Approach

The Lagrange multiplier λ_i will be zero only if the position-level constraint $c_i(\mathbf{q})$ and the velocity-level constraint $\dot{c}_i(\mathbf{q})$ are both zero. In all other cases, a restoring force proportional to the extent of the constraint violation will appear. In this class of approaches, the Lagrange multipliers are approximated by a virtual spring and damper that produces a restoring force that is proportional to the extent of the constraint violation. This may be expressed as $\lambda_i = k_i c_i(\mathbf{q}) + d_i \dot{c}_i(\mathbf{q})$, where k_i is the spring constant, d_i is the damping constant and c_i is the extent of the constraint violation.

The net effect of this approximation is to eliminate the algebraic relationship from the index-1 DAE system permitting it to be reduced to a system of ordinary differential equations that can be integrated using standard numerical code.

Substituting the value of $\boldsymbol{\lambda}$ into eq.(2.5) one can solve for $\ddot{\mathbf{q}}$, the DAE system (2.5 & 2.6) thus reducing to an ODE system, which can be integrated by means of standard numerical methods. By defining the state $\mathbf{x} = [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T$ the extended-set of $2n$ equations of motion in state space form may be written as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{I}^{-1}(\mathbf{q})[\mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{b} - \mathbf{A}^T(\mathbf{K}\mathbf{c}(\mathbf{q}) + \mathbf{D}\dot{\mathbf{c}})] \end{bmatrix} \tag{2.7}$$

where

$$\mathbf{K} = \mathrm{diag}(k_i) \text{ and } \mathbf{D} = \mathrm{diag}(d_i)$$

for $i = 1 \dots m$. For our system, the constraint vector $\mathbf{c}(\mathbf{q})$ is six-dimensional, while the state vector $\mathbf{q}$ is nine-dimensional. Hence, the constraint Jacobian $\mathbf{A}$ is a 6×9 matrix, which is partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^I & \mathbf{A}^{II} & \mathbf{A}^{III} & \mathbf{A}^P \end{bmatrix} \tag{2.8}$$

where $\mathbf{A}^I$, $\mathbf{A}^{II}$ and $\mathbf{A}^{III}$ are all 6×2 matrices while $\mathbf{A}^P$ is a 6×3 matrix formed by selecting the last three columns of $\mathbf{A}$. With this partitioning, the overall dynamics of the system is now readily partitioned into components that can now be numerically integrated on the corresponding component subsystem. The overall distributed system is now given by

$$\dot{\mathbf{x}}^i = \begin{bmatrix} \dot{\mathbf{q}}^i \\ \ddot{\mathbf{q}}^i \end{bmatrix} = \begin{bmatrix} \mathbf{v}^i \\ (\mathbf{I}^i)^{-1}[\mathbf{u}^i - \mathbf{b}^i - \mathbf{A}^{iT}(k\mathbf{c} + d\dot{\mathbf{c}})] \end{bmatrix} \tag{2.9}$$

for $i = I,\ II$ and III, while, for the platform,

$$\dot{\mathrm{x}}^P = \begin{bmatrix} \dot{\mathrm{q}}^P \\ \ddot{\mathrm{q}}^P \end{bmatrix} = \begin{bmatrix} \mathrm{t}^P \\ (\mathrm{M}^P)^{-1}[\mathrm{w}^P - (\mathrm{A}^P)^T(k\mathrm{c} + d\dot{\mathrm{c}})] \end{bmatrix} \tag{2.10}$$

where a single spring constant k and damping factor d is used for simplicity thereby completing the model.

2.3. Method II: The Loop-Closure Orthogonal Complement

We will adopt the approach outlined in Yun and Sarkar (1998) and examine the possibility of distributing the computations of this approach. In this method, the configuration of the system is described by the full set of joint angles, the loop-closure constraints then being expressed in terms of these joint-space configuration variables. The differentiation of these constraints gives velocity-level constraint equations and the orthogonal complement, in the joint space, now provides a basis for the feasible joint velocities. We term the coordinate axes of these velocities the *feasible directions.* The dynamics equations are projected onto the instantaneous feasible directions, which are tangent to the constraint manifold; with an adequately small step size, the resulting integrated solution is guaranteed to obey the constraints. In particular, of the many variants possible, we adopt the *embedding technique* previously discussed. Selecting the state vector as $\begin{bmatrix}\mathrm{q}^T & \dot{\mathrm{q}}_i^T\end{bmatrix}^T$, the dynamics is expressed in state-space form as an extended set of $(2n\text{-}m)$ differential equations.

A key feature is the incorporation of stabilization. The differentiated constraints of the velocity and acceleration manifold are mildly unstable, thereby making troublesome the enforcement of constraints and satisfaction of initial conditions, which is known as the drift problem. Inspired by Baumgarte's method of stabilization, Yun and Sarkar (1998) approximate the holonomic constraints with those of a stable first-order system

$$\dot{\mathrm{c}}(\mathrm{q}) + \sigma \mathrm{c}(\mathrm{q}) = 0,\ \sigma > 0 \tag{2.11}$$

where σ is the rate of convergence and can be appropriately chosen based on the problem at hand. Thus, regardless of the selection of the initial conditions or the presence of disturbances, this first-order system exponentially stabilizes to the attracting equilibrium condition $\mathbf{c}(\mathbf{q}) = \mathbf{0}$, where the constraints are satisfied. Specifically, taking $\mathbf{A}(\mathbf{q})$ as the Jacobian of $\mathbf{c}(\mathbf{q})$, eq.(2.11) can be re-written as

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = -\sigma\mathbf{c}(\mathbf{q}) \tag{2.12}$$

Let $\mathbf{S}(\mathbf{q})$ be a $n \times (n-m)$ full-rank matrix, whose column space is the nullspace of $\mathbf{A}(\mathbf{q})$ i.e. $\mathbf{A}(\mathbf{q})\mathbf{S}(\mathbf{q}) = \mathbf{O}$, with $\mathbf{O}$ denoting the $m \times (n-m)$ zero matrix $\eta(\mathbf{q})$ being a particular solution of eq.(2.12) and $\boldsymbol{v}(t)$ any smooth $(n-m)$ dimensional vector. The general solution of eq.(2.12) is thus given by

$$\dot{\mathbf{q}} = \mathbf{v} = \mathbf{S}(\mathbf{q})\boldsymbol{v}(t) + \eta(\mathbf{q}) \tag{2.13}$$

Differentiating the above relation once we obtain:

$$\dot{\mathbf{v}} = \mathbf{S}(\mathbf{q})\dot{\boldsymbol{v}}(t) + \dot{\mathbf{S}}(\mathbf{q})\boldsymbol{v}(t) + \dot{\eta}(\mathbf{q}) \tag{2.14}$$

Or

$$\dot{\mathbf{v}} = \mathbf{S}(\mathbf{q})\dot{\boldsymbol{v}}(t) + \gamma(\mathbf{q}, \boldsymbol{v}) \tag{2.15}$$

where

$$\gamma(\mathbf{q}, \boldsymbol{v}) = \dot{\mathbf{S}}(\mathbf{q})\boldsymbol{v}(t) + \dot{\eta}(\mathbf{q}) \tag{2.16}$$

Pre-multiplying both sides of eq.(2.5) by $\mathbf{S}^T$ and noting that $\mathbf{S}^T\mathbf{A}^T = \mathbf{O}$ we obtain:

$$\mathbf{S}^T\mathbf{I}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{S}^T[\mathbf{u} - \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})] \tag{2.17}$$

where $\dot{\mathbf{v}}$ is given by eq.(2.15). Substituting $\dot{\mathbf{v}}$ into eq.(2.17) and solving for $\dot{\boldsymbol{v}}$ we obtain

$$\dot{\boldsymbol{v}} = -(\mathbf{S}^T\mathbf{I}(\mathbf{q})\mathbf{S})^{-1}[\mathbf{S}^T\mathbf{I}(\mathbf{q})\gamma - \mathbf{S}^T[\mathbf{u} - \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})]] \tag{2.18}$$

If $\mathbf{x} = [\mathbf{q}^T\ \boldsymbol{v}^T]^T$, a $(2n-m)$ dimensional vector, then we can express the system in the standard form as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{S}\boldsymbol{v} + \boldsymbol{\eta} \\ -(\mathbf{S}^T\mathbf{I}(\mathbf{q})\mathbf{S})^{-1}[\mathbf{S}^T\mathbf{I}(\mathbf{q})\boldsymbol{\gamma} - \mathbf{S}^T[\mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{b}(\mathbf{q},\dot{\mathbf{q}})]] \end{bmatrix} \tag{2.19}$$

Additionally, Yun and Sarkar (1998) implement a numerical method to calculate the orthogonal complement $\mathbf{S}$, the particular solution $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ as follows:

- Computing $\mathbf{S}(\mathbf{q})$
  (i) Obtain the orthogonal projector $\mathbf{H}$ onto the nullspace $\mathcal{N}(\mathbf{A})$ of $\mathbf{A}$ i.e. $\mathbf{H} = \mathbf{1} - \mathbf{A}^\dagger\mathbf{A}$, where $\mathbf{1}$ is the $n \times n$ identity matrix and $\mathbf{A}^+$ is the Moore-Penrose generalized inverse of $\mathbf{A}(\mathbf{q})$.
  (ii) Compute the singular-value decomposition of $\mathbf{H}(\mathbf{q})$, i.e., $\mathbf{H} = \mathbf{U}\boldsymbol{\Delta}\mathbf{V}^T$ where $\mathbf{U}$ and $\mathbf{V}$ are $n \times n$ orthogonal matrices, and $\boldsymbol{\Delta}$ is the diagonal matrix containing the singular values of $\mathbf{H}(\mathbf{q})$
  (iii) Choose $\mathbf{S}(\mathbf{q})$ as the first $n-m$ columns of the $\mathbf{U}$ matrix.
- Computing $\boldsymbol{\eta}$

$$\boldsymbol{\eta}(\mathbf{q}) = -\sigma\mathbf{A}^\dagger\mathbf{c}(\mathbf{q}) \tag{2.20}$$

- Computing $\boldsymbol{\gamma}(\mathbf{q},\boldsymbol{v})$. Differentiating eq.(2.12) and rearranging the expression thus resulting, we obtain:

$$\mathbf{A}\ddot{\mathbf{q}} = -\sigma\dot{\mathbf{c}} - \dot{\mathbf{A}}\dot{\mathbf{q}} \tag{2.21}$$

  whence,

$$\ddot{\mathbf{q}} = [\mathbf{1} - \mathbf{A}^+\mathbf{A}]\boldsymbol{\mu} + \mathbf{A}^+(-\sigma\dot{\mathbf{c}} - \dot{\mathbf{A}}\dot{\mathbf{q}}) \tag{2.22}$$

  where $\boldsymbol{\mu}$ is a $(n-m)$-dimensional vector. Comparing the above equation with eq.(2.14), and realizing that $\dot{\mathbf{c}} = \mathbf{A}\dot{\mathbf{q}}$, we obtain $\boldsymbol{\gamma}$ as

$$\boldsymbol{\gamma}(\mathbf{q},\boldsymbol{v}) = \mathbf{A}^+(-\sigma\mathbf{A} - \dot{\mathbf{A}})\dot{\mathbf{q}}$$

At the end of this process we obtain the required system in standard state-space form:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\upsilon}} \end{bmatrix} = \begin{bmatrix} \mathbf{S}\boldsymbol{\upsilon} + \boldsymbol{\eta} \\ -(\mathbf{S}^T\mathbf{I}(\mathbf{q})\mathbf{S})^{-1}\mathbf{S}^T[\mathbf{I}(\mathbf{q})\boldsymbol{\gamma} - \mathbf{u} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})] \end{bmatrix} \tag{2.23}$$

where γ is given by eq.(2.16). In order to distribute the above set, we may partition eq.(2.23) into four parts. For $i = I,\ II$ and III,

$$\dot{\mathbf{x}}^i = \begin{bmatrix} \dot{\mathbf{q}}^i \\ \dot{\boldsymbol{\nu}}^i \end{bmatrix} = \begin{bmatrix} \mathbf{S}^i\boldsymbol{\upsilon} + \boldsymbol{\eta}^i \\ -(\mathbf{S}^T\mathbf{I}\mathbf{S})^{-1}\mathbf{S}^{iT}(\mathbf{I}^i\boldsymbol{\gamma}^i + \mathbf{b}^i - \mathbf{1}^i\mathbf{u}^i) \end{bmatrix} \tag{2.24}$$

Moreoverm for platform-$\mathcal{P}$

$$\dot{\mathbf{x}}^P = \begin{bmatrix} \dot{\mathbf{q}}^P \\ \dot{\boldsymbol{\nu}}^P \end{bmatrix} = \begin{bmatrix} \mathbf{S}^P\boldsymbol{\upsilon} + \boldsymbol{\eta}_P \\ -(\mathbf{S}^T\mathbf{I}\mathbf{S})^{-1}(\mathbf{S}^P)^T(\mathbf{M}^p\boldsymbol{\gamma}^P - \mathbf{1}^P\mathbf{u}^P) \end{bmatrix} \tag{2.25}$$

where

$$\boldsymbol{\upsilon} = \boldsymbol{\upsilon}^I + \boldsymbol{\upsilon}^{II} + \boldsymbol{\upsilon}^{III} + \boldsymbol{\upsilon}^P \tag{2.26}$$

which is a 3-dimensional vector. If n^I, n^{II} and n^{III} are the numbers of state variables of dyads I, II and III, while n^P is that of the platform $\mathcal{P}$, i.e, $n = n^I + n^{II} + n^{III} + n^P$, then

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}^I \\ \mathbf{S}^{II} \\ \mathbf{S}^{III} \\ \mathbf{S}^P \end{bmatrix};\ \boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}^I \\ \boldsymbol{\gamma}^{II} \\ \boldsymbol{\gamma}^{III} \end{bmatrix};\ \boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\eta}^I \\ \boldsymbol{\eta}^{II} \\ \boldsymbol{\eta}^{III} \end{bmatrix} \tag{2.27}$$

where $\mathbf{S}^i$ is a $(n - m) \times n^i$ matrix, while $\boldsymbol{\eta}^i$ and $\boldsymbol{\upsilon}^i$ are n^i-dimensional vectors, for $i = I,\ II,\ III$ and P. For our system $n^I = n^{II} = n^{III} = 2$, while $n^P = 3$.

We note that the first part of the acceleration equation in all cases is $(\mathbf{S}^T\mathbf{I}\mathbf{S})^{-1}$. We also note that it is possible to perform each integration of the state sub-vectors independently, provided that this matrix is known. This could be achieved by communicating the full state of the system to all nodes at the beginning of each time step and computing the same matrix in *all* nodes.

This repetition of the computation does not slow down the performance when compared for the case where this computation is performed on one processor and communicated to others. In fact, the former approach is superior because it eliminates the communication step within an integration time-step.

Further, we note that the method suggested by Yun and Sarkar (1998) to compute the orthogonal complement S, involves the process of singular-value decomposition at *each* time-step. Since SVD is an iterative procedure, it renders the suggested method useless for real-time simulations. Efficient methods for the computation of orthogonal complement exist, such as that based on QR factorization, and are available in the appendix.

2.4. Method III: The Recursive Decoupled Natural Orthogonal Complement

The decoupled natural orthogonal complement (DeNOC) was developed as an extension to the natural orthogonal complement (NOC). We provide a brief outline of the NOC here and refer the reader to (Angeles, 2002) for further details. The equations of motion for a closed loop may be expressed in Newton-Euler form as

$$\mathbf{M}\dot{\mathbf{t}} = -\dot{\mathbf{M}}\mathbf{t} + \mathbf{w}^A + \mathbf{w}^G + \mathbf{w}^C \tag{2.28}$$

$$\mathbf{K}\mathbf{t} = \mathbf{0} \tag{2.29}$$

where

$\mathbf{M} \equiv (6n \times 6n)$-mass matrix

$\mathbf{t} \equiv [\mathbf{t}_1^T \dots \mathbf{t}_n^T]^T$, the $6n$-dimensional twist vector

$\mathbf{K} \equiv$ Constraint Jacobian

$\mathbf{w}^C \equiv$ $6n$-dimensional vector of joint constraint wrenches

$\mathbf{w}^A \equiv$ $6n$-dimensional vector of actuator wrenches

$\mathbf{w}^G \equiv$ $6n$-dimensional vector representing all other external wrenches

$$\mathbf{w} \equiv [\mathbf{w}_1^T \ldots \mathbf{w}_n^T]^T$$

$\mathbf{t}_i \equiv [\omega_i^T \;\; \mathbf{v}_i^T]^T$ for $i = 1, \ldots, n$ is the i^{th} six-dimensional twist vector.

$\omega_i \equiv$ three-dimensional vector of angular velocity of link-i

$\mathbf{v}_i \equiv$ three-dimensional velocity of the mass centre of link-i

$$\mathbf{w}_i \equiv [\mathbf{n}_i^T \;\; \mathbf{f}_i^T]^T$$

$\mathbf{n}_i \equiv$ three-dimensional moment acting on link-i

$\mathbf{f}_i \equiv$ three-dimensional forces acting on link-i

For a closed-loop mechanism, the constrained dynamics may be evaluated using a three-stage process, by a combination of the Newton-Euler and the Euler-Lagrange approaches.

In the *first* stage, a closed loop mechanism is cut open and Newton-Euler equations for the open chains obtained are written in task-space twist coordinates and projected from the task-space to the joint-space using the *natural orthogonal complement* $\mathbf{U}$. With this approach, the natural orthogonal complement $\mathbf{U}$ establishes the relation between the twist vector $\mathbf{t}$ and generalized velocities $\dot{\mathbf{q}}$ i.e.,

$$\mathbf{t} = \mathbf{U}\dot{\mathbf{q}} \tag{2.30}$$

In the *second* stage, the equations of motion are coupled by introducing Lagrange multipliers to incorporate loop-closure constraints resulting in a set of non-minimal Newton-Lagrange equations in joint space. *Finally*, the loop-closure natural orthogonal complement Θ is used to project the resulting equations to obtain a minimal set of equations of motion. The loop-closure orthogonal complement Θ establishes a relation between generalized velocities and actuated joint velocities, namely,

$$\dot{\mathbf{q}} = \Theta\dot{\mathbf{q}}_{ac} \tag{2.31}$$

The total orthogonal complement $\mathbf{T}$, which relates the $6n$-dimensional twist vector with the vector of actuated velocities, $\dot{\mathrm{q}}_{ac}$, is defined as

$$\mathrm{t} = \mathrm{U}\dot{\mathrm{q}} = \mathrm{U}\Theta\dot{\mathrm{q}}_{ac} = \mathrm{T}\dot{\mathrm{q}}_{ac} \tag{2.32}$$

Differentiating this equation once, we obtain:

$$\dot{\mathrm{t}} = \dot{\mathrm{T}}\dot{\mathrm{q}}_{ac} + \mathrm{T}\ddot{\mathrm{q}}_{ac} \tag{2.33}$$

Since the power developed by virtue of the constraint wrenches is zero, $\mathrm{T}^T\mathrm{w}^C = 0$, and constraint-free equations of motion are obtained by pre-multiplying eq.(2.28) with T^T as

$$\mathrm{T}^T\mathrm{M}\dot{\mathrm{t}} = \mathrm{T}^T(-\dot{\mathrm{M}}\mathrm{t} + \mathrm{w}^A + \mathrm{w}^G) \tag{2.34}$$

Substituting $\dot{\mathrm{t}}$ from eq.(2.33) in the above equation and rearranging term, we obtain

$$\mathrm{T}^T\mathrm{M}\mathrm{T}\ddot{\mathrm{q}}_{ac} = \mathrm{T}^T[-(\mathrm{M}\dot{\mathrm{T}} + \dot{\mathrm{M}}\mathrm{T})\dot{\mathrm{q}}_{ac} + \mathrm{w}^A + \mathrm{w}^G] \tag{2.35}$$

By using a state vector $\mathrm{x} = \begin{bmatrix}\mathrm{q}_{ac}^T & \dot{\mathrm{q}}_{ac}^T\end{bmatrix}^T$ these equations are written in the minimal state-space form of $2(n-m)$ equations, namely,

$$\dot{\mathrm{x}} = \begin{bmatrix}\dot{\mathrm{q}}_{ac}\\ \ddot{\mathrm{q}}_{ac}\end{bmatrix} = \begin{bmatrix}\dot{\mathrm{q}}_{ac}\\ (\mathrm{T}^T\mathrm{M}\mathrm{T})^{-1}\mathrm{T}^T[-(\mathrm{M}\dot{\mathrm{T}} + \dot{\mathrm{M}}\mathrm{T})\dot{\mathrm{q}}_{ac} + \mathrm{w}^A + \mathrm{w}^G]\end{bmatrix} \tag{2.36}$$

Alternatively by defining the state $\mathrm{x} = \begin{bmatrix}\mathrm{q}^T & \dot{\mathrm{q}}_{ac}^T\end{bmatrix}^T$ we may use the extended set of $(2n-m)$ state equations in order to avoid iterative solution of the position level/holonomic constraints, as shown below:

$$\dot{\mathrm{x}} = \begin{bmatrix}\dot{\mathrm{q}}\\ \ddot{\mathrm{q}}_{ac}\end{bmatrix} = \begin{bmatrix}\Theta\dot{\mathrm{q}}_{ac}\\ (\mathrm{T}^T\mathrm{M}\mathrm{T})^{-1}\mathrm{T}^T[-(\mathrm{M}\dot{\mathrm{T}} + \dot{\mathrm{M}}\mathrm{T})\dot{\mathrm{q}}_{ac} + \mathrm{w}^A + \mathrm{w}^G]\end{bmatrix} \tag{2.37}$$

The DeNOC method was derived as an extension of the NOC method by noting that the total orthogonal complement for the closed loop mechanism may be factorized as

$$\mathrm{T} = \mathrm{N}_l\mathrm{N}_c\mathrm{N}_d \tag{2.38}$$

where $\mathbf{N}_l$ is a lower triangular matrix, $\mathbf{N}_c$ is a full matrix and $\mathbf{N}_d$ a block diagonal matrix. The procedure to obtain these matrices is rather elaborate and is being left out in the interest of space. The interested reader is referred to (Saha and Schiehlen, 2001) for details.

The principal advantage of the DeNOC method is that *it admits a recursive solution of minimal order*, as compared with other methods, which are either recursive but non-minimal or minimal but non-recursive. A modified procedure based on the concepts of the DeNOC was developed that is better suited to distributed computation and will be introduced presently.

CHAPTER 3

The Recursive Decoupled Natural Orthogonal Complement

In this chapter, we introduce a simplified method for the application of the DeNOC to parallel manipulators using the 3 $\underline{R}RR$ planar parallel mechanism studied in the previous chapter as a case study. Specifically, we adopt a *two-step* approach and emphasize the use of spatial parallelism in the mechanical system for distribution of the dynamics computations. While we discuss only a three-dof planar parallel manipulator, the concepts are readily extendable to other serial and parallel mechanical systems.

Finally, we note that there has been an increasing trend towards redundant actuation in parallel manipulators in order to better distribute the load and employ smaller actuators. The DeNOC formulation lends itself quite naturally to the handling parallel mechanisms with or without redundant actuation.

3.1. Forward Kinematics

The forward kinematics problem for a parallel manipulator is defined as: Given the actuated-joint angles, velocities and accelerations, find the position, twists and twist-rates of the platform and all the other links. Figure 3.1 shows the three-dof planar parallel manipulator. We divide the manipulator into three serial chains, I, II

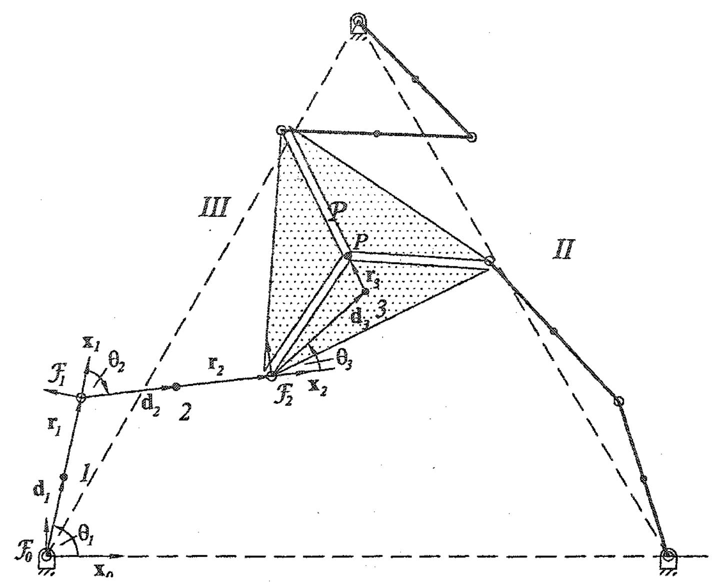


Figure 3.1: 3-dof planar parallel manipulator.

and III, by dividing the rigid platform $\mathcal{P}$ into *three parts* such that the end effector of *each open chain* lies at point P, the mass-centre of the platform $\mathcal{P}$. Cutting the platform in this manner is advantageous, as compared to the division schemes in the previous chapters, due to the reasons below:

- Torques may be applied to the joints that otherwise need to be cut to open the chains.
- Joint friction may be accommodated directly for such joints.
- A cut platform produces a more streamlined recursive kinematic and dynamic modelling for parallel manipulators.
- Dynamic models of *fully redundant* manipulators may be obtained.

The first two advantages are discussed in greater detail in Yiu *et al.* (2001). Below we give an outline of the remaining issues.

Position Analysis. The displacement analysis is a critical first step; we adopt here the approach proposed by Ma and Angeles (1989) to this end.

Velocity Analysis. For *each chain* we define the two-dimensional position vectors $\mathbf{d}_i$ from the ith joint axis to the mass centre of link i, $\mathbf{r}_i$ from mass centre of link i to the $(i+1)$st joint axis and $\mathbf{a}_i = \mathbf{d}_i + \mathbf{r}_i$ as shown in Fig. 3.1.

The twist of the end effector of *each chain* is given by Saha and Schiehlen (2001) as

$$\mathbf{t}_P = \mathbf{B}_{P3}\mathbf{t}_3 \tag{3.1}$$

where

$$\mathbf{B}_{P3} = \begin{bmatrix} \mathbf{1} & \mathbf{0}^T \\ \mathbf{E}\mathbf{r}_3 & 1 \end{bmatrix}; \mathbf{E} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

and $\mathbf{t}_3$ is the twist of the third link with respect to its mass centre, namely,

$$\mathbf{t}_3 = \mathbf{B}_{32}\mathbf{t}_2 + \mathbf{p}_3\dot{\theta}_3;\ \mathbf{B}_{32} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{E}(\mathbf{r}_2 + \mathbf{d}_3) & \mathbf{1} \end{bmatrix};\ \mathbf{p}_3 = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{d}_3 \end{bmatrix} \tag{3.2}$$

where the 3×3 matrix $\mathbf{B}_{32}$ is called the *twist-propagation matrix* and $\mathbf{p}_3$ is called the *twist generator*; $\mathbf{t}_2$ is the twist of link-2 with respect to its mass centre; $\dot{\theta}_3$ is the relative angular joint velocity of the third joint, while $\mathbf{0}$ is the 2 dimensional zero vector and $\mathbf{1}$ is the 2×2 identity matrix.

A useful relation is first introduced, which will be exploited in the ensuing analysis. Let $\mathbf{a} = \mathbf{b}x + \mathbf{c}$, where $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are three-dimensional vectors, while x is a scalar; then,

$$x = \frac{\mathbf{b}^T}{\mathbf{b}^T\mathbf{b}}(\mathbf{a} - \mathbf{c})$$

Substituting x back in $\mathbf{a} = \mathbf{b}x + \mathbf{c}$ and rearranging terms we obtain

$$(\mathbf{1} - \frac{\mathbf{b}\mathbf{b}^T}{\mathbf{b}^T\mathbf{b}})\mathbf{a} = (\mathbf{1} - \frac{\mathbf{b}\mathbf{b}^T}{\mathbf{b}^T\mathbf{b}})\mathbf{c}$$

where $\mathbf{1}$ is the 3×3 identity matrix.

Further, substituting $\mathbf{t}_3$ into eq.(3.1), we obtain

$$\mathbf{t}_P = \mathbf{B}_{P3}(\mathbf{B}_{32}\mathbf{t}_2 + \mathbf{p}_3\dot{\theta}_3) \tag{3.3}$$

Solving eq.(3.3) for $\dot{\theta}_3$:

$$\dot{\theta}_3 = \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}(\mathbf{t}_P - \mathbf{B}_{P2}\mathbf{t}_2) \tag{3.4}$$

where the three-dimensional vector $\tilde{\mathbf{p}}_3$ is defined as $\tilde{\mathbf{p}}_3 = \mathbf{B}_{P3}\mathbf{p}_3$ and $\delta_3 = \tilde{\mathbf{p}}_3^T\tilde{\mathbf{p}}_3$. Therefore, when we finally substitute $\dot{\theta}_3$ into eq.(3.3) we obtain:

$$\mathbf{\Phi}_3\mathbf{t}_P = \mathbf{\Phi}_3\mathbf{B}_{P2}\mathbf{t}_2 \tag{3.5}$$

where $\mathbf{\Phi}_3 = \mathbf{1} - \tilde{\mathbf{p}}_3\tilde{\mathbf{p}}_3^T/\delta_3$ and the property $\mathbf{B}_{P3}\mathbf{B}_{32} = \mathbf{B}_{P2}$ has been used. Again, the twist of link-2 is given by

$$\mathbf{t}_2 = \mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2; \ \mathbf{B}_{21} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{E}(\mathbf{r}_1 + \mathbf{d}_2) & \mathbf{1} \end{bmatrix}; \ \mathbf{p}_2 = \begin{bmatrix} 1 \\ \mathbf{E}\mathbf{d}_2 \end{bmatrix}$$

where $\mathbf{t}_1$ is the twist of link-1 with respect to its mass centre, $\dot{\theta}_2$ is the relative angular joint velocity of the second joint, $\mathbf{0}$ is the two-dimensional zero vector and $\mathbf{1}$ is the 2×2 identity matrix. Substituting $\mathbf{t}_2$ into eq.(3.5), we obtain

$$\mathbf{\Phi}_3\mathbf{t}_P = \mathbf{\Phi}_3\mathbf{B}_{P2}(\mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2) \tag{3.6}$$

Solving for $\dot{\theta}_2$ we obtain

$$\dot{\theta}_2 = \frac{\tilde{\mathbf{p}}_2^T}{\delta_2}(\mathbf{t}_P - \mathbf{B}_{P1}\mathbf{t}_1) \tag{3.7}$$

where $\tilde{\mathbf{p}}_2 = \mathbf{\Phi}_3\mathbf{B}_{P2}\mathbf{p}_2$ and $\delta_2 = \tilde{\mathbf{p}}_2^T\tilde{\mathbf{p}}_2$. Substituting $\dot{\theta}_2$ into eq.(3.6) leads to

$$\mathbf{\Phi}_2\mathbf{t}_P = \mathbf{\Phi}_2\mathbf{B}_{P1}\mathbf{t}_1 \tag{3.8}$$

where the 3×3 matrix $\mathbf{\Phi}_2$ is defined as $\mathbf{\Phi}_2 = \mathbf{\Phi}_3 - \tilde{\mathbf{p}}_2\tilde{\mathbf{p}}_2^T/\delta_2$ and the properties $\mathbf{B}_{P2}\mathbf{B}_{21} = \mathbf{B}_{P1}$ and $\mathbf{\Phi}_3^T\mathbf{\Phi}_3 = \mathbf{\Phi}_3$ have been used.

Finally, substituting $\mathbf{t}_1 = \mathbf{p}_1\dot{\theta}_1$ into eq.(3.8), we can express the twist of the platform $\mathcal{P}$ in terms of $\dot{\theta}_1$ as

$$\mathbf{\Phi}_2\mathbf{t}_P = \mathbf{\Phi}_2\mathbf{B}_{P1}\mathbf{p}_1\dot{\theta}_1 \tag{3.9}$$

Note that $\mathbf{\Phi}_2$ is a projection matrix and is thus singular.

Writing eq.(3.9) for *each open chain* we obtain:

$$\begin{aligned}
\mathbf{K}\mathbf{t}_P &= \Phi\mathbf{B}\mathbf{P}\dot{\boldsymbol{\theta}}_{ac} \quad \text{where} \\
\mathbf{K} &= [\Phi_2^I + \Phi_2^{II} + \Phi_2^{III}] \;:\; 3\times 3 \\
\Phi &= \begin{bmatrix}\Phi_2^I & \Phi_2^{II} & \Phi_2^{III}\end{bmatrix} \;:\; 3\times 9 \\
\mathbf{B} &= \mathrm{diag}(\mathbf{B}_{P1}^I, \mathbf{B}_{P1}^{II}, \mathbf{B}_{P1}^{III}) \;:\; 9\times 9 \\
\mathbf{P} &= \mathrm{diag}(\mathbf{p}_1^I, \mathbf{p}_1^{II}, \mathbf{p}_1^{III}) \;:\; 9\times 3 \\
\dot{\boldsymbol{\theta}}_{ac} &= \begin{bmatrix}\dot{\theta}_1^I & \dot{\theta}_1^{II} & \dot{\theta}_1^{III}\end{bmatrix}^T
\end{aligned}$$

where all dimensions have been stated for clarity. Finally if $\mathbf{K}$ is nonsingular,

$$\mathbf{t}_P = \mathbf{K}^{-1}\Phi\mathbf{B}\mathbf{P}\dot{\boldsymbol{\theta}}_{ac} \tag{3.10}$$

Substituting $\mathbf{t}_P$ from eq.(3.10) into eq.(3.7) we obtain

$$\dot{\theta}_2 = \frac{\tilde{\mathbf{p}}_2^T}{\delta_2}(\mathbf{K}^{-1}\Phi\mathbf{B}\mathbf{P}\dot{\boldsymbol{\theta}}_{ac} - \mathbf{B}_{P1}\mathbf{p}_1\dot{\theta}_1) \tag{3.11}$$

and substituting $\mathbf{t}_P$, $\mathbf{t}_2$, $\mathbf{t}_1$ and $\dot{\theta}_2$ into eq.(3.4),

$$\begin{aligned}
\dot{\theta}_3 &= \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}[\mathbf{t}_P - \mathbf{B}_{P2}(\mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2)] \\
&= \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}(\mathbf{t}_P - \mathbf{B}_{P1}\mathbf{p}_1\dot{\theta}_1 - \mathbf{B}_{P2}\mathbf{p}_2\dot{\theta}_2) \\
&= \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}[(\mathbf{t}_P - \mathbf{B}_{P1}\mathbf{t}_1) - \mathbf{B}_{P2}\mathbf{p}_2\frac{\tilde{\mathbf{p}}_2^T}{\delta_2}(\mathbf{t}_p - \mathbf{B}_{P1}\mathbf{t}_1)] \\
&= \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}(\mathbf{1} - \frac{1}{\delta_2}\mathbf{B}_{P2}\mathbf{p}_2\tilde{\mathbf{p}}_2^T)(\mathbf{t}_P - \mathbf{B}_{P1}\mathbf{t}_1)
\end{aligned} \tag{3.12}$$

which can be written as

$$\dot{\theta}_3 = \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}\Psi_2^T(\mathbf{K}^{-1}\Phi\mathbf{B}\mathbf{P}\dot{\boldsymbol{\theta}}_{ac} - \mathbf{B}_{P1}\mathbf{p}_1\dot{\theta}_1) \tag{3.13}$$

where the 3×3 matrix Ψ_2 is defined as $\Psi_2 = (\mathbf{1} - \mathbf{B}_{P2}\mathbf{p}_2\tilde{\mathbf{p}}_2^T/\delta_2)^T$ and $\mathbf{1}$ is the 3×3 identity matrix.

We note that eqs.(3.11) and (3.13) are general and applicable to *each open chain* and that the bracketed term on the right hand side of each equation is the same. This term can be written specifically for *each open chain* as

$$[\mathrm{K}^{-1}\Phi\mathrm{BP}\dot{\theta}_{ac} - \mathrm{B}_{P1}\mathrm{p}_1\dot{\theta}_1]^I = \left[[\mathrm{K}^{-1}\Phi_2 - 1]^I \;\; [\mathrm{K}^{-1}\Phi_2]^{II} \;\; [\mathrm{K}^{-1}\Phi_2]^{III}\right]\mathrm{BP}\dot{\theta}_{ac}$$

$$[\mathrm{K}^{-1}\Phi\mathrm{BP}\dot{\theta}_{ac} - \mathrm{B}_{P1}\mathrm{p}_1\dot{\theta}_1]^{II} = \left[[\mathrm{K}^{-1}\Phi_2]^I \;\; [\mathrm{K}^{-1}\Phi_2 - 1]^{II} \;\; [\mathrm{K}^{-1}\Phi_2]^{III}\right]\mathrm{BP}\dot{\theta}_{ac}$$

$$[\mathrm{K}^{-1}\Phi\mathrm{BP}\dot{\theta}_{ac} - \mathrm{B}_{P1}\mathrm{p}_1\dot{\theta}_1]^{III} = \left[[\mathrm{K}^{-1}\Phi_2]^I \;\; [\mathrm{K}^{-1}\Phi_2]^{II} \;\; [\mathrm{K}^{-1}\Phi_2 - 1]^{III}\right]\mathrm{BP}\dot{\theta}_{ac}$$

Finally we end up with the relation between joint rates and actuated joint rates

$$\dot{\theta} = \begin{bmatrix} \bar{\mathrm{P}}^I & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \bar{\mathrm{P}}^{II} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \bar{\mathrm{P}}^{III} \end{bmatrix} \begin{bmatrix} \mathrm{L}^I \\ \mathrm{L}^{II} \\ \mathrm{L}^{III} \end{bmatrix} \mathrm{BP}\dot{\theta}_{ac} \tag{3.14}$$

where the 3×9 matrix $\bar{\mathrm{P}}_i$ is defined as $\bar{\mathrm{P}}_i = [\mathrm{diag}(\tilde{\mathrm{p}}_1^T/\delta_1, \tilde{\mathrm{p}}_2^T/\delta_2, \tilde{\mathrm{p}}_3^T\Psi_2^T/\delta_3)]^i$, while $\tilde{\mathrm{p}}_1^i$ as explicitly $\tilde{\mathrm{p}}_1^i = (\mathrm{B}_{P1}\mathrm{p}_1)^i$ for $i = I, II$ and III, and the 9×9 matrices $\mathbf{L}$ are defined for *each open chain* as

$$\mathrm{L}^I = \begin{bmatrix} 1 & \mathrm{O} & \mathrm{O} \\ [\mathrm{K}^{-1}\Phi_2 - 1]^I & [\mathrm{K}^{-1}\Phi_2]^{II} & [\mathrm{K}^{-1}\Phi_2]^{III} \\ [\mathrm{K}^{-1}\Phi_2 - 1]^I & [\mathrm{K}^{-1}\Phi_2]^{II} & [\mathrm{K}^{-1}\Phi_2]^{III} \end{bmatrix}$$

$$\mathrm{L}^{II} = \begin{bmatrix} \mathrm{O} & 1 & \mathrm{O} \\ [\mathrm{K}^{-1}\Phi_2]^I & [\mathrm{K}^{-1}\Phi_2 - 1]^{II} & [\mathrm{K}^{-1}\Phi_2]^{III} \\ [\mathrm{K}^{-1}\Phi_2]^I & [\mathrm{K}^{-1}\Phi_2 - 1]^{II} & [\mathrm{K}^{-1}\Phi_2]^{III} \end{bmatrix}$$

$$\mathrm{L}^{III} = \begin{bmatrix} \mathrm{O} & \mathrm{O} & 1 \\ [\mathrm{K}^{-1}\Phi_2]^I & [\mathrm{K}^{-1}\Phi_2]^{II} & [\mathrm{K}^{-1}\Phi_2 - 1]^{III} \\ [\mathrm{K}^{-1}\Phi_2]^I & [\mathrm{K}^{-1}\Phi_2]^{II} & [\mathrm{K}^{-1}\Phi_2 - 1]^{III} \end{bmatrix}$$

Equation (3.14) can be written in compact form as

$$\dot{\theta} = \bar{\mathrm{P}}\mathrm{L}\mathrm{B}\mathrm{P}\dot{\theta}_{ac} \tag{3.15}$$

where the 9×27 matrix $\bar{\mathbf{P}}$ is defined as $\bar{\mathbf{P}} = \text{diag}(\bar{\mathbf{P}}^I, \bar{\mathbf{P}}^{II}, \bar{\mathbf{P}}^{III})$ and the 27×9 matrix $\mathbb{L}$ is defined as $\mathbb{L} = \begin{bmatrix}(\mathbf{L}^I)^T & (\mathbf{L}^{II})^T & (\mathbf{L}^{III})^T\end{bmatrix}^T$. Note that, except for $\mathbb{L}$, which is full but still retains a special form, all other matrices are block-diagonal.

Acceleration Analysis. We find now the acceleration terms, for any chain, by differentiating eq.(3.3) as

$$\dot{\mathbf{t}}_P = \dot{\mathbf{B}}_{P3}\mathbf{t}_3 + \mathbf{B}_{P3}(\dot{\mathbf{B}}_{32}\mathbf{t}_2 + \mathbf{B}_{32}\dot{\mathbf{t}}_2 + \dot{\mathbf{p}}_3\dot{\theta}_3) + \mathbf{B}_{P3}\mathbf{p}_3\ddot{\theta}_3 \tag{3.16}$$

Solving for $\ddot{\theta}_3$,

$$\ddot{\theta}_3 = \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}[\dot{\mathbf{t}}_P - \dot{\mathbf{B}}_{P3}\mathbf{t}_3 - \mathbf{B}_{P3}(\dot{\mathbf{B}}_{32}\mathbf{t}_2 + \mathbf{B}_{32}\dot{\mathbf{t}}_2 + \dot{\mathbf{p}}_3\dot{\theta}_3)] \tag{3.17}$$

Substituting $\ddot{\theta}_3$ back into eq.(3.16),

$$\mathbf{\Phi}_3\dot{\mathbf{t}}_P = \mathbf{\Phi}_3[\dot{\mathbf{B}}_{P3}\mathbf{t}_3 + \mathbf{B}_{P3}(\dot{\mathbf{B}}_{32}\mathbf{t}_2 + \mathbf{B}_{32}\dot{\mathbf{t}}_2 + \dot{\mathbf{p}}_3\dot{\theta}_3)] \tag{3.18}$$

Here we can obtain the expression for $\dot{\mathbf{\Phi}}_3$. Substituting $\mathbf{t}_3$ and $\dot{\theta}_3$ into eq.(3.18) and re-arranging,

$$\begin{aligned}\mathbf{\Phi}_3\dot{\mathbf{t}}_P &= \mathbf{\Phi}_3\mathbf{B}_{P2}\dot{\mathbf{t}}_2 + \mathbf{\Phi}_3\dot{\mathbf{B}}_{P2}\mathbf{t}_2 - [\mathbf{\Phi}_3(\dot{\mathbf{B}}_{P3}\mathbf{p}_3 + \mathbf{B}_{P3}\dot{\mathbf{p}}_3)\frac{\tilde{\mathbf{p}}_3^T}{\delta_3}]\mathbf{B}_{P2}\mathbf{t}_2 \\ &\quad + [\mathbf{\Phi}_3(\dot{\mathbf{B}}_{P3}\mathbf{p}_3 + \mathbf{B}_{P3}\dot{\mathbf{p}}_3)\frac{\tilde{\mathbf{p}}_3^T}{\delta_3}]\mathbf{t}_P\end{aligned} \tag{3.19}$$

where the property $(\dot{\mathbf{B}}_{P3}\mathbf{B}_{32} + \mathbf{B}_{P3}\dot{\mathbf{B}}_{32}) = \dot{\mathbf{B}}_{P2}$ has been used. Time-differentiating eq.(3.5) we obtain:

$$\mathbf{\Phi}_3\dot{\mathbf{t}}_P = \mathbf{\Phi}_3\mathbf{B}_{P2}\dot{\mathbf{t}}_2 + \mathbf{\Phi}_3\dot{\mathbf{B}}_{P2}\mathbf{t}_2 + \dot{\mathbf{\Phi}}_3\mathbf{B}_{P2}\mathbf{t}_2 - \dot{\mathbf{\Phi}}_3\mathbf{t}_P \tag{3.20}$$

Comparing eq.(3.19) with eq.(3.20), we obtain:

$$\dot{\mathbf{\Phi}}_3 = -\mathbf{\Phi}_3(\dot{\mathbf{B}}_{P3}\mathbf{p}_3 + \mathbf{B}_{P3}\dot{\mathbf{p}}_3)\frac{\tilde{\mathbf{p}}_3^T}{\delta_3} \tag{3.21}$$

or

$$\dot{\mathbf{\Phi}}_3 = -\mathbf{\Phi}_3\frac{\dot{\overline{\mathbf{p}_3}}\tilde{\mathbf{p}}_3^T}{\delta_3} \tag{3.22}$$

Now $\mathbf{t}_2 = \mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2$, and hence, $\dot{\mathbf{t}}_2 = \dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \mathbf{p}_2\ddot{\theta}_2 + \dot{\mathbf{p}}_2\dot{\theta}_2$. Substituting $\mathbf{t}_2$ and $\dot{\mathbf{t}}_2$ in eq.(3.20) we obtain:

$$
\begin{aligned}
(\dot{\Phi}_3\mathbf{t}_P + \Phi_3\dot{\mathbf{t}}_P) &= \Phi_3\mathbf{B}_{P2}\mathbf{p}_2\ddot{\theta}_2 + [\Phi_3\mathbf{B}_{P2}(\dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{p}}_2\dot{\theta}_2) \\
&\quad + \Phi_3\dot{\mathbf{B}}_{P2}\mathbf{t}_2 + \dot{\Phi}_3\mathbf{B}_{P2}\mathbf{t}_2] \qquad (3.23)
\end{aligned}
$$

Solving for $\ddot{\theta}_2$,

$$
\begin{aligned}
\ddot{\theta}_2 &= \frac{\tilde{\mathbf{p}}_2^T}{\delta_2}[(\dot{\Phi}_3\mathbf{t}_P + \Phi_3\dot{\mathbf{t}}_P) - [\Phi_3\mathbf{B}_{P2}(\dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{p}}_2\dot{\theta}_2) \\
&\quad + \Phi_3\dot{\mathbf{B}}_{P2}\mathbf{t}_2 + \dot{\Phi}_3\mathbf{B}_{P2}\mathbf{t}_2]] \qquad (3.24)
\end{aligned}
$$

substituting $\ddot{\theta}_2$ back into eq.(3.23),

$$
\begin{aligned}
\bar{\Phi}_2(\dot{\Phi}_3\mathbf{t}_P + \Phi_3\dot{\mathbf{t}}_p) &= \bar{\Phi}_2[\Phi_3\mathbf{B}_{P2}(\dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{p}}_2\dot{\theta}_2) \\
&\quad + \Phi_3\dot{\mathbf{B}}_{P2}\mathbf{t}_2 + \dot{\Phi}_3\mathbf{B}_{P2}\mathbf{t}_2] \qquad (3.25)
\end{aligned}
$$

where $\bar{\Phi}_2 = \mathbf{1} - \tilde{\mathbf{p}}_2\tilde{\mathbf{p}}_2^T/\delta_2$. However, we note that

$$
\bar{\Phi}_2\Phi_3 = (\Phi_3 - \frac{\tilde{\mathbf{p}}_2\tilde{\mathbf{p}}_2^T}{\delta_2}) = \Phi_2
$$

where the property $\Phi_3^T\Phi_3 = \Phi_3$ was used. Substituting $\bar{\Phi}_2\Phi_3 = \Phi_2$ and rearranging eqs.(3.24 & 3.25) leads to

$$
\begin{aligned}
\ddot{\theta}_2 &= \frac{\tilde{\mathbf{p}}_2^T}{\delta_2}[\Phi_3(\dot{\mathbf{t}}_P - \mathbf{a}_1) - \mathbf{a}_2] \qquad (3.26) \\
\Phi_2\dot{\mathbf{t}}_P &= \Phi_2\mathbf{a}_1 + \bar{\Phi}_2\mathbf{a}_2 \qquad (3.27)
\end{aligned}
$$

where $\mathbf{a}_1 = \mathbf{B}_{P2}(\dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{p}}_2\dot{\theta}_2) + \dot{\mathbf{B}}_{P2}\mathbf{t}_2$ and $\mathbf{a}_2 = \dot{\Phi}_3(\mathbf{B}_{P2}\mathbf{t}_2 - \mathbf{t}_P)$. Finally adding eq.(3.27) for *each open chain* and solving for $\dot{\mathbf{t}}_p$,

$$
\dot{\mathbf{t}}_P = \mathbf{K}^{-1}([\Phi_2\mathbf{a}_1 + \bar{\Phi}_2\mathbf{a}_2]^I + [\Phi_2\mathbf{a}_1 + \bar{\Phi}_2\mathbf{a}_2]^{II} + [\Phi_2\mathbf{a}_1 + \bar{\Phi}_2\mathbf{a}_2]^{III}) \qquad (3.28)
$$

Summary. The forward-kinematics problem is solved as follows: First, the displacement analysis can be implemented using available methods such as that proposed by Ma and Angeles (1989). Next, given the activated joint rates all unactuated joint rates are calculated recursively using eqs.(3.10), (3.11) and (3.12); finally, the twists are obtained from eq.(3.10). Further, using actuated joint accelerations, $\dot{\mathbf{t}}_P$ is calculated. Once $\dot{\mathbf{t}}_P$ is available all unactuated joint accelerations are calculated using eqs.(3.22), (3.24) and (3.17), which is again recursive in nature. Finally the twist rates are obtained from eq.(3.28), thus completing the kinematics part of the derivation.

Distribution of Forward Kinematics. A four-processor parallel processing configuration can be used for distributed evaluation of forward kinematics, where three processors, referred to as *nodes*, compute independent quantities for chain I, II and III, while a fourth processor, referred to as *central*, is required to evaluate dependent quantities. A brief outline the process of distribution of the forward kinematics is as follows:

(i) Nodes: With the data, calculate $\mathbf{B}_{21}$, $\mathbf{B}_{32}$, $\mathbf{B}_{31}$, $\mathbf{B}_{P3}$,$\mathbf{B}_{P2}$, $\mathbf{B}_{P1}$, $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$. For each chain, we calculate:

$$\begin{aligned}
\tilde{\mathbf{p}}_3 &= \mathbf{B}_{P3}\mathbf{p}_3 \\
\delta_3 &= \tilde{\mathbf{p}}_3^T\tilde{\mathbf{p}}_3 \\
\mathbf{\Phi}_3 &= [\mathbf{1} - \frac{\tilde{\mathbf{p}}_3\tilde{\mathbf{p}}_3^T}{\delta_3}] \\
\tilde{\mathbf{p}}_2 &= \mathbf{B}_{P2}\mathbf{p}_2 \\
\delta_2 &= \tilde{\mathbf{p}}_2^T\tilde{\mathbf{p}}_2 \\
\mathbf{\Phi}_2 &= \mathbf{\Phi}_3 - \frac{\tilde{\mathbf{p}}_2\tilde{\mathbf{p}}_2^T}{\delta_2}
\end{aligned}$$

(ii) Central: Form matrices $\mathbf{K}$, $\mathbf{\Phi}$, $\mathbf{B}$ and $\mathbf{P}$ with values received from each chain and calculate the platform twist $\mathbf{t}_P$ from:

$$\mathbf{K}\mathbf{t}_p = \mathbf{\Phi}\mathbf{B}\mathbf{P}\dot{\boldsymbol{\theta}}_{ac}$$

(iii) Nodes: Obtain the twists and joint rates recursively for each chain, using $\mathbf{t}_P$ calculated by the central processor.

$$\begin{aligned}
\mathbf{t}_1 &= \mathbf{p}_1\dot{\theta}_1 \\
\dot{\theta}_2 &= \frac{\tilde{\mathbf{p}}_2^T}{\delta_2}(\mathbf{t}_P - \mathbf{B}_{P1}\mathbf{t}_1) \\
\mathbf{t}_2 &= \mathbf{B}_{21}\mathbf{t}_1 + \mathbf{p}_2\dot{\theta}_2 \\
\dot{\theta}_3 &= \frac{\tilde{\mathbf{p}}_3^T}{\delta_3}(\mathbf{t}_P - \mathbf{B}_{P2}\mathbf{t}_2) \\
\mathbf{t}_3 &= \mathbf{B}_{32}\mathbf{t}_2\mathbf{p}_3\dot{\theta}_3
\end{aligned}$$

Now calculate twist rates and joint accelerations. First, calculate i.e. $\dot{\mathbf{B}}_{21}$, $\dot{\mathbf{B}}_{32}$, $\dot{\mathbf{B}}_{31}$, $\dot{\mathbf{B}}_{P3}$,$\dot{\mathbf{B}}_{P2}$, $\dot{\mathbf{B}}_{P1}$, $\dot{\mathbf{p}}_1$, $\dot{\mathbf{p}}_2$ and $\dot{\mathbf{p}}_3$ using joint velocities calculated above:

$$\begin{aligned}
\dot{\mathbf{t}}_1 &= \mathbf{p}_1\ddot{\theta}_1 + \dot{\mathbf{p}}_1\dot{\theta}_1 \\
\mathbf{a}_1 &= \mathbf{B}_{P2}(\dot{\mathbf{B}}_{21}\mathbf{t}_1 + \mathbf{B}_{21}\dot{\mathbf{t}}_1 + \dot{\mathbf{p}}_2\dot{\theta}_2) + \dot{\mathbf{B}}_{P2}\mathbf{t}_2 \\
\dot{\mathbf{\Phi}}_3 &= -\mathbf{\Phi}_3(\dot{\mathbf{B}}_{P3}\mathbf{p}_3 + \mathbf{B}_{P3}\dot{\mathbf{p}}_3)\frac{\tilde{\mathbf{p}}_3^T}{\delta_3} \\
\mathbf{a}_2 &= \dot{\mathbf{\Phi}}_3(\mathbf{B}_{P2}\mathbf{t}_2 - \mathbf{t}_P) \\
\bar{\mathbf{\Phi}}_2 &= 1 - \frac{\tilde{\mathbf{p}}_2\tilde{\mathbf{p}}^T}{\delta_2}
\end{aligned}$$

(iv) Central: using $\mathbf{\Phi}_2$, $\bar{\mathbf{\Phi}}_2$, $\mathbf{a}_1$ and $\mathbf{a}_2$ from each chain calculate $\dot{\mathbf{t}}_P$ from:

$$\mathbf{K}\dot{\mathbf{t}}_P = [\mathbf{\Phi}_2\mathbf{a}_1 + \bar{\mathbf{\Phi}}_2\mathbf{a}_2]^I + [\mathbf{\Phi}_2\mathbf{a}_1 + \bar{\mathbf{\Phi}}_2\mathbf{a}_2]^{II} + [\mathbf{\Phi}_2\mathbf{a}_1 + \bar{\mathbf{\Phi}}_2\mathbf{a}_2]^{III}]$$

(v) Nodes: Calculate joint accelerations and twist rates for each chain:

$$
\begin{aligned}
\ddot{\theta}_2 &= \frac{\tilde{\mathrm{p}}_2^T}{\delta_2}[\Phi_3(\dot{\mathrm{t}}_P - \mathrm{a}_1) - \mathrm{a}_2] \\
\dot{\mathrm{t}}_2 &= \dot{\mathrm{B}}_{21}\mathrm{t}_1 + \mathrm{B}_{21}\dot{\mathrm{t}}_1\dot{\mathrm{p}}_2\dot{\theta}_2 + \mathrm{p}_2\ddot{\theta}_2 \\
\ddot{\theta}_3 &= \frac{\tilde{\mathrm{p}}_3^T}{\delta_3}[\dot{\mathrm{t}}_P - \dot{\mathrm{B}}_{P3}\mathrm{t}_3 - \mathrm{B}_{P3}(\dot{\mathrm{B}}_{32}\mathrm{t}_2 + \mathrm{B}_{32}\dot{\mathrm{t}}_2 + \dot{\mathrm{p}}_3\dot{\theta}_3)] \\
\dot{\mathrm{t}}_3 &= \mathrm{B}_{32}\dot{\mathrm{t}}_2 + \dot{\mathrm{B}}_{32}\mathrm{t}_2 + \mathrm{p}_3\dot{\theta}_3\dot{\mathrm{p}}_3\ddot{\theta}_3
\end{aligned}
$$

3.2. Inverse Dynamics

The inverse-dynamics problem is defined as: Given the time-histories of all the system degrees of freedom, compute the time-histories of the controlling actuated joint torques and forces. As in the kinematics calculations, we again divide the platform into three parts and assign cut sections of platform $\mathcal{P}$ to each open chain. Each cut section thus becomes the "third link" of the corresponding chain. Further, we divide the mass of the platform (including any tool carried by the platform) and assign its corresponding moment of inertia, with respect to the mass center of the platform, to the "third link" of each chain. Any working wrench applied to the platform has to be appropriately divided in a similar fashion. The Newton-Euler equations for *each open chain* is, thus,

$$\mathbf{M}\dot{\mathbf{t}} + \dot{\mathbf{M}}\mathbf{t} = \mathbf{w}^{AC} + \underbrace{\mathbf{w}^W + \mathbf{w}^g}_{\mathbf{w}^G} + \mathbf{w}^C \tag{3.29}$$

where $\mathbf{M}$ is the 9×9 mass matrix, $\mathbf{t}$ is the 9-dimensional twist vector of the whole chain, $\mathbf{w}^{AC}$ is the wrench applied by the actuators, $\mathbf{w}^W$ is the working wrench applied at the platform, $\mathbf{w}^g$ is the gravity wrench and $\mathbf{w}^C$ are the constraint wrenches all these being 9-dimensional vectors. The friction forces have been neglected for the sake of simplicity. The twist vector $\mathbf{t}$ is given by (Saha, 1999):

$$\mathrm{t} = \mathrm{N}_l\mathrm{N}_d\dot{\theta} \tag{3.30}$$

where $\mathbf{N}_l\mathbf{N}_d$ is the decoupled orthogonal complement and $\dot{\boldsymbol{\theta}}$ is the joint-rate vector of the chain. For our manipulator, for *each open chain* eq.(3.30) is

$$\mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{O} & \mathbf{O} \\ \mathbf{B}_{21} & \mathbf{1} & \mathbf{O} \\ \mathbf{B}_{31} & \mathbf{B}_{32} & \mathbf{1} \end{bmatrix}}_{\mathbf{N}_l} \underbrace{\begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_3 \end{bmatrix}}_{\mathbf{N}_d} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{3.31}$$

where all terms have been previously defined. Now, the constraint wrenches $\mathbf{w}^C$ do not develop any power, and hence, $\mathbf{t}^T\mathbf{w}^C$ is 0; by virtue of eq.(3.30), $\mathbf{w}^C$ lies in the nullspace of $\mathbf{N}_d^T\mathbf{N}_l^T$. To eliminate joint constraint wrenches, we pre-multiply both sides of eq.(3.29) by $\mathbf{N}_d^T\mathbf{N}_l^T$, and noting that, for planar manipulators $\dot{\mathbf{M}} = \mathbf{O}$,

$$\mathbf{N}_d^T\mathbf{N}_l^T\mathbf{M}\dot{\mathbf{t}} = \tilde{\boldsymbol{\tau}} + \mathbf{N}_d^T\mathbf{N}_l^T\mathbf{w}^G \tag{3.32}$$

Time differentiating eq.(3.30) and substituting the expression for $\dot{\mathbf{t}}$ in eq.(3.32),

$$\mathbf{N}_d^T\mathbf{N}_l^T[\mathbf{M}\mathbf{N}_l\mathbf{N}_d\ddot{\boldsymbol{\theta}} + (\mathbf{M}\dot{\mathbf{N}}_l\mathbf{N}_d + \mathbf{M}\mathbf{N}_l\dot{\mathbf{N}}_d)\dot{\boldsymbol{\theta}}] = \tilde{\boldsymbol{\tau}} + \mathbf{N}_d^T\mathbf{N}_l^T\mathbf{w}^G \tag{3.33}$$

where $\tilde{\boldsymbol{\tau}} = \mathbf{N}_d^T\mathbf{N}_l^T\mathbf{w}^{AC}$ is the joint torque vector for the chain and is given by $\tilde{\boldsymbol{\tau}} = \begin{bmatrix} \tilde{\tau}_1 & \tilde{\tau}_2 & \tilde{\tau}_3 \end{bmatrix}^T$ for *each open chain.* We can write the above equation in compact form as

$$\mathbf{I}\ddot{\boldsymbol{\theta}} + \mathbf{C}\dot{\boldsymbol{\theta}} = \tilde{\boldsymbol{\tau}} + \boldsymbol{\tau}^G \tag{3.34}$$

where $\mathbf{I} = \mathbf{N}_d^T\mathbf{N}_l^T\mathbf{M}\mathbf{N}_l\mathbf{N}_d$ and $\mathbf{C} = \mathbf{N}_d^T\mathbf{N}_l^T(\mathbf{M}\dot{\mathbf{N}}_l\mathbf{N}_d + \mathbf{M}\mathbf{N}_l\dot{\mathbf{N}}_d)$, $\mathbf{I}$ being the *generalized inertia matrix of the chain* and $\mathbf{C}$ the matrix of coriolis and centrifugal forces.

The inverse dynamics analysis for a *fully-redundant* manipulator i.e., all joints actuated is now complete. Note that the joint torques obtained for each chain would produce compatible twists for each part of the platform because the platform was cut and its twists from all section are equal. One important point is the distribution of working wrench $\mathbf{w}^W$. Its distribution between open-chains is critical and affects the computed joint torques for each chain. Such a distribution of the working wrench may be accomplished based on many alternative methods proposed in the literature, including the one based on the condition number of the Jacobian of each chain for which a chain with a better condition number would bear higher loads. This should provide maximum-possible dexterity to the manipulator at any pose.

In case of systems without redundant actuation, the distribution of the working wrench is not important, as the torques evaluated at this stage are projected onto the minimum actuated joint space in the step that follows. We will discuss the case where the working wrench is assumed to have been distributed evenly among the subsystems.

Projecting Joint Torques onto Minimal-Coordinate Space. As a second step, we write the dynamics equation for *each chain* and couple them with Lagrange multipliers, thereby obtaining the dynamics equation of the whole manipulator as

$$\begin{bmatrix} [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta}]^{I} \\ [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta}]^{II} \\ [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta}]^{III} \end{bmatrix} = \begin{bmatrix} \tau^{I} \\ \tau^{II} \\ \tau^{III} \end{bmatrix} + \begin{bmatrix} \tau_G^{I} \\ \tau_G^{II} \\ \tau_G^{III} \end{bmatrix} - \mathbf{A}^T\lambda \tag{3.35}$$

where $\mathbf{A}$ is the *loop-closure constraint Jacobian*, which appears in the constraints in the form

$$\mathbf{A}\dot{\theta} = 0$$

Now, by noticing that $\dot{\theta} = \mathbf{J}\dot{\theta}_{AC}$, it is clear that $\mathbf{J}$ lies in the nullspace of $\mathbf{A}$ and may be called the *loop-closure orthogonal complement.* Pre-multiplying both sides of eq.(3.35) by $\mathbf{J}^T$ we obtain:

$$\mathbf{J}^T \begin{bmatrix} [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta} - \tau_G]^{I} \\ [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta} - \tau_G]^{II} \\ [\mathbf{I}\ddot{\theta} + \mathbf{C}\dot{\theta} - \tau_G]^{III} \end{bmatrix} = \tau_{AC} \tag{3.36}$$

Notice that the bracketed terms are nothing but $\tilde{\tau}^j$, which can be found for *each open chain*, for $j = I, II$ and III, recursively (Saha, 1999). We may therefore write eq.(3.36) as

$$\mathbf{J}^T \begin{bmatrix} \tilde{\tau}^I \\ \tilde{\tau}^{II} \\ \tilde{\tau}^{III} \end{bmatrix} = \tau_{AC} \tag{3.37}$$

which is the relation sought, where $\mathbf{J}$ is given by eq.(3.15) i.e.

$$\mathbf{J} = \bar{\mathbf{P}}\mathbf{L}\mathbf{B}\mathbf{P} \tag{3.38}$$

The right hand side of the above equation can be re-written in slightly different expanded form as

$$\begin{bmatrix} \bar{\mathbf{P}}^I & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \bar{\mathbf{P}}^{II} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \bar{\mathbf{P}}^{III} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{O} & \mathbf{O} \\ (\mathbf{S}^I - 1) & \mathbf{S}^{II} & \mathbf{S}^{III} \\ (\mathbf{S}^I - 1) & \mathbf{S}^{II} & \mathbf{S}^{III} \\ \mathbf{O} & 1 & \mathbf{O} \\ \mathbf{S}^I & (\mathbf{S}^{II} - 1) & \mathbf{S}^{III} \\ \mathbf{S}^I & (\mathbf{S}^{II} - 1) & \mathbf{S}^{III} \\ \mathbf{O} & \mathbf{O} & 1 \\ \mathbf{S}^I & \mathbf{S}^{II} & (\mathbf{S}^{III} - 1) \\ \mathbf{S}^I & \mathbf{S}^{II} & (\mathbf{S}^{III} - 1) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{p}}_1^I & 0 & 0 \\ 0 & \tilde{\mathbf{p}}_1^{II} & 0 \\ 0 & 0 & \tilde{\mathbf{p}}_1^{III} \end{bmatrix} \tag{3.39}$$

where $\mathbf{S}^j = \mathbf{K}^{-1}\mathbf{\Phi}_2^j$ for $j = I, II$ and III. Substituting $\mathbf{J}$ into eq.(3.37) and rearranging

$$\tau_{AC} = \begin{bmatrix} (\tilde{\mathbf{p}}_1^I)^T[(\mathbf{S}^I)^T(\hat{\mathbf{p}}_2^I + \hat{\mathbf{p}}_3^I + \hat{\mathbf{p}}_2^{II} + \hat{\mathbf{p}}_3^{II} + \hat{\mathbf{p}}_2^{III} + \hat{\mathbf{p}}_3^{III}) + \hat{\mathbf{p}}_1^I - \hat{\mathbf{p}}_2^I - \hat{\mathbf{p}}_3^I] \\ (\tilde{\mathbf{p}}_1^{II})^T[(\mathbf{S}^{II})^T(\hat{\mathbf{p}}_2^I + \hat{\mathbf{p}}_3^I + \hat{\mathbf{p}}_2^{II} + \hat{\mathbf{p}}_3^{II} + \hat{\mathbf{p}}_2^{III} + \hat{\mathbf{p}}_3^{III}) + \hat{\mathbf{p}}_1^{II} - \hat{\mathbf{p}}_2^{II} - \hat{\mathbf{p}}_3^{II}] \\ (\tilde{\mathbf{p}}_1^{III})^T[(\mathbf{S}^{III})^T(\hat{\mathbf{p}}_2^I + \hat{\mathbf{p}}_3^I + \hat{\mathbf{p}}_2^{II} + \hat{\mathbf{p}}_3^{II} + \hat{\mathbf{p}}_2^{III} + \hat{\mathbf{p}}_3^{III}) + \hat{\mathbf{p}}_1^{III} - \hat{\mathbf{p}}_2^{III} - \hat{\mathbf{p}}_3^{III}] \end{bmatrix}$$

where $\tau_{AC} = [\tau_1^I,\ \tau_1^{II},\ \tau_1^{III}]^T$ and $\hat{\mathbf{p}}_k^j = [\tilde{\tau}_k \mathbf{\Psi}_{k-1}\tilde{\mathbf{p}}_k/\delta_k]^j$ for $k = 1,\ 2$ and 3 and $j = I,\ II$ and III, where $\mathbf{\Psi}_0$ and $\mathbf{\Psi}_1$ are the three-dimensional identity matrices.

But $\tilde{\mathbf{p}}_1^T \hat{\mathbf{p}}_1 = \tilde{\tau}_1$ and hence the above equation is written finally as

$$\tau_{AC} = \begin{bmatrix} \tau_1^I \\ \tau_1^{II} \\ \tau_1^{III} \end{bmatrix} = \begin{bmatrix} \tilde{\tau}_1^I + (\tilde{\mathbf{p}}_1^I)^T[(\mathbf{S}^I)^T(\bar{\mathbf{p}}^I + \bar{\mathbf{p}}^{II} + \bar{\mathbf{p}}^{III}) - \bar{\mathbf{p}}^I] \\ \tilde{\tau}_1^{II} + (\tilde{\mathbf{p}}_1^{II})^T[(\mathbf{S}^{II})^T(\bar{\mathbf{p}}^I + \bar{\mathbf{p}}^{II} + \bar{\mathbf{p}}^{III}) - \bar{\mathbf{p}}^{II}] \\ \tilde{\tau}_1^{III} + (\tilde{\mathbf{p}}_1^{III})^T[(\mathbf{S}^{III})^T(\bar{\mathbf{p}}^I + \bar{\mathbf{p}}^{II} + \bar{\mathbf{p}}^{III}) - \bar{\mathbf{p}}^{III}] \end{bmatrix} \tag{3.40}$$

where $\bar{\mathbf{p}} = \hat{\mathbf{p}}_2 + \hat{\mathbf{p}}_3$ for corresponding chains.

Distribution of Inverse Dynamics. We briefly outline the distribution of the inverse dynamics of the 3 RRR planar parallel mechanism in this subsection.

(i) Nodes: From eq.(3.32) it is clear that

$$\tilde{\tau} = \mathbf{N}_d^T \mathbf{N}_l^T (\mathbf{M}\dot{\mathbf{t}} + \mathbf{w}^G)$$

which can be calculated recursively for each chain as follows:

$$\begin{aligned} \gamma_3 &= (\mathbf{M}_3\dot{\mathbf{t}}_3 + \mathbf{w}_3^G) \\ \gamma_2 &= (\mathbf{M}_2\dot{\mathbf{t}}_2 + \mathbf{w}_2^G) + \mathbf{B}_{32}^T\gamma_3 \\ \gamma_1 &= (\mathbf{M}_1\dot{\mathbf{t}}_1 + \mathbf{w}_1^G) + \mathbf{B}_{21}^T\gamma_2 \\ \tilde{\tau}_3 &= \mathbf{p}_3^T\gamma_3 \\ \tilde{\tau}_2 &= \mathbf{p}_2^T\gamma_2 \\ \tilde{\tau}_1 &= \mathbf{p}_1^T\gamma_1 \end{aligned}$$

Now we calculate $\bar{\mathbf{p}}$:

$$\bar{\mathbf{p}} = \tilde{\tau}_2\frac{\tilde{\mathbf{p}}_2}{\delta_2} + \tilde{\tau}_3\frac{\Psi_2\tilde{\mathbf{p}}_3}{\delta_3}$$

(ii) Central: Add all $\bar{\mathbf{p}}$ from each chain

(iii) Nodes: Calculate actuated joint torques:

Chain I:

$$\tau_1^I = \tilde{\tau}_1^I + (\tilde{\mathbf{p}}_1^I)^T[(\mathbf{S}^I)^T(\bar{\mathbf{p}}^I + \bar{\mathbf{p}}^{II} + \bar{\mathbf{p}}^{III}) - \bar{\mathbf{p}}^I]$$

Chain *II*:

$$\tau_1^{II} = \tilde{\tau}_1^{II} + (\tilde{\mathrm{p}}_1^{II})^T[(\mathrm{S}^{II})^T(\bar{\mathrm{p}}^I + \bar{\mathrm{p}}^{II} + \bar{\mathrm{p}}^{III}) - \bar{\mathrm{p}}^{II}]$$

Chain *III*:

$$\tau_1^{III} = \tilde{\tau}_1^{III} + (\tilde{\mathrm{p}}_1^{III})^T[(\mathrm{S}^{III})^T(\bar{\mathrm{p}}^I + \bar{\mathrm{p}}^{II} + \bar{\mathrm{p}}^{III}) - \bar{\mathrm{p}}^{III}]$$

3.3. Forward Dynamics

Now we obtain expressions for the forward dynamics of the manipulator. Rewriting eq.(3.36) in a slightly different form we obtain:

$$\mathbf{J}^T(\underbrace{\begin{bmatrix} \mathrm{I}^I & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{I}^{II} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{I}^{III} \end{bmatrix}}_{\bar{\mathrm{I}}} \begin{bmatrix} \ddot{\theta}^I \\ \ddot{\theta}^{II} \\ \ddot{\theta}^{III} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathrm{C}^I & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{C}^{II} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & C^{III} \end{bmatrix}}_{\bar{\mathrm{C}}} \begin{bmatrix} \dot{\theta}^I \\ \dot{\theta}^{II} \\ \dot{\theta}^{III} \end{bmatrix} - \underbrace{\begin{bmatrix} \tau_G^I \\ \tau_G^{II} \\ \tau_G^{III} \end{bmatrix}}_{\bar{\tau}_G}) = \tau_{AC}$$

$$\mathbf{J}^T(\bar{\mathbf{I}}\ddot{\boldsymbol{\theta}} + \bar{\mathbf{C}}\dot{\boldsymbol{\theta}} - \bar{\tau}_G) = \tau_{AC}$$

Moreover, $\dot{\boldsymbol{\theta}} = \mathbf{J}\dot{\boldsymbol{\theta}}_{AC}$, and hence $\ddot{\boldsymbol{\theta}} = \mathbf{J}\ddot{\boldsymbol{\theta}}_{AC} + \dot{\mathbf{J}}\dot{\boldsymbol{\theta}}_{AC}$. Substituting these into the above equation, we obtain:

$$(\mathbf{J}^T\bar{\mathbf{I}}\mathbf{J})\ddot{\boldsymbol{\theta}}_{AC} = -\mathbf{J}^T(\bar{\mathbf{I}}\dot{\mathbf{J}}\dot{\boldsymbol{\theta}}_{AC} + \bar{\mathbf{C}}\mathbf{J}\dot{\boldsymbol{\theta}}_{AC} - \bar{\tau}_G) + \tau_{AC} \tag{3.41}$$

which is the minimal-order dynamics equation sought. The left hand side matrix coefficient is *generalized inertia matrix* of the manipulator.

The right hand side of the equation may be gathered in a single vector $\bar{\tau}$ and may be computed recursively by using the above inverse dynamics algorithm for $\ddot{\boldsymbol{\theta}}_{AC} = \mathbf{0}$ as originally suggested by Walker and Orin (1982). Each diagonal block of $\bar{\mathbf{I}}$ is the *generalized inertia matrix of each serial chain* and may be computed recursively, as already suggested by Saha (1999) using DeNOC. The *loop-closure* *orthogonal complement* $\mathbf{J} = \bar{\mathbf{P}}\mathbf{L}\mathbf{B}\mathbf{P}$ is, in turn,

$$\mathbf{J} = \underbrace{\begin{bmatrix} \bar{\mathbf{P}}^{I} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \bar{\mathbf{P}}^{II} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \bar{\mathbf{P}}^{III} \end{bmatrix} \begin{bmatrix} \mathbf{L}^{I} \\ \mathbf{L}^{II} \\ \mathbf{L}^{III} \end{bmatrix}}_{\mathbf{T}} \underbrace{\mathbf{B}\mathbf{P}}_{\bar{\mathbf{B}}}$$

or

$$\mathbf{J} = \begin{bmatrix} \mathbf{T}^{I} \\ \mathbf{T}^{II} \\ \mathbf{T}^{III} \end{bmatrix} \bar{\mathbf{B}}$$

Substituting $\mathbf{J}$ into eq.(3.41), we obtain:

$$\bar{\mathbf{B}}^{T} \begin{bmatrix} (\mathbf{T}^{I})^{T} & (\mathbf{T}^{II})^{T} & (\mathbf{T}^{III})^{T} \end{bmatrix} \begin{bmatrix} \mathbf{I}^{I} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}^{II} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{I}^{III} \end{bmatrix} \begin{bmatrix} \mathbf{T}^{I} \\ \mathbf{T}^{II} \\ \mathbf{T}^{III} \end{bmatrix} \bar{\mathbf{B}}\ddot{\boldsymbol{\theta}}_{AC} = \bar{\boldsymbol{\tau}}$$

Or

$$\bar{\mathbf{B}}^{T}\left(\sum_{i=I}^{III}[\mathbf{T}^{T}\mathbf{I}\mathbf{T}]^{i}\right)\bar{\mathbf{B}}\ddot{\boldsymbol{\theta}}_{AC} = \bar{\boldsymbol{\tau}} \tag{3.42}$$

The terms in the parentheses are the contributions from the separate chains and can be distributed. The 9×9 matrix $[\mathbf{T}^{T}\mathbf{I}\mathbf{T}]^{j}$, for $j = I,\ II$ and III, is written in block form as

$$[\mathbf{T}^{T}\mathbf{I}\mathbf{T}]^{j} = \begin{bmatrix} \mathbf{L}_{11} & & \mathrm{sym} \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} \end{bmatrix}^{j} \tag{3.43}$$

where each $\mathbf{L}_{(k,l)}$ is a 3×3 block; it is given below for each chain:

<u>Chain *I*</u>

$$\mathbf{L}_{11}^{I} = \mathbf{I}_{1,1}^{I}\mathbf{a}_{1}^{I}(\mathbf{a}_{1}^{I})^{T} + \mathbf{a}_{1}^{I}(\hat{\mathbf{a}}_{1}^{I})^{T}\mathbf{A}^{I} + (\mathbf{A}^{I})^{T}\hat{\mathbf{a}}_{1}^{I}(\mathbf{a}_{1}^{I})^{T} + (\mathbf{A}^{I})^{T}\bar{\mathbf{A}}^{I}\mathbf{A}^{I}$$

$$\mathbf{L}_{21}^{I} = (\mathbf{S}^{II})^{T}[\hat{\mathbf{a}}_{1}^{I}(\mathbf{a}_{1}^{I})^{T} + \bar{\mathbf{A}}^{I}\mathbf{A}^{I}]$$

$$
\begin{aligned}
L_{22}^{I} &= (S^{II})^T \bar{A}^I S^{II} \\
L_{31}^{I} &= (S^{III})^T [\hat{a}_1^I (a_1^I)^T + \bar{A}^I A^I] \\
L_{32}^{I} &= (S^{III})^T \bar{A}^I S^{II} \\
L_{33}^{I} &= (S^{III})^T \bar{A}^I S^{III}
\end{aligned}
$$

<u>Chain II</u>

$$
\begin{aligned}
L_{11}^{II} &= (S^{I})^T \bar{A}^{II} S^{I} \\
L_{21}^{II} &= [a_1^{II} (\hat{a}_1^{II})^T + (A^{II})^T \bar{A}^{II}] S^I \\
L_{22}^{II} &= I_{1,1}^{II} a_1^{II} (a_1^{II})^T + a_1^{II} (\hat{a}_1^{II})^T A^{II} + (A^{II})^T \hat{a}_1^{II} (a_1^{II})^T + (A^{II})^T \bar{A}^{II} A^{II} \\
L_{31}^{II} &= (S^{III})^T \bar{A}^{II} S^{I} \\
L_{32}^{II} &= (S^{III})^T [\hat{a}_1^{II} (a_1^{II})^T + \bar{A}^{II} A^{II}] \\
L_{33}^{II} &= (S^{III})^T \bar{A}^{II} S^{III}
\end{aligned}
$$

<u>Chain III</u>

$$
\begin{aligned}
L_{11}^{III} &= (S^{I})^T \bar{A}^{III} S^{I} \\
L_{21}^{III} &= (S^{II})^T \bar{A}^{III} S^{I} \\
L_{22}^{III} &= (S^{II})^T \bar{A}^{III} S^{II} \\
L_{31}^{III} &= [a_1^{III} (\hat{a}_1^{III})^T + (A^{III})^T \bar{A}^{III}] S^I \\
L_{32}^{III} &= [a_1^{III} (\hat{a}_1^{III})^T + (A^{III})^T \bar{A}^{III}] S^{II} \\
L_{33}^{III} &= I_{1,1}^{III} a_1^{III} (a_1^{III})^T + a_1^{III} (\hat{a}_1^{III})^T A^{III} + (A^{III})^T \hat{a}_1^{III} (a_1^{III})^T \\
&\quad + (A^{III})^T \bar{A}^{III} A^{III}
\end{aligned}
$$

where $a_k^j = [\Psi_{(k-1)} \tilde{p}_k / \delta_k]^j$, $\hat{a}_k^j = [I_{(2,k)} a_2^T + I_{(3,k)} a_3^T]^j$, $\bar{A}^j = [a_2 \hat{a}_2^T + a_3 \hat{a}_3^T]^j$ and $A^j = (S^j - 1)$ for $k = 1, 2$ and 3 and $j = I, II$ and III. Further, matrix $\bar{B}$ is block-diagonal, namely,

$$
\begin{aligned}
\bar{\mathbf{B}} &= \mathrm{diag}(\mathbf{B}^{I}_{P1}\mathbf{p}^{I}_{1}, \mathbf{B}^{II}_{P1}\mathbf{p}^{II}_{1}, \mathbf{B}^{III}_{P1}\mathbf{p}^{III}_{1}) \\
& \mathrm{diag}(\bar{\mathbf{b}}^{I}, \bar{\mathbf{b}}^{II}, \bar{\mathbf{b}}^{III})
\end{aligned} \tag{3.44}
$$

Substituting $\bar{\mathbf{B}}$ into eq.(3.42) we obtain

$$
\begin{bmatrix}
(\bar{\mathbf{b}}^{I})^T\mathbf{L}_{11}\bar{\mathbf{b}}^{I} & & \mathrm{sym} \\
(\bar{\mathbf{b}}^{II})^T\mathbf{L}_{21}\bar{\mathbf{b}}^{I} & (\bar{\mathbf{b}}^{II})^T\mathbf{L}_{22}\bar{\mathbf{b}}^{II} & \\
(\bar{\mathbf{b}}^{III})^T\mathbf{L}_{31}\bar{\mathbf{b}}^{I} & (\bar{\mathbf{b}}^{III})^T\mathbf{L}_{32}\bar{\mathbf{b}}^{II} & (\bar{\mathbf{b}}^{III})^T\mathbf{L}_{33}\bar{\mathbf{b}}^{III}
\end{bmatrix} \ddot{\boldsymbol{\theta}}_{AC} = \begin{bmatrix} \bar{\tau}_1 \\ \bar{\tau}_2 \\ \bar{\tau}_3 \end{bmatrix} = \bar{\tau} \tag{3.45}
$$

where $\mathbf{L}_{(k,l)} = \mathbf{L}^{I}_{(k,l)} + \mathbf{L}^{II}_{(k,l)} + \mathbf{L}^{III}_{(k,l)}$. Now we solve the above system for $\ddot{\boldsymbol{\theta}}_{AC}$ using the reverse Gaussian elimination technique, as suggested by Saha (1999) and Saha and Schiehlen (2001), to obtain:

$$
\ddot{\theta}^{I}_{1} = \frac{\hat{\hat{\tau}}_1}{\alpha_1} \tag{3.46}
$$

$$
\ddot{\theta}^{II}_{1} = \frac{\hat{\tau}_2 - (\mathbf{b}^{II})^T\hat{\mathbf{L}}_{21}\mathbf{b}^{I}\ddot{\theta}^{T}_{1}}{\alpha_2} \tag{3.47}
$$

$$
\ddot{\theta}^{III}_{1} = \frac{\bar{\tau}_3 - [(\mathbf{b}^{III})^T\mathbf{L}_{31}\mathbf{b}^{I}\ddot{\theta}^{I}_{1} + (\mathbf{b}^{III})^T\mathbf{L}_{32}\mathbf{b}^{II}\ddot{\theta}^{II}_{1}]}{\alpha_3} \tag{3.48}
$$

where

$$
\begin{aligned}
\alpha_3 &= (\mathbf{b}^{III})^T\mathbf{L}_{33}\mathbf{b}^{III} \\
\hat{\mathbf{L}}_{11} &= \mathbf{L}_{11} - \frac{\mathbf{L}^{T}_{31}\mathbf{b}^{III}}{\alpha_3}(\mathbf{b}^{III})^T\mathbf{L}_{31} \\
\hat{\mathbf{L}}_{21} &= \mathbf{L}_{21} - \frac{\mathbf{L}^{T}_{32}\mathbf{b}^{III}}{\alpha_3}(\mathbf{b}^{III})^T\mathbf{L}_{31} \\
\hat{\mathbf{L}}_{22} &= \mathbf{L}_{22} - \frac{\mathbf{L}^{T}_{32}\mathbf{b}^{III}}{\alpha_3}(\mathbf{b}^{III})^T\mathbf{L}_{32} \\
\hat{\tau}_1 &= \bar{\tau}_1 - (\mathbf{b}^{I})^T\frac{\mathbf{L}^{T}_{31}\mathbf{b}^{III}}{\alpha_3}\bar{\tau}_3
\end{aligned}
$$

$$
\begin{aligned}
\hat{\tau}_2 &= \bar{\tau}_2 - (\mathbf{b}^{II})^T \frac{\mathbf{L}_{32}^T \mathbf{b}^{III}}{\alpha_3} \bar{\tau}_3 \\
\alpha_2 &= (\mathbf{b}^{II})^T \hat{\mathbf{L}}_{22} \mathbf{b}^{II} \\
\hat{\hat{\tau}}_1 &= \hat{\tau}_1 - (\mathbf{b}^{I})^T \frac{\hat{\mathbf{L}}_{21}^T \mathbf{b}^{II}}{\alpha_2} \hat{\tau}_2 \\
\alpha_1 &= (\mathbf{b}^{I})^T \hat{\mathbf{L}}_{11} \mathbf{b}^{I} - (\mathbf{b}^{I})^T \frac{\hat{\mathbf{L}}_{21}^T \mathbf{b}^{II}}{\alpha_2} (\mathbf{b}^{II})^T
\end{aligned}
$$

thereby completing the analysis.

Distribution of Forward Dynamics. In a similar manner, the steps required for the distribution of the forward dynamics computations are summarized below:

(i) Central: Conduct a displacement analysis and obtain all joint angles for the actuated joint angles $\boldsymbol{\theta}_{AC}$ available.

(ii) Perform the distributed inverse dynamics for $\ddot{\boldsymbol{\theta}}_{AC} = \mathbf{0}$ of the platform, as discussed above, to obtain $\bar{\tau}$ for *each open chain.*

(iii) Node: Calculate $\mathbf{L}^j = [\mathbf{T}^T\mathbf{I}\mathbf{T}]^j$ for $j = I,\ II$ and III.

(iv) Central: Calculate the actuated joint accelerations $\ddot{\boldsymbol{\theta}}_{AC}$.

(v) Nodes: Integrate the actuated joint accelerations in corresponding chain nodes, to obtain the actuated joint angles and velocities.

However conducting a displacement analysis at each iteration is not advisable. The alternative, as mentioned previously, is to integrate all joint rates, which is done as described below:

(i) Conduct velocity analysis and obtain the unactuated joint rates for the actuated joint rates $\dot{\boldsymbol{\theta}}_{AC}$ available.

(ii) Perform the distributed inverse dynamics for $\ddot{\boldsymbol{\theta}}_{AC} = \mathbf{0}$ of the platform, as discussed above to obtain $\bar{\tau}$ for *each open chain.*

(iii) Node: Calculate $\mathbf{L}^j = [\mathbf{T}^T\mathbf{I}\mathbf{T}]^j$, for $j = I,\ II$ and III.

(iv) Central: Calculate the actuated joint accelerations $\ddot{\boldsymbol{\theta}}_{AC}$.

(v) Nodes: Integrate the actuated joint accelerations and *all* joint rates in corresponding chain nodes, to obtain the actuated joint velocities and *all* joint angles.

However, it should be noted that the distribution in this case *cannot be full.* i.e. communication is required between integration steps, data transfer not being possible solely as exchange of state information at each time step.

CHAPTER 4

Simulation Results

In the two previous chapters we examined the development and distributed implementation of *three* forward dynamics formulations: (a) the compliance based approach; (b) the numerical projection-based approach, with stabilization; and (c) the recursive decoupled natural orthogonal complement. These approaches, henceforth referred to as method A, B and C were applied to the 3 $\underline{R}RR$ planar parallel mechanism.

The presentation discussion of these results is the focus of this chapter.

4.0.1. Computation Environment. In each case, the model was converted into a suitable state-space form for use in conjunction with various numerical time-stepping algorithms, henceforth called ODE suite, designed to advance the state of the system from a specified initial state and create the time-histories desired for the forward-dynamics simulations.

MATLAB's Simulink offers a convenient tool for the implementation of the time stepping algorithms. In particular, Simulink offers an extensive set of various types of integration schemes in a user friendly format, which will therefore be the primary mode of implementation of our examples. Further details on the implementation of the ODE suite in MATLAB are available in (Shampine and Reichelt, 1997). Simulink enabled the development in block-diagram form. In conjunction with a toolbox called *Real-Time Workshop*, the high-level block-diagram description can be used to generate C-code which can then be compiled for efficient execution.

Finally, in conjunction with RT-Lab, a commercial software, the model developed was run in a distributed manner on four PCs connected together by either *ethernet* or *firewire* interconnects to form a computational cluster. In particular, RT-Lab provides the necessary glue to run the models in a distributed manner, while meeting strict deterministic performance and synchronization requirements.

Analysis of the Results. The results will be analyzed, discussed and compared based on several factors, including modularity, efficiency, accuracy and distribution.

Briefly, we also note that MATLAB's ODE suite offers two groups of integration schemes: (a) fixed time-stepping schemes, where the user can specify the size of the time step; and (b) adaptive time-stepping, where an estimate of the integration error is made, and the time step is adapted to keep this error below a specific tolerance level. In particular, in the latter case, the error in each state is estimated to ensure the $\mathbf{e}(i) <= \max(\text{RelTol} * |\mathbf{x}(i)|, \text{AbsTol}(i))$ condition for each component i of the state vector $\mathbf{x}$, where $\mathbf{e}$ is the error vector. The two quantities under user-control here, in addition to actual selection of the algorithm, are the value for the relative Tolerance (RelTol) and the absolute tolerance (AbsTol). In the results that follow, we will report on the results from the forward dynamics simulations with *both* fixed time-stepping and adaptive time stepping algorithms.

Adaptive-Time-Stepping Case. In this case, the relative tolerance was pre-specified and an adaptive time-stepping scheme was used for the simulation. The two primary metrics of performance evaluation for this case were: (a) extent of the constraint error and (b) number of iterations. Four different relative tolerances, varying in orders of magnitude from 10e-3 to 10e-6, were examined in this case. It is important to note that adaptive time-stepping schemes are not particularly suitable for parallel/distributed processing, owing to the uncertainty of time required by each iteration in an integration scheme to converge within the desired tolerances. Specifically, these uncertainties create challenges for the synchronization of information exchange among multiple processors, which should occur at the end of each time-step. Hence, adaptive schemes will be avoided from the viewpoint of actual distributed implementation.

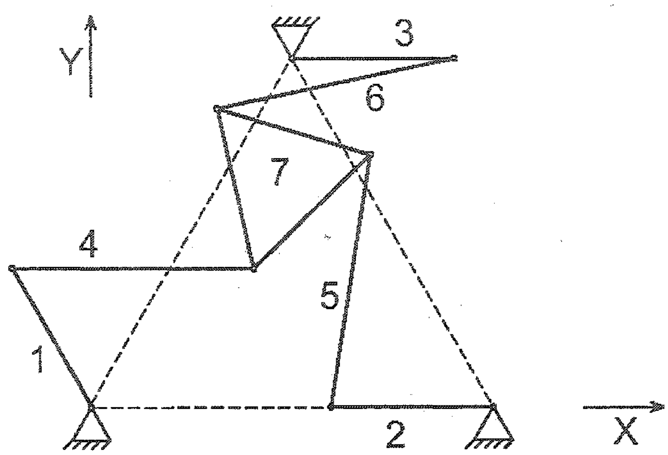


Figure 4.1: The three-dof planar parallel manipulator used in the example.

However, testing the models with adaptive time-stepping methods can give insight into the overall characteristics of a formulation, including: (a) formulation stiffness and computational complexity of implementation, as measured by the number of iterations or the total time taken to simulate a fixed simulation time. Hence, what follows, all results from the adaptive time-stepping cases have been computed using a distributed formulation, but as subsystems on a single processor to eliminate the need for explicit synchronization.

Fixed Time-Stepping Case. In this case, the principal parameter that can be selected by the user, in addition to the actual algorithm from the ODE suite, is the step size of fixed time step. This selection has critical implications in that an order of magnitude reduction in step-size increases the number of iterations by the same order of magnitude. The principal metric for evaluating the effectiveness of reducing the time step, and thereby slowing the computation by increasing the number of iterations, will be the actuated joint-angle errors between the prescribed and the simulated joint-trajectories.

4.0.2. Parameters and Initial Conditions. We use the same parameters for the planar parallel RRR manipulator as in Ma and Angeles (1989) shown in

Link i	L_i(m)	m_i(kg)	I_i(Kg m2)
1,2,3	0.4	3.0	0.04
4,5,6	0.6	4.0	0.12
7	0.4	8.0	0.0817

Table 4.1: Dimension and inertia properties

Fig. 4.1. The end effector, labelled 7, has the shape of an equilateral triangle, with sides of length l_7, links 1,2 and 3 have a length l_1, links 4, 5 and 6 have length l_4, and the three fixed revolute joints form an equilateral triangle with sides of length l_0. The prescribed motion drivers given by Ma and Angeles (1989) were:

$$\begin{aligned}
\theta_1 &= \frac{1}{3}\pi + \frac{1}{6}(\frac{2\pi t}{T} - \sin\frac{2\pi t}{T}) \\
\theta_2 &= \frac{4}{3}\pi + \frac{1}{6}(\frac{2\pi t}{T} - \sin\frac{2\pi t}{T}) \\
\theta_3 &= \frac{11}{6}\pi + \frac{1}{12}(\frac{2\pi t}{T} - \sin\frac{2\pi t}{T})
\end{aligned}$$

where $T = 3$s; the equations have been modified to measure against the x-axis. However, *these initial conditions are not sufficient to define the initial posture of the manipulator.* We thus use the initial configuration given by Geike and McPhee (2002):

$$\begin{aligned}
\theta_1 &= \tfrac{1}{3}\pi \text{ rad} & \theta_4 &= -0.865 \text{ rad} & \mathrm{x}_7 &= 0.728 \text{ m} \\
\theta_2 &= \tfrac{4}{3}\pi \text{ rad} & \theta_5 &= -2.102 \text{ rad} & \mathrm{y}_7 &= 0.233 \text{ m} \\
\theta_3 &= \tfrac{11}{6}\pi \text{ rad} & \theta_6 &= -0.976 \text{ rad} & \theta_7 &= 3.916 \text{ rad}
\end{aligned}$$

The parameters of the manipulator are given in Table 4.1, gravity acts in the $-Y$ direction.

4.0.3. Inverse Dynamics. We first perform the inverse dynamics in order to compute a time history of actuation forces that would realize the prescribed motions. Using the above parameters, the resulting torques for the actuated joints 1, 2 and 3 were evaluated using the inverse dynamics model discussed in the previous chapter. The resulting set of torques which realize the prescribed motions is shown in Fig. 4.2, which tally with those given by Ma and Angeles (1989).

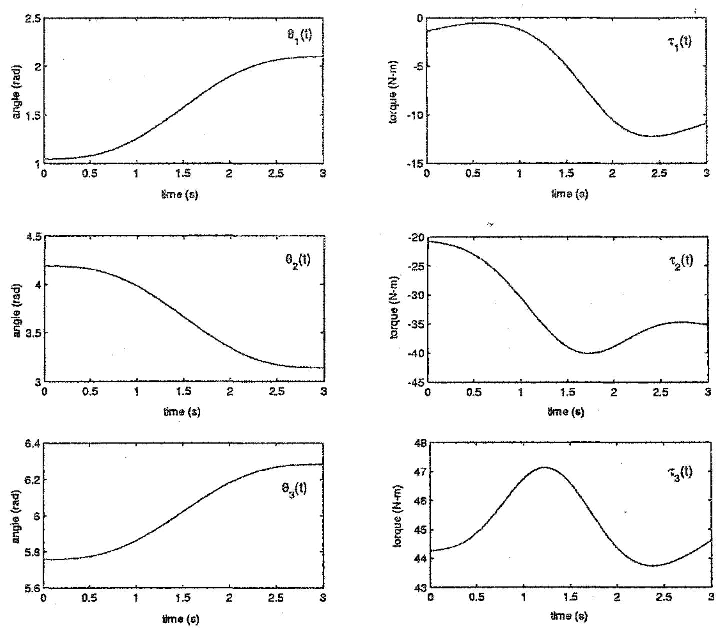


Figure 4.2: Desired trajectory and required driving torques.

The motion of the manipulator, using these driving torques inputs, was simulated for all the three models, the results being reported below. Both *adaptive-time-stepping* and *fixed-time-stepping* schemes were examined and discussed in subsequent sections for each model.

4.1. Penalty-Based Approach

As noted previously, the joint error between the prescribed joint motion trajectory and the actually simulated joint trajectories is selected to be representative of the error in the forward-dynamics simulation. In order to suppress the oscillations we included dampers parallel to each spring. Hence, the approximation was now obtained using the equation

$$\boldsymbol{\lambda} = k\mathbf{c} + d\mathbf{A}\dot{\mathbf{q}} \tag{4.1}$$

where $d = 2\sqrt{k}$ was fixed. Note, however, that inclusion of dampers improves the problem of stiffness; however, damping has its disadvantages, namely, (a) an extra parameter has to be tuned and (b) dampers dissipate energy, and thus, lead to inaccuracies.

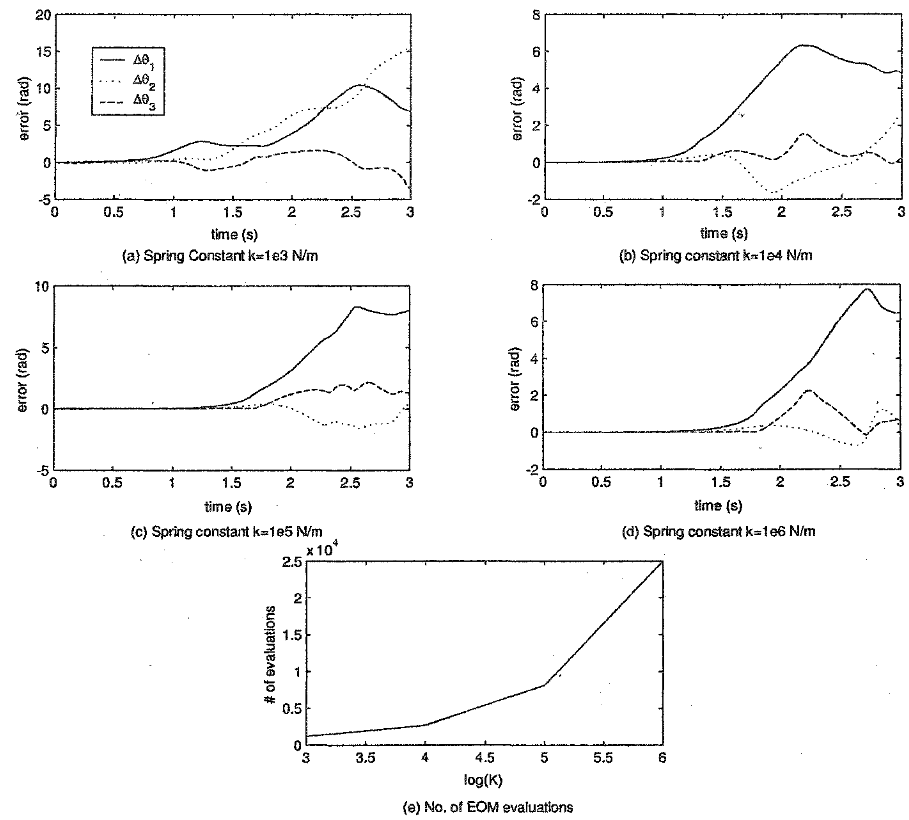


Figure 4.3: Simulation results using the penalty-based approach (adaptive time stepping, with relative tolerance = 1e-3): (a–d) joint-angle errors between desired and actual joint trajectories, for increasing values of the spring constant k; (e) number of EOM evaluations.

Figure 4.3a–d shows the resulting joint angle errors between the desired and the simulated motion trajectories for values of k ranging from 1e3 to 1e6 N/m , for *adaptive-time-stepping* with a relative tolerance of 1e-3. Figure 4.3e shows the number of evaluations of the equation of motion for a full 3-s simulation. Figure 4.4 shows the same, for adaptive time stepping with a relative tolerance of 1e-6.

In all cases, we note the poor overall performance of the penalty formulation for relatively large setting of the relative tolerance. The lack of sensitivity of the method to the actual compliance and damping parameters in terms of error reduction is noticeable. However, we do note that larger values of k tend to increase the number of iterations required to complete a given simulation time.

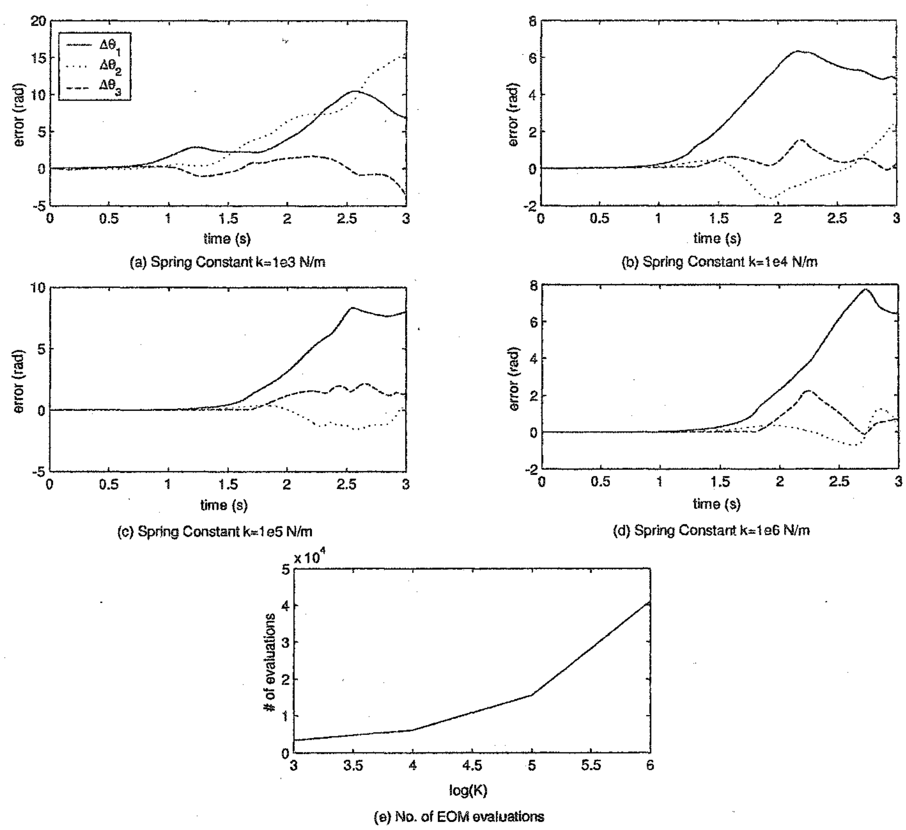
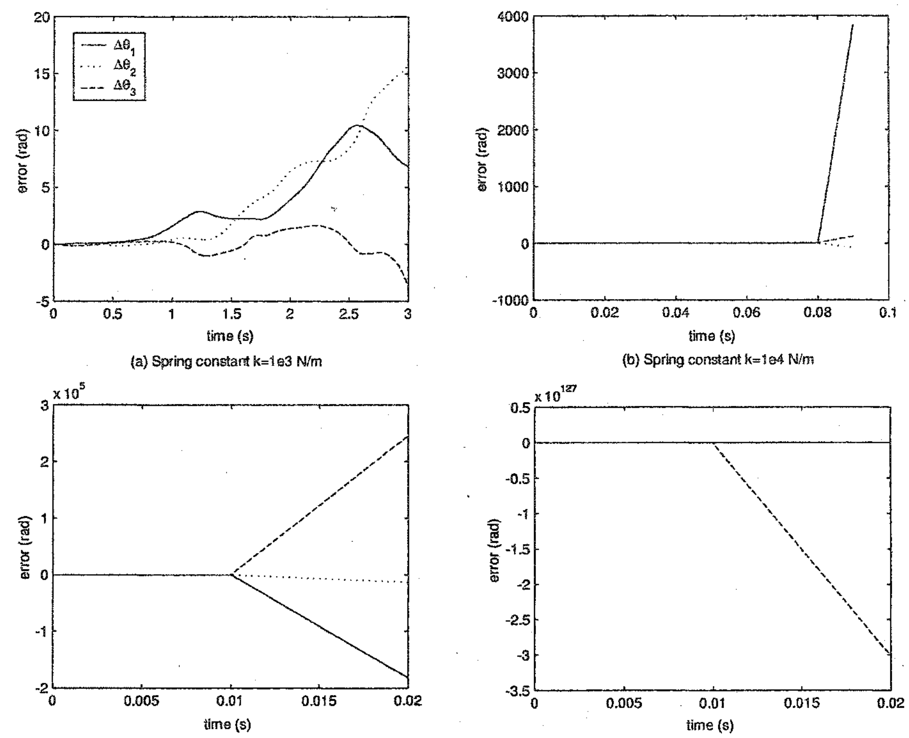


Figure 4.4: Simulation results using the penalty-based approach (adaptive time stepping, with relative tolerance = 1e-6): (a–d) joint-angle errors between desired and actual joint trajectories, for increasing values of the spring constant k; (e) number of EOM evaluations.

Figure 4.5 shows the resulting joint errors between the desired and the simulated motion trajectories, for a fixed time stepping of 0.01-s. We note specially that the model fails for $k = 1e4, 1e5$ and 1e6 N/m. Figure 4.6 depicts the same results for the fixed-time-stepping case with step size of of 0.001-s. and in this case the model fails only for $k = 1e6$ N/m.



(a) Spring constant k=1e3 N/m

(b) Spring constant k=1e4 N/m

(c) Spring constant k=1e5 N/m

(d) Spring constant k=1e6 N/m

Figure 4.5: Simulation results using the penalty-based approach with a fixed time-step of 0.01-s. Joint-angle errors, between desired and actual joint trajectories for spring constants of: (a) 1e3 N/m; (b) 1e4 N/m; (c) 1e5 N/m; and (d) 1e6 N/m.

In order to have a better picture of the accuracy of the model, we created a feedback compensation control scheme to force the simulated actuated joint trajectories to match their prescribed counterparts, as depicted in Fig. 4.7 for the case of k = 1e5 N/m and fixed time stepping of 0.001-s. Figure 4.7a shows the error between the desired trajectory and the trajectory obtained from simulation with feedback control. The additional torque correction required to force the actual motion close to the prescribed motion is now taken to be indicative of the error created in the computation of the forward dynamics model. Figure 4.7b shows the amount of percentage of torque correction required to obtain the desired trajectory, where we note that these values

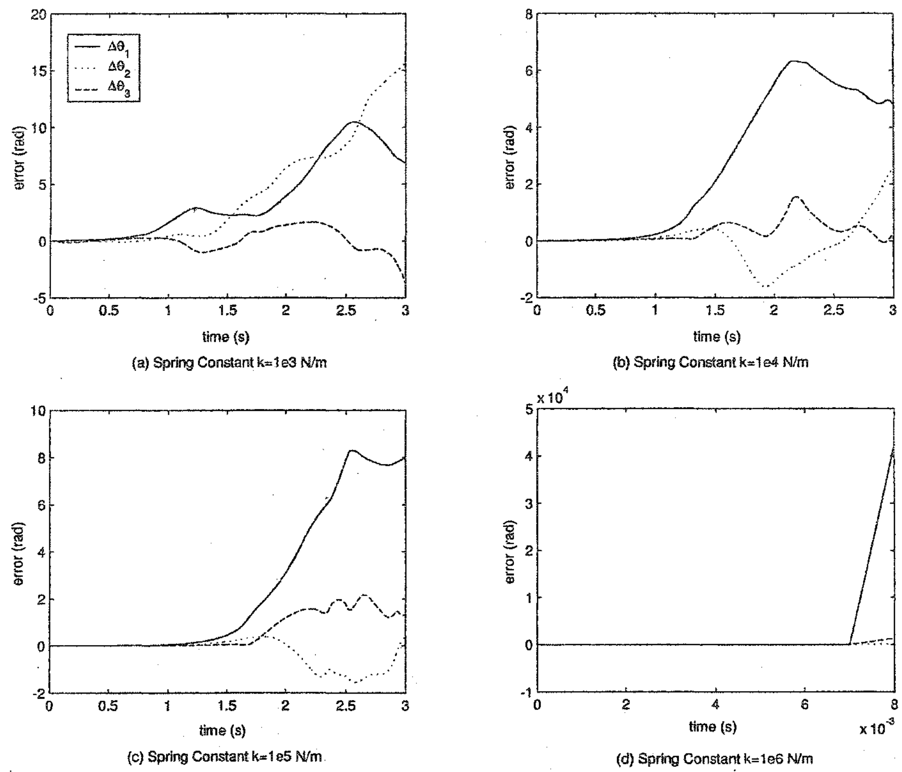


Figure 4.6: Simulation results using the penalty-based approach with a fixed time-step of 0.001-s. Joint-angle errors, between desired and actual joint trajectories for spring constants of: (a) 1e3 N/m; (b) 1e4 N/m; (c) 1e5 N/m; and (d) 1e6 N/m.

are within 15% of the input torque. Some general observations based on the above results are given below:

(i) The high number of iterations for variable time stepping implies that the method introduces stiffness in the dynamics model.

(ii) The additional parameters introduced in this formulation, i.e., spring constant k and damping factor d are sensitive parameters and have great influence on the overall dynamics.

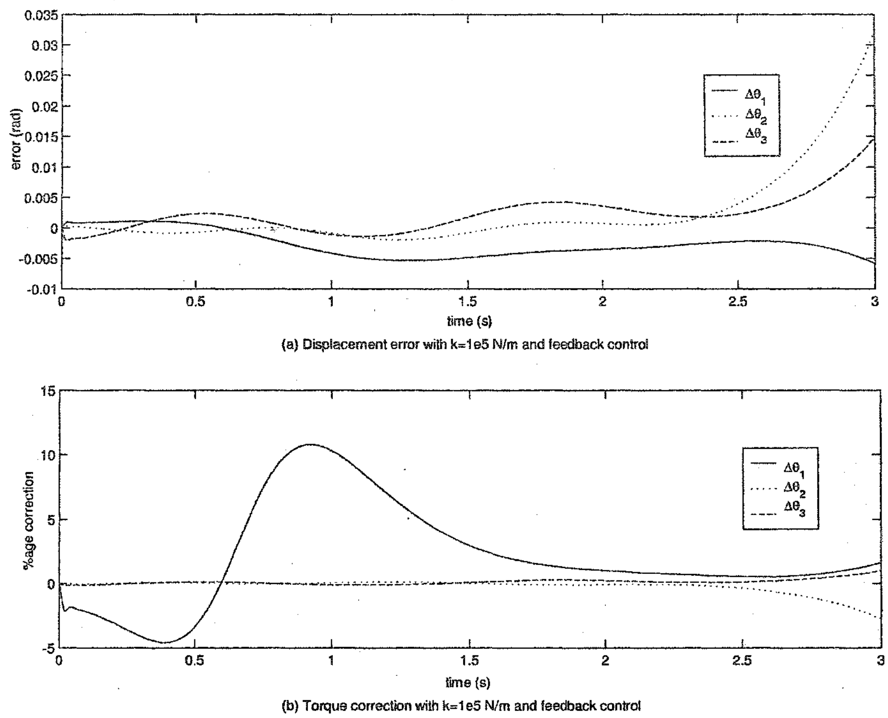
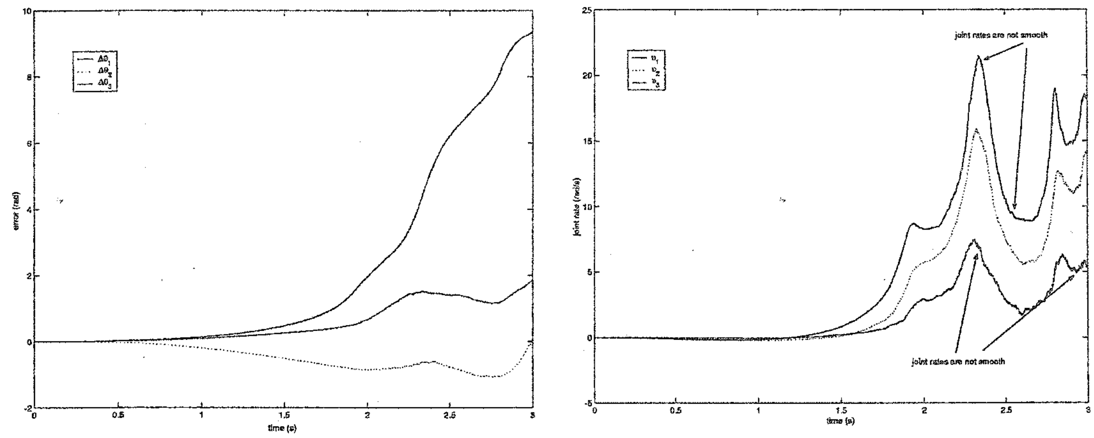


Figure 4.7: Simulation results using the penalty-based approach with feedback controller added and a fixed time-step of 0.001-s and k=1e5 N/m: (a) joint-angle errors between desired and actual joint trajectories; (b)percentage of torque correction.

(iii) The method tends to be marginally accurate based on the observations that the percentage of correction torques required to obtain relatively accurate trajectories were around 15%.

4.2. Loop-Closure Orthogonal Complement

The method suggested by Yun and Sarkar (1998) was used to obtain the orthogonal complement. Figure 4.8a shows the resulting error between desired and simulated motion trajectories for values of σ = 10, for adaptive time stepping with relative tolerance of 1e-3. We note that the number of evaluations of the equation of motion for a full 3-s simulation turned out to be 166132, which is unrealistically high. Upon further investigation, the source of the error becomes evident, and is best illustrated



(a) (b)

Figure 4.8: Simulation results using the Yun and Sarkar method, with adaptive time stepping and relative tolerance=1e-3: (a) joint angle errors between desired; and actual joint trajectories and (b) joint rates.

using Fig. 4.8b, where we note that the velocities are not smooth, there being a minor flaw in the recommended algorithm for determining the orthogonal complement. The said algorithm is reproduced below for quick reference:

(i) Obtain the orthogonal projector $\mathbf{H}$ onto the nullspace $\mathcal{N}(\mathbf{A})$ of $\mathbf{A}$, i.e., $\mathbf{H} = \mathbf{1} - \mathbf{A}^\dagger\mathbf{A}$, where $\mathbf{1}$ is the $n \times n$ identity matrix and $\mathbf{A}^\dagger$ is the Moore-Penrose generalized inverse of $\mathbf{A}(\mathbf{q})$.

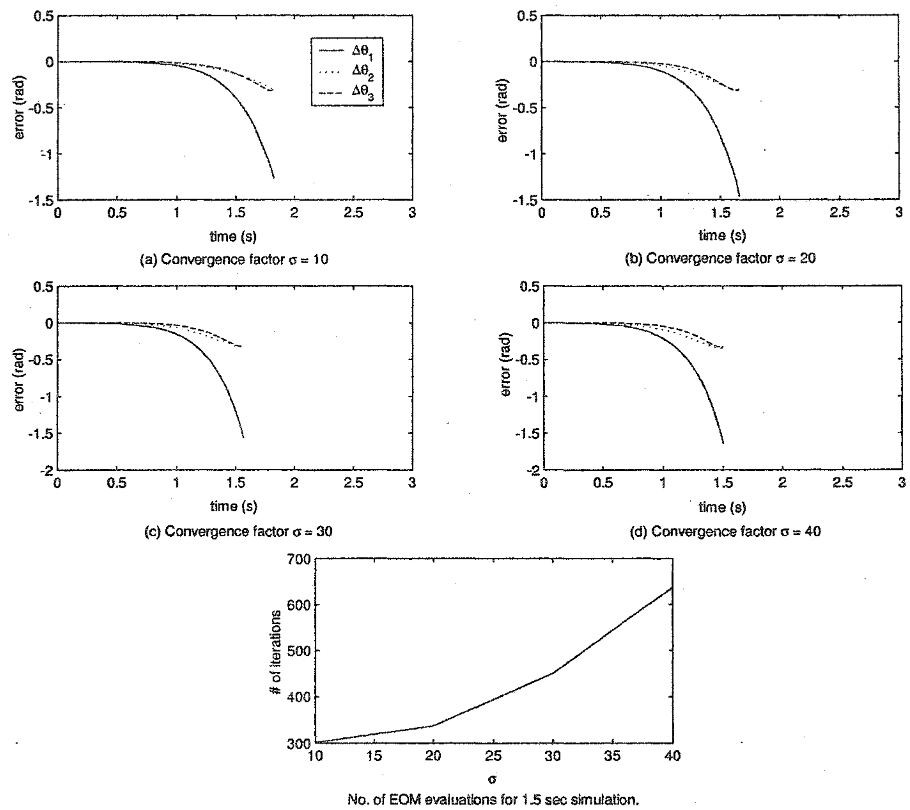
(ii) Compute the singular-value decomposition of $\mathbf{H}(\mathbf{q})$, i.e., $\mathbf{H} = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are $n \times n$ orthogonal matrices, and $\mathbf{\Delta}$ is the diagonal matrix containing the singular values of $\mathbf{H}(\mathbf{q})$.

(iii) Choose $\mathbf{S}(\mathbf{q})$ as the first $(n - m)$ columns of the $\mathbf{U}$ matrix.

The flaw in this algorithms lies in that, although the orthogonal complement ($\mathbf{S}$) found has a unique set of vectors that define a unique orthonormal basis, there is no guarantee that these vectors appear in the same column every time. The reason being that, for identical singular values, the corresponding columns of $\mathbf{U}$ and rows of

V may be switched. Careful observation of υ and $\dot{\upsilon}$ during simulation confirmed this claim. A sudden switching of two columns of the matrix S would naturally result in jump discontinuities in $\dot{\upsilon}$ and υ, the integration scheme thus diverging.

We also examined the use of some of the other methods for obtaining the orthogonal complement, as discussed in detail in (García de Jalón and Bayo, 1994) and outlined in the Appendix, such as the method based on *Gaussian Triangularization.* Figure 4.9a–d shows the resulting error between desired and simulated motion trajec-



(a) Convergence factor σ = 10

(b) Convergence factor σ = 20

(c) Convergence factor σ = 30

(d) Convergence factor σ = 40

No. of EOM evaluations for 1.5 sec simulation.

Figure 4.9: Simulation results using the loop-closure orthogonal complement by Gaussian triangularization with adaptive time stepping and relative tolerance=1e-6: (a–d) joint-angle-errors between desired and actual joint trajectories for varying values of convergence factor σ; (e) number of EOM evaluations.

tories for values of σ ranging from 10 to 40, for adaptive time stepping with relative tolerance of 1e-6. Figure 4.9e shows the number of evaluations of the equation of motion for the simulation time, where it is evident that the dynamics model is not stiff. A number of different relative tolerances were considered, but only the case of 1e-6 is reported here for conciseness. Similarly, Fig. 4.10 shows the resulting error

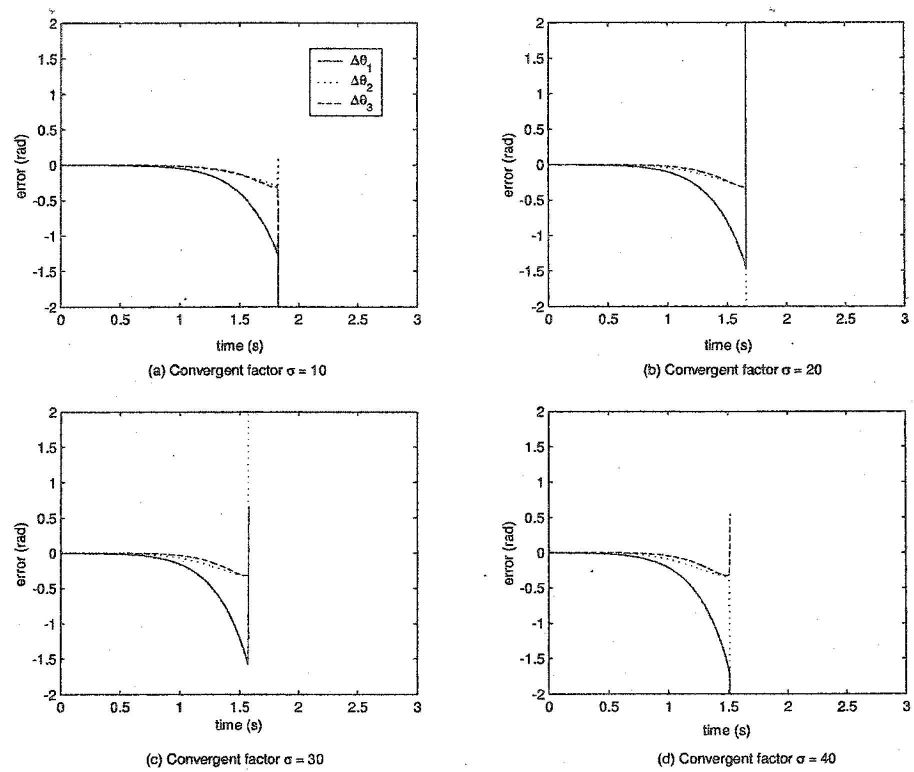
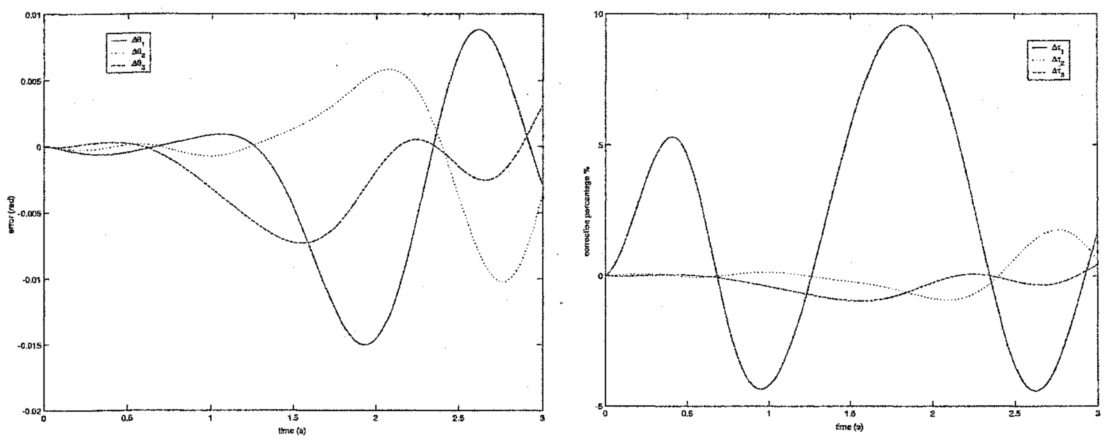


Figure 4.10: Simulation results using the loop-closure orthogonal complement by Gaussian triangularization with fixed time-step of 0.01-s. Joint-angle errors between the desired and actual joint trajectories for values of the convergence factor varying as: (a) $\sigma = 10$; (b) $\sigma = 20$; (c) $\sigma = 30$; and (d) $\sigma = 40$.

between desired and simulated motion trajectories, for *fixed time-stepping* of 0.01-s and is representative of the results for smaller step sizes.

Specifically we note that, in each case, the model broke because the manipulator reached a posture where, for the selected independent set of generalized coordinates, the 6×6 matrix $\mathbf{A}_d$ became singular to machine precision and could not be inverted.

Traditionally, this problem is overcome, by selecting an alternate set of independent coordinates so that matrix $\mathbf{A}_d$ *becomes invertible, resetting the integrators and continuing in the usual manner.*



(a) (b)

Figure 4.11: Joint-angle errors between desired and actual joint trajectories for the numerical projection approach, with Gaussian triangularization fixed time stepping of 0.001-s, and $\sigma = 10$: (a)error between desired and actual trajectories after feedback correction; (b)correction in fed driving torques feedback.

Finally, to assess the accuracy of the model, we obtained the results for $\sigma = 10$ for a fixed time stepping of 0.001-s using feedback control. Figure 4.11a shows the error between the desired trajectory and the trajectory obtained from simulation with feedback control. Figure 4.11b shows the amount of percentage of torque correction required to obtain the desired trajectory. These values are within 15% of the input torque. Based on the above results we can conclude that:

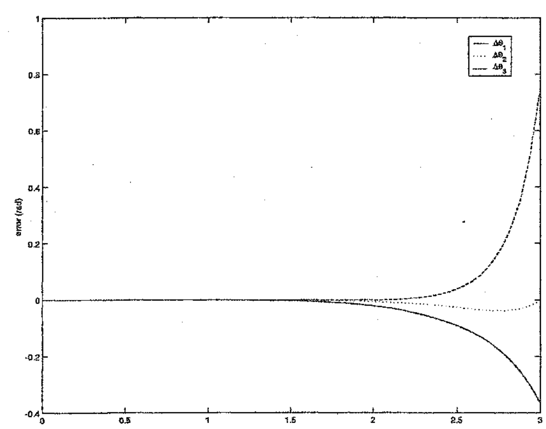
(i) The number of EOM evaluations is much lower than that in the virtual-spring approach. However, as the the convergence factor σ increases, the number of evaluation also increases.

(ii) The accuracy of this method, for our experiment, is again around 85%

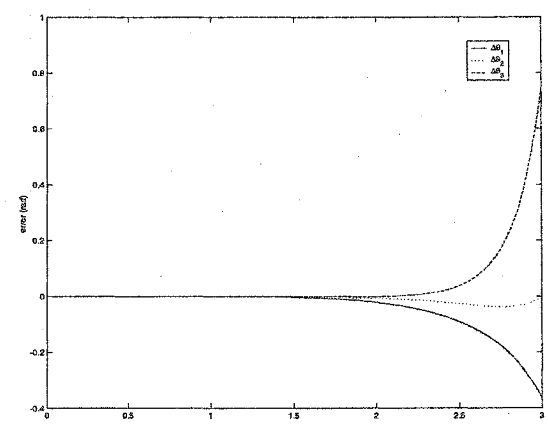
(iii) A change of minimal set of generalized coordinates being integrated may be required during simulation, which is not easy to (a)realize and (b)execute. Resetting of integrators is also required.

Finally, we also examined the use of *QR-decomposition*, as discussed in (García de Jalón and Bayo, 1994) and in the Appendix, to obtain the orthogonal complement for use in the projection; the corresponding results are similar to the one obtained using the method based on Gaussian triangularization and are not included here for brevity.

4.3. Recursive Decoupled Natural Orthogonal Complement



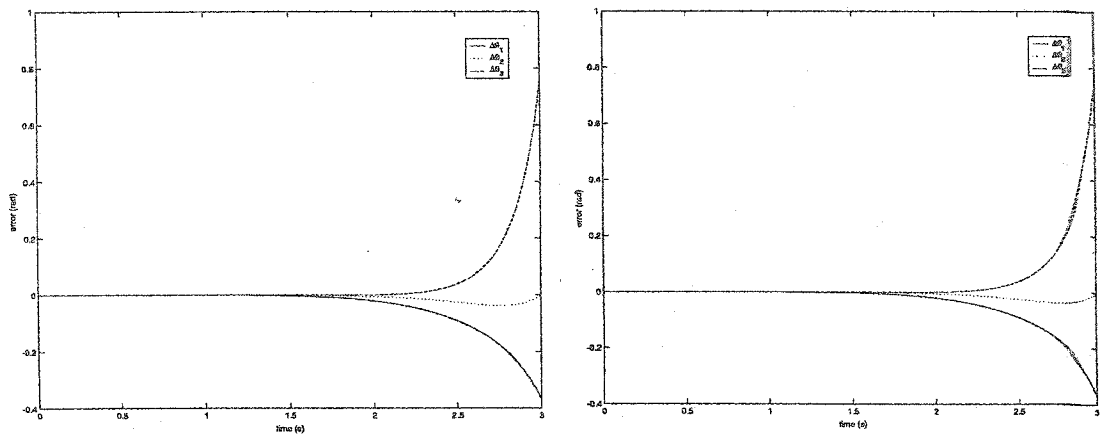
(a)



(b)

Figure 4.12: Joint-angle errors between desired and actual joint trajectories for the decoupled natural orthogonal complement with adaptive time stepping for: (a)relative tolerance of 1e-3; and (b) relative tolerance of 1e-6.

The results of the DeNOC model now follow: Figure 4.12 shows the resulting error between desired and simulated motion trajectories for adaptive time stepping with relative tolerances of 1e-3 and 1e-6, respectively. A relatively small number of time-steps was required, namely, 301, when simulated with a relative tolerance of 1e-3, and 319 when simulated with a tolerance of 1e-6. Figure 4.13a shows the errors



(a) (b)

Figure 4.13: Simulation results for the DeNOC with fixed time-step. Joint-angle errors between desired and actual joint trajectories for time steps of: (a) 0.01-s; and (b) 0.001-s.

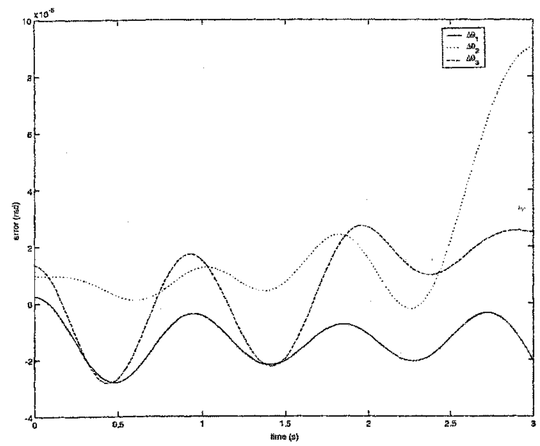
between the desired trajectory and the trajectory obtained from simulation, for a fixed time step of 0.01-s. Figure 4.13b shows the same for fixed time stepping of 0.001-s. The deviation is due to the corresponding zero eigenvalue instability, which is overcome using feedback control.

Figure 4.14a shows the error between the desired trajectory and the trajectory obtained from simulation with feedback control. Figure 4.14b shows the percentage of torque correction required to obtain the desired trajectory where we would like to highlight the extremely small correction required.

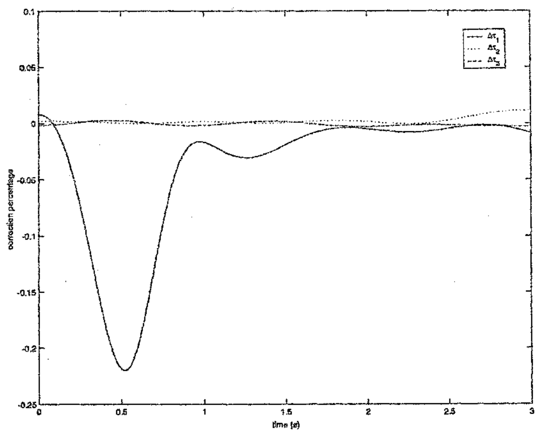
Based on the above results we can conclude that:

(i) The number of iterations required for adaptive time stepping for relative tolerances of 1e-3 and 1e-6 is low.

(ii) The accuracy of this method, for our experiment, is quite high, of around 99%.



(a)



(b)

Figure 4.14: Simulation results for the DeNOC with feedback control for a fixed time step of 0.001-s: (a) joint angle errors between the desired and actual joint trajectories; and (b) driving-torque correction time history.

CHAPTER 5

Conclusions and Recommendations for Future Work

5.1. Conclusions

Three alternative methods: compliance-based; numerical-projection-based; and the decoupled natural orthogonal complement, which showed promise from the viewpoint of both modularity and distributivity, were compared in this study. Fully distributed computational implementations, i.e., those which do not require any information exchange during each integration step, were derived for the compliance-based and the numerical-projection-based methods. This was accomplished by integrating parts of the minimal-acceleration-set separately. The distributed model showed no variations compared to non-distributed models.

A new minimal order recursive method, DeNOC, recently proposed by Saha and Schiehlen (2001), was implemented with slight variations made in light of modularity and distribution. This method may not be "fully" distributed, but its parallel recursive nature makes it feasible for distributed dynamics. A summary of the comparison is displayed in Table 5.1

The formulation stiffness of the penalty-based approaches coupled with significant inaccuracies due to the approximation of the actual model were major detracting factors for its poor performance. However, this approach rated very well from the

	Compliance-Based (Wang *et al.*, 2000)	Numerical-Projection-Based (Yun and Sarkar, 1998)	Numerical-Projection-Based (Gaussian)	Numerical-Projection-Based (QR)	Decoupled Orthogonal Complement
Modularity	High	High	High	High	Med
Distribution	High	High	High	High	Med
Preparation Cost	Low	Med	Med	Med	High
Accuracy	Med	Med	Med	Med	High
EOM Computation Cost	Low	Med	Med	Med	High
Stiffness	High	Low	Low	Low	Low
Efficiency	Low	Med	Med	Med	Med
Comments		Suggested numerical approach applicable only to 1-dof systems	In-simulation integrator resets may be required	In-simulation integrator resets may be required	Full distribution is not possible

Table 5.1: Comparison table of three simulation methods

viewpoint of modularity, i.e., new subsystems may be added or deleted to the existing ones quite easily; exploitation of spatial parallelism was also easy. Additionally, while the equations of motion are relatively inexpensive to evaluate, we note that the traditional method of counting floating point operations (flops) for an EOM evaluation should *not* be used as the sole measure of efficient dynamic models. Formulation stiffness in this case, makes the problem more expensive than it might seem at a first glance. Full distribution, i.e., no data transfer during state updates, is possible by passing states to every node. The virtual spring approach is not minimal.

The loop-closure orthogonal complement-based numerical projection approaches can be fully distributed as we showed. However, many calculations must either be repeated at each node or calculated by a single node and broadcast to other nodes within the time-step. However, the latter approach would need information-transfer between time-steps and hence the former approach, with some redundant computations, is adopted. The systems of interest are modular, where incorporation of new models is fairly easy, although this is not as straightforward as in the penalty-based approach. Although finding an orthogonal complement to the constraint matrix is easy numerically, the problem of maintaining the directions of this orthogonal basis introduces slight difficulties.

Simulation models based on the decoupled natural orthogonal complement were found most accurate, but the distribution of this model is very difficult. Although full distribution may be applied, the resulting model does not show appreciable increase in simulation speed for this type of distribution. The best result should be obtained with partial distribution, i.e., minimal data-transfer during state updates as developed in Chapter 3.

5.2. Future Work

As an extension to the work, further research is recommended as follows:

(i) The parallel computation system based on RT-Lab version 5.2.3 passes data as states and no communication between integration time-steps is possible.

On the other hand, the DeNOC-based model developed in Chapter 3 is a distributed model, but needs communication within the same integration time-step. For this reason, simulations for this approach were performed on a single PC, thus avoiding synchronization issues. Further investigation of this issue and simulation of the DeNOC-based model on an actual parallel computation system is thus recommended.

(ii) To avoid displacement analysis, an extended set of states was used for the DeNOC-based model, which raises stability issues. Incorporation of a stabilization scheme in the DeNOC-based model could be attractive for future work.

(iii) A study of the modelling methods discussed on systems incorporating both holonomic and nonholonomic constraints should be conducted.

(iv) A study on applicability of the DeNOC approach to symbolic dynamics formulations is also recommended.

BIBLIOGRAPHY

Abou-Samah, M., 2001, *A Kinematically Compatible Framework for Collaboration of Multiple Non-holonomic Wheeled Mobile Robots*, M.Eng. thesis, McGill University, Montreal.

Anderson, K. and Duan, S., 2000, "Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems," *AIAA Journal on Guidance, Control and Dynamics* 23, no. 2, pp. 355–364.

Angeles, J., 2002, *Fundamentals of Robotic Mechanical Systems*, 2nd ed., Springer-Verlag, New York.

Armstrong, W., 1979, "Recursive solution to the equations of motions of an n-link manipulator," *Proc. 5th World Congress on Theory of Machines and Mechanisms*, Montreal, pp. 1343–1346.

Ascher, U. and Petzold, L., 1998, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia.

Bae, D. and Han, J., 1999, "A generalized recursive formulation for constrained mechanical system dynamics," *Mech. Struct. Mach.* 27, no. 3, pp. 293–315.

Bae, D. and Haug, E., 1987, "A recursive formulation for constrained mechanical system dynamics: part 2. closed loop systems," *Mech. Struct. Mach.* 15, no. 4, pp. 481–506.

Balafoutis, C., Patel, R., and Cloutier, B., 1988, "Efficient modelling and computation of manipulator dynamics using orthogonal cartesian tensors," *IEEE Journal of Robotics and Automation* 4, pp. 665–676.

Birta, L. and Abou-Rabia, A., 1987, "Parallel block predictor-corrector methods of ODE's," *IEE Trans. Computers* C-36, pp. 299–311.

Brandl, H., Johanni, R., and Otter, M., 1986, "A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix," *Proc. IFAC/IFIP/IMACS International Symposium on Theory of Robots*, Vienna.

Chaudhry, V. and Aggarwal, J., 1990, *Parallel Algorithms for Machine Intelligence and Vision*, Chap. Parallelism in Computer Vision: A Review, Springer-Verlag, New York.

Featherstone, R., 1983, "The calculation of robot dynamics using articulated-body inertias," *Int. Journal of Robotics Research* 2, no. 1, pp. 13–30.

Featherstone, R., 1987, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Boston/Dordrecht/Lancaster.

Featherstone, R., 1999, "A divide-and-conquer articulated-body algorithm for parallel $O(\log(n))$ calculation of rigid-body dynamics. part 2: trees, loops and accuracy," *Int. Journal of Robotics Research* 18, no. 9, pp. 876–892.

Featherstone, R. and Fijani, A., 1999, "A technique for analyzing constrained rigid-body systems, and its application of the constraint force algorithm," *IEEE Trans. Robotics and Automation* 15, no. 6, pp. 1140–1144.

Fijani, A. and Bejczy, A., 1992, *Parallel Computation Systems for Robotics: Algorithms and Architectures*, World Scientific, River Edge, NJ.

Fijani, A., Sharf, I., and D'Eleuterio, G. M., 1995, "Parallel O(logN) algorithms for computation of manipulator forward dynamics," *IEEE Trans. Robotics and Automation* 11, no. 3, pp. 389–400.

Fijany, A. and Bejczy, A. K., 1991, "Parallel algorithms and architecture for computation of manipulator forward dynamics," *Proc. IEEE International Conference on Robotics and Automation*, Sacramento, California, pp. 1156–1162.

García de Jalón, J. and Bayo, E., 1994, *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*, Springer-Verlag, New York.

Geike, T. and McPhee, J., 2002, "On the automatic generation of inverse dynamic solutions for parallel manipulators," *Proc. Workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators*, Quebec City, pp. 348–358.

Goldenberg, A. and He, X., 1989, "An algorithm for efficient computation of dynamics of robotic manipulators," *Proc. Fourth International Conference on Advanced Robotics*, Columbus, OH, pp. 175–188.

Haug, E., 1989, *Computer Aided Kinematics and Dynamics of Mechanical Systems*, Allyn and Bacon, Boston.

Henrich, D. and Höniger, T., 1997, "Parallel processing approaches in robotics," *Proc. IEEE International Symposium on Industrial Electronics*, Guimaraes, Portugal, pp. 702–707.

Kecskemethy, A., Krupp, T., and Hiller, M., 1997, "Symbolic processing of multi-loop mechanism dynamics using closed form kinematic solutions," *Multibody System Dynamics* 1, no. 1, pp. 23–45.

Kim, S. and Vanderploeg, M., 1986, "QR Decomposition for state space representation of constrained mechanical dynamic systems," *ASME Journal of Mechanisms, Transmissions and Automation in Design* 108, pp. 183–188.

Kogge, P., 1974, "Parallel solution of recurrence problems," *IBM Journal Res. Develop.* 18, pp. 138–148.

Kogge, P. and Stone, H., 1973, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Computers* C-22, pp. 789–793.

Lee, C. and Chang, P., 1986, "Efficient parallel algorithms for inverse dynamics computation," *IEEE Trans. Systems, Man. and Cybernetics* SMC-16, no. 4, pp. 532–542.

Lee, C. and Chang, P., 1988, "Efficient parallel algorithms for robot forward dynamics computation," *IEEE Trans. Systems, Man. and Cybernetics* 18, no. 2, pp. 238–251.

Luh, J., Walker, M., and Paul, R., 1980, "On-line computational schemes for mechanical manipulators," *ASME Trans. J. Dynamics Systems, Measurement and Control*

102, no. 2, pp. 69–76.

Ma, O. and Angeles, J., 1989, "Direct kinematics and dynamics of a planar 3-dof parallel manipulator," *Advances in Design Automation*, vol. 3, Montreal, Quebec, pp. 313–320.

McMillan, S. and Orin, D., 1995, "Efficient computation of articulated-body inertias using successive axial screws," *IEEE Trans. Robotics and Automation* 11, pp. 606–611.

McMillan, S., Sadayappan, P., and Orin, D. E., 1994, "Parallel dynamic simulation of multiple manipulator systems: temporal versus spatial methods," *IEEE Trans. Systems, Man and Cybernetics* 24, no. 7, pp. 982–990.

Murray, R., Li, Z., and Sastry, S., 1994, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, FL.

Orin, D., McGhee, R., Vukobratovic, M., *et al.*, 1979, "Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods," *Mathematical Biosciences* 43, pp. 107–130.

Orin, D. and Walker, M., 1982, "Efficient dynamic computer simulation of robotic mechanisms," *ASME Trans. J. dynamics Systems, Measurement and Control* 104, pp. 205–211.

Roosta, S. H., 2000, *Parallel Processing and Parallel Algorithms: Theory and Computation*, Springer, New York.

Sachdev, C., 2002, "Cooperative robots share the load," Tech. rep., TRN News, URL `http://www.trnmag.com/Stories/2002/021302/Cooperative_robots_share_the_load_021302.html`.

Sadayappan, P., Ling, Y., Olson, L., *et al.*, 1989, "A Re-structurable VLSI robotics vector processor architecture for real-time control," *IEEE Trans. Robotics and Automation* 5, pp. 583–599.

Saha, S. K., 1999, "Dynamics of serial multibody systems using the decoupled natural orthogonal complement matrices," *ASME Journal of Applied Mechanics* 66, pp. 986–996.

Saha, S. K. and Schiehlen, W. O., 2001, "Recursive kinematics and dynamics for parallel structured closed-loop multibody systems," *Mechanics of Structures and Machines, An International Journal* 29, no. 2, pp. 143–175.

Schiehlen, W., 1990a, "Multibody systems and robot dynamics." *Proc. 8thCISM-IFToMM Symposium on Theory and Practice of Robot Manipulators* (A. Morecki, G. Bianchi, and K. Jaworek, eds.), Warsaw, Poland, pp. 14–21.

Schiehlen, W., 1990b, *Multibody Systems Handbook*, Springer-Verlag, Berlin.

Serna, M., Avilés, R., and García de Jalón, J., 1982, "Dynamic analysis of plane mechanisms with lower-pairs in basic coordinates," *Mechanism and Machine Theory* 17, pp. 397–403.

Shabana, A. A., 2001, *Computational dynamics*, Wiley, New York.

Shampine, L. F. and Reichelt, M. W., 1997, "The MATLAB ODE suite," *SIAM Journal on Scientific Computing* 18, pp. 1–22.

Stejskal, V. and Valasek, M., 1996, *Kinematics and Dynamics of Machinery*, Marcel Dekker, New York.

Stepanenko, Y. and Vukobratovic, M., 1976, "Dynamics of articulated open-chain active mechanism," *Math. Biosciences* 28, pp. 137–170.

Vereshchagin, A., 1974, "Computer simulation of the dynamics of complicated mechanisms of robot manipulators," *Engineering Cybernetics* 6, pp. 65–70.

Walker, M. and Orin, D., 1982, "Efficient dynamic computer simulation of robotic mechanisms," *ASME Journal of Dynamic Systems, Measurement and Control* 104, pp. 205–211.

Wang, J., Gosselin, C., and Cheng, L., 2000, "Dynamic modelling and simulation of parallel mechanisms using virtual spring approach," *Proc. 2000 ASME Design Engineering Technical Conferences*, Baltimore, Maryland, pp. 1–10.

Yiu, Y., Cheng, H., Xiong, Z., *et al.*, 2001, "On the dynamics of parallel manipulator," *Proc. IEEE international Conference on Robotics and Automation*, Seoul, Korea, pp. 3766–3771.

Yun, X. and Sarkar, N., 1998, "A unified formulation of robotic systems with holonomic and non-holonomic constraints," *IEEE Trans. Robotics and Automation* 14, no. 4, pp. 640–650.

Zoyama, A., 1993, *Modelling and Simulation of Robot Manipulators: A Parallel Processing Approach*, World Scientific, River Edge, NJ.

APPENDIX A

Alternative Methods to Obtain an Orthogonal Complement

Here we discuss some of the typical methods to obtain the orthogonal complement $\mathbf{T}$ introduced in Chapter 2.

A.1. Singular-Value Decomposition

Using the SVD, we can factor the constraint Jacobian in the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{A.1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $m \times m$ and $n \times n$ orthogonal matrices, while $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix containing the singular-values of $\mathbf{A}$. For a full-rank matrix $\mathbf{A}$ this could be further elaborated as

$$\mathbf{A} = \mathbf{U}\begin{bmatrix}\bar{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{bmatrix} \tag{A.2}$$

where $\mathbf{V}_2$ is a $n \times (n-m)$ orthogonal matrix and $\bar{\mathbf{\Sigma}}$ is a $m \times m$ diagonal matrix containing the non-zero singular values of $\mathbf{A}$. The columns of matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ are orthogonal vectors, and

$$\mathbf{V}_1^T\mathbf{V}_2 = \mathbf{O}, \text{ or } \mathbf{V}_2^T\mathbf{V}_1 = \mathbf{O} \tag{A.3}$$

From eq.(A.2) it follows that

$$\mathbf{A} = \mathbf{U}\bar{\boldsymbol{\Sigma}}\mathbf{V}_1^T \tag{A.4}$$

Post-multiplying this equation by $\mathbf{V}_2$ we obtain

$$\mathbf{A}\mathbf{V}_2 = \mathbf{O} \tag{A.5}$$

which implies that the orthogonal columns of the matrix $\mathbf{V}_2$ span the nullspace of the matrix $\mathbf{A}$ and is thus an orthogonal complement of $\mathbf{A}$.

A.2. QR factorization

Using *QR factorization*, we can decompose the constraint jacobian in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_1^T \\ \mathbf{Q}_2^T \end{bmatrix} \tag{A.6}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are $n \times m$ and $n \times (n-m)$ orthogonal matrices, and $\mathbf{R}$ is a $m \times m$ lower triangular matrix. Again

$$\mathbf{A} = \mathbf{R}\mathbf{Q}_1^T \tag{A.7}$$

and thus

$$\mathbf{A}\mathbf{Q}_2 = \mathbf{R}\mathbf{Q}_1^T\mathbf{Q}_2 = \mathbf{O} \tag{A.8}$$

or

$$\mathbf{A}\mathbf{Q}_2 = \mathbf{O} \tag{A.9}$$

which implies that the orthogonal columns of the matrix $\mathbf{Q}_2$ span the nullspace of the matrix $\mathbf{A}$ and is thus an orthogonal complement of $\mathbf{A}$.

Both SVD and QR decomposition are expensive when it comes to computer costs. Although SVD is without doubt very stable around singularities, it is essentially an iterative procedure rendering it useless for real-time simulations where a less accurate solution *within the given time* is better than no solution at all.

The second important point to note here is that in projection methods, the set of independent accelerations is integrated. Hence, the basis of the nullspace in which

these are defined should not change directions during the simulation process; else the integrator should be reset. Orthogonal complements obtained from SVD and QR decomposition are not unique; it is, therefore numerically difficult to preserve the directional continuity of the bases represented by their columns. One way to counter this problem is to obtain the orthogonal matrices at the initial configuration and use the velocity constraint relationships to iteratively update these matrices in order to preserve the directional continuity of the nullspace of the constraint Jacobian (Kim and Vanderploeg, 1986). This idea is summarized below:
Consider a system with rheonomic constraints i.e.

$$\begin{aligned} &\mathrm{c}(\mathrm{q}, t) = 0 \\ &\frac{\partial \mathrm{c}}{\partial \mathrm{q}}\dot{\mathrm{q}} = -\dot{\mathrm{c}} \end{aligned} \tag{A.10}$$

which can be written as

$$\mathrm{A}\dot{\mathrm{q}} = \mathrm{b} \tag{A.11}$$

where $\mathrm{A} = \partial \mathrm{c}/\partial \mathrm{q}$ and $\mathrm{b} \equiv -\dot{\mathrm{c}}$. Let N be the $n \times (n-m)$ orthogonal complement found by either SVD or QR factorization, obtained at the initial configuration and $\dot{\mathrm{z}}$ be the $(n-m)$-dimensional independent velocity vector. Then,

$$\dot{\mathrm{z}} = \mathrm{N}^T \dot{\mathrm{q}} \tag{A.12}$$

Together with eq.(A.11), we obtain a system of n equations in n unknowns, namely,

$$\begin{bmatrix} \mathrm{A}(\mathrm{q}) \\ \mathrm{N}^T \end{bmatrix} \dot{\mathrm{q}} = \begin{bmatrix} \mathrm{b} \\ \dot{\mathrm{z}} \end{bmatrix} \tag{A.13}$$

or

$$\dot{\mathrm{q}} = \begin{bmatrix} \mathrm{A}(\mathrm{q}) \\ \mathrm{N}^T \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{b} \\ \dot{\mathrm{z}} \end{bmatrix} = \begin{bmatrix} \mathrm{R} & \mathrm{T} \end{bmatrix} \begin{bmatrix} \mathrm{b} \\ \dot{\mathrm{z}} \end{bmatrix} = \mathrm{Rb} + \mathrm{T}\dot{\mathrm{z}} \tag{A.14}$$

where T is the sought orthogonal complement. As the configuration changes with time the constraint jacobian A changes. As long as $\begin{bmatrix} \mathrm{A}(\mathrm{q}) \\ \mathrm{N}^T \end{bmatrix}$ is full rank i.e. rows of $\mathbf{N}^T$ remain linearly independent to $\mathbf{A}(\mathbf{q})$ it can be inverted and the last $n-m$ columns selected to obtain $\mathbf{T}$. When the configuration of the system is such that $\begin{bmatrix} \mathbf{A}(\mathbf{q}) \\ \mathbf{N}^T \end{bmatrix}$ is close to singularity, a new set of orthogonal basis must be selected and the procedure continued. Note here, however, that the integrators must be reset at this change over point.

A.3. Gaussian Triangularization

The last method based on *Gaussian triangularization* is explained next. In this method, described by Serna *et al.* (1982), the constraint jacobian $\mathbf{A}(\mathbf{q})$ is first triangularized by means of Gaussian elimination method with total pivoting. For a full rank $\mathbf{A}$ matrix, once the elimination is complete, the matrix may be partitioned in to two parts, i.e. a $m \times m$ upper diagonal matrix, $\mathbf{A}_d$, and $m \times (n-m)$ full matrix, $\mathbf{A}_i$, resulting eq.(A.11) to be

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{A}_d & \mathbf{A}_i \end{bmatrix} \dot{\mathbf{q}} \tag{A.15}$$

A boolean matrix $\mathbf{B}$ is then formed by a set of ones and zeros that extracts $n-m$ components of $\mathbf{q}$ as independent coordinates $\mathbf{z}$

$$\dot{\mathbf{z}} = \mathbf{B}\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{O} & \mathbf{1} \end{bmatrix} \dot{\mathbf{q}} \tag{A.16}$$

This is then augmented with eq.(A.15) giving

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{A}_d & \mathbf{A}_i \\ \mathbf{O} & \mathbf{1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \dot{\mathbf{z}} \end{bmatrix} = \mathbf{R}\mathbf{b} + \mathbf{T}\dot{\mathbf{z}} \tag{A.17}$$

This method is similar to the coordinate partitioning method since it can be seen that the independent velocities $\dot{\mathbf{z}}$ are chosen as a subset or extraction of the dependent velocities. Note that the matrix $\mathbf{A}_d$ should be invertible in order to express the dependent velocities in terms of the independent ones. All the pivots must be sufficiently different from zero. This guarantees that the chosen rows of $\mathbb{B}$ are independent from those of $\mathbb{A}$. Calculating matrix $\mathbb{T}$ is fairly simple. Matrix $\mathbb{A}_d$ is already upper triangular. Hence, matrix

$$\begin{bmatrix} \mathbb{A}_d & \mathbb{A}_i \\ \mathbb{O} & \mathbb{1} \end{bmatrix}$$

remains upper triangular. Matrix $\mathbb{T}$ may thus be found by performing $n-m$ successive forward and backward substitutions with the $n-m$ last vectors of $n \times n$ identity matrix $\mathbb{1}$ as the right hand side.