Real-Time Compensation of Static Deflections in Robotic Manipulators

by

Joseph M. Calkins

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Approved:

C. F. Reinholtz, Co-Chairman

R. J. Salerno, Co-Chairman

A. L. Abbott

December, 1994
Blacksburg, Virginia

Real-Time Compensation of Static Deflections in Robotic Manipulators

by

Joseph M. Calkins

Committee Chairmen: Charles F. Reinholtz and Robert J. Salerno

Virginia Polytechnic Institute and State University

Department of Mechanical Engineering

December 1994

Abstract

The focus of this work is the real-time prediction and compensation of static deflections in robotic manipulator arms. A general manipulator deflection model is developed based on static beam theory and robot kinematics. An optimization technique is proposed to determine the stiffness of the manipulator components using end-effector deflection measurements. Strategies for incorporating this modeling approach into a manipulator controller are also presented along with the results of a successful application of this research.

This work is an extension of previous manipulator deflection research. Multiple pairs of torsional stiffness elements and beam elements are used to model complex link and joint geometries whereas previous models only used a single beam per manipulator link. In addition, the modeling algorithms and stiffness characterization methods are general and may be applied directly to any serial manipulator. Also, the optimization techniques used

to characterize a manipulator's stiffness provide a more accurate and flexible approach than previous analytical methods.

The deflection model was successfully tested using a nuclear steam generator service manipulator. Since this manipulator is considerably more flexible than common industrial robots, it serves as a near worst-case test for deflection modeling. The end-effector was found to deflect as much as 1.5 inches due to the weight of the links and joints. The deflection model was able to compensate for 94% of the end-effector deflection, allowing the manipulator to perform tasks requiring a positioning accuracy of 0.09 inches.

The algorithms for flexible forward and inverse kinematics as well as trajectory generation were incorporated directly into the manipulator controller code. These modules were capable of running in real-time with little computational expense.

Acknowledgments

Working as a graduate student for Prof. Charles Reinholtz has been an enjoyable and rewarding experience. Prof. Reinholtz's optimism, enthusiasm, and generosity has provided an excellent research environment. I am deeply grateful for the insights he has provided as both a friend and a mentor.

I am also grateful to Prof. Robert Salerno for his excellent advice and direction as a Co-Chair of my committee. I would also like to thank Prof. Lynn Abbott for serving on my committee and providing his unique perspective and insights.

Special thanks are also due to Gary Wallace and Joe Schlepko. These students provided a wealth of data and analysis essential to the success of this work. I must also acknowledge the contributions of Jon Jarvis who first encouraged me to pursue graduate studies. Jon's initial work with the deflection of the Cobra manipulator system laid the foundations for this work.

I am also grateful to my parents, Joseph and Nancy, for their encouragement and support throughout my life. My father's interest in engineering, science, and computing sparked my interest in building, experimenting, and hacking early in my life.

I must also express appreciation to my wife, Kara, who has endured my attachment to the computer over the last several months.

The financial and staff support from the Manipulator Development group at B&W Nuclear Technologies is gratefully acknowledged. Specifically, I would like to thank Bill Glass, Joe Steinbrunner, and David Oliver for the time and energy they invested in the quest for a more accurate Cobra manipulator.

Contents

List of Figures

List of Tables

1. Introduction

The human arm is capable of lifting ten times its own weight. Common industrial manipulators, however, can only lift approximately one-twentieth of their weight (Lee and Wang 1987). This comparison illustrates the relatively low strength-to-weight ratio common in today's robotic manipulators. These robots are designed to be inherently stiff in order to eliminate positioning errors caused by deflection.

In order for robotic systems to be used in hazardous environments and other service applications, they must be lightweight and portable. In addition, they must be strong, capable of lifting heavy repair and inspection tools. The heavy industrial manipulators found on factory-floors are not well suited to these requirements.

The design of light-weight, high-strength robotic manipulators that maintain a high-degree of positional accuracy is a difficult task. Often, the stiffness of the manipulator is sacrificed in order to reduce weight, enabling humans to carry the individual components to the workspace. Methods for compensating for the flexibility of these systems are needed in order to achieve high levels of positioning accuracy.

The goal of this work is to design a general manipulator deflection compensation algorithm capable of running as an integral part of a robot's control system. To achieve this, a method must also be developed for determining the stiffnesses of the manipulator components. This chapter will present the primary motivation behind this research, and provide a brief synopsis of previous work.

1.1 Motivation

This work is in response to a request by B&W Nuclear Technologies to increase the positioning accuracy of their Cobra manipulator series. These manipulators are used to position a variety of inspection and repair tools beneath the tubesheet of a nuclear steam generator. The Cobra robots are designed to be carried by humans through the containment area of a nuclear power-plant. For this reason, the Cobra robots must be lightweight and portable. This design requirement has the side-effect of lower stiffness characteristics that result in unacceptable positioning errors. A more detailed description of the Cobra manipulator and its operating environment will be given in Chapter 5.

This flexibility problem also exists in industrial manipulators. While the magnitudes of the deflection errors are relatively small, they are significant in some applications. For this reason, the deflection model was developed for the general case of a flexible serial manipulator.

1.2 Previous Work

The study of flexible manipulator systems may be categorized into elasto-dynamic and elasto-static analysis. The dynamics associated with flexible manipulators has been studied in great detail, usually with the emphasis on minimizing structural vibrations. In addition, many researchers have investigated compliance issues pertaining to insertion tasks. The emphasis of this literature review, however, will be on the modeling and compensation of robot positioning errors due to static deflections.

In 1983, Derby used computer graphics to represent the static deflections of manipulators. The model assumed small bending deflections that did not change the components of the applied forces. The method used to determine deflections was not suitable for on-line

calculations, and was intended as a first-order approximation to assist in the robot design process.

Mooring (1983) determined the effect of joint axis misalignment on robot positioning. He found that a one degree misalignment in the 2nd joint of the PUMA 600 manipulator resulted in a maximum tool positioning error exceeding one inch. Though Mooring did not discuss the deflection of the manipulator links or joints, the analysis may be applied to flexible manipulators. Though the initial machined locations of the joint axes may line up satisfactorily, deflection may induce the slight misalignments studied by Mooring.

In 1987, Naganathan and Soni proposed a finite-element based, nonlinear model for flexible manipulators. Timoshenko beam theory was used to model the elements composing the links of a manipulator. The focus of the work was on the dynamics of the flexible manipulator system applied to a two-link planar manipulator, though the theory applies to spatial manipulators as well.

In 1988, Bodur and Derby developed a deflection model incorporating link bending and joint clearances. This model was applied to the off-line programming of a Cincinnati-Milacron T^3 manipulator. Though links were modeled as cantilever beams, the rotational deflection at the end of the beam was ignored. Clearances in the joints were used to model deflections, but the rotational stiffnesses of the joints themselves were ignored.

Lee and Wang (1988) used finite element techniques to analyze a single-link manipulator without attempting to linearize the system. The focus of their work was control of the dynamic properties of flexible manipulators.

In 1990, Lin, Chiang, and Cui modeled manipulator links as cantilever beams with both rotational and positional deflection. The rotational beam deflection component neglected

by Bodur and Derby (1988) was found to dominate the overall end-effector deflection. To verify their theoretical model, Lin, Chiang, and Cui constructed a two-link planar manipulator with locking joints.

In 1991, Chang and Hamilton used an equivalent rigid link system to model flexible link manipulators. The rigid-body dynamics were separated from the structural dynamics so that existing methods could be applied to these problems. The derivation of the Jacobian and the inverse solution methodology were also outlined.

Xi and Fenton (1991) investigated the motion planning of flexible manipulators. They used the algebra of rotations in conjunction with three Jacobians to model the static deflection in a Puma-type manipulator. The wrist structure was assumed to be rigid, and the stiffness parameters were determined analytically based on constant cross-section assumptions. The problems associated with Jacobian singularity were discussed, and several methods were suggested to resolve singularity problems.

In 1992, Xi and Fenton proposed numerical methods for solving the inverse kinematics of flexible manipulators using the algebra of rotations. They expanded this work in 1994 to include the dynamic effects of flexible manipulators. The inverse statics problem was solved by incorporating a Jacobian representation of the flexible components of the manipulator into the overall Jacobian. This composite Jacobian was then inverted and used in a iterative scheme to converge to the correct set of joint values satisfying the goal specification.

Lin et al. (1992) proposed the theoretical basis for an iterative forward transformation algorithm to determine end-effector pose under the effects of deflection. An identification scheme was outlined to determine the kinematic and structural parameters

of a flexible manipulator using end-effector pose measurements. Their work focused on very flexible space manipulators.

Yang and Sadler (1992) developed a joint-beam finite element to be used in modeling completely flexible manipulators. This element consisted of two Euler beams connected by a torsional member with a stiffness proportional to applied torque. Though the authors applied this model to dynamic behavior, the approach used to model the members of the manipulator may be applied to the static case as well.

The works of Patterson and Lipkin (1990a, 1990b, 1992a, 1992b) treat the entire manipulator system as an unconstrained, elastically suspended rigid body in order to improve robot insertion and other compliance-plagued tasks. An alternative to the remote center of compliance end-effector was proposed. An approach using centers of stiffness, compliance, and elasticity was developed by Cibiak and Lipkin (1994a, 1994b). A comparison was made between a remote center of compliance device and a robot hand using the problem of rivet insertion as an example.

In 1994, Sun, Griffis, and Duffy proposed a methodology for comparing stiffness mappings of various robotic manipulators. Their method uses geometric invariants in the mapping comparison to eliminate coordinate-system-dependent elements of a stiffness matrix.

1.3 Outline of Contents

Chapter 2 will describe the general modeling approach used to determine the deflection of a manipulator's links and joints. Several modeling elements will be presented along with the process used to apply the deflection of these elements to the kinematics of a robotic manipulator. Since the model that will be presented in Chapter 2 will require stiffness

parameter values for the manipulator components, Chapter 3 will address the methods that may be used to determine these unknown stiffness parameters. Chapter 4 will discuss the implementation of the deflection modeling algorithms into a robot control system. The successful application of this theory to a nuclear service manipulator will be presented in Chapter 5. Finally, Chapter 6 will present recommendations for future research in this area as well as concluding remarks.

2. Theoretical Deflection Modeling Approach

The traditional kinematic model of a serial manipulator uses coordinate frames attached to the individual links to determine the relationship between the base of the manipulator and the end-effector. Each pair of adjacent links is connected by a joint with either a rotational or a translational degree of freedom. The relationship between each of the link coordinate frames is defined by four parameters, where one parameter is the variable joint value. This four-parameter link coordinate representation is called the Denavit-Hartenberg notation (Denavit and Hartenberg 1955, Craig 1989).

In the Denavit-Hartenberg link description, it is assumed that the links and joints of the manipulator are rigid. The modeling approach described in this chapter will model the effects of deflection by translating and rotating the individual link coordinate frames according to the calculated deflection. This is done by breaking the manipulator down into a series of flexible sub-components that may be analyzed individually.

The deflection of a manipulator is a function of the loads applied to the arm. These include both external and internal loads such as the weight of the manipulator's links and joints. The first step in calculating the effects of deflection on a manipulator is to characterize all the loads applied to the system. Once the loading configuration of the manipulator is known, the individual components are treated as free bodies. The deflections of the individual members are determined, and the arm is reassembled using the deflected components to yield the total positional and rotational deflection at the end-effector.

The manipulator modeling approach presented in this chapter is independent of the component deflection model. Each component could be modeled using a general function of the applied loads to obtain the rotation and translation at the end of the

component. The specific application of the deflection model presented in this work uses beam elements along each link of the manipulator and torsional stiffness elements at each revolute joint. These sub-components are used in pairs to model the deflection of the manipulator.

This chapter will use a three-link planar manipulator to present the deflection modeling scheme. First, the general modeling process will be presented. Next, the kinematics of the manipulator will be used to determine the loading conditions of each component. Finally, a specific component modeling approach will be presented.

2.1 Manipulator Modeling

Figure 2.1 illustrates the link coordinate frames of a planar manipulator both with and without the effects of deflection.

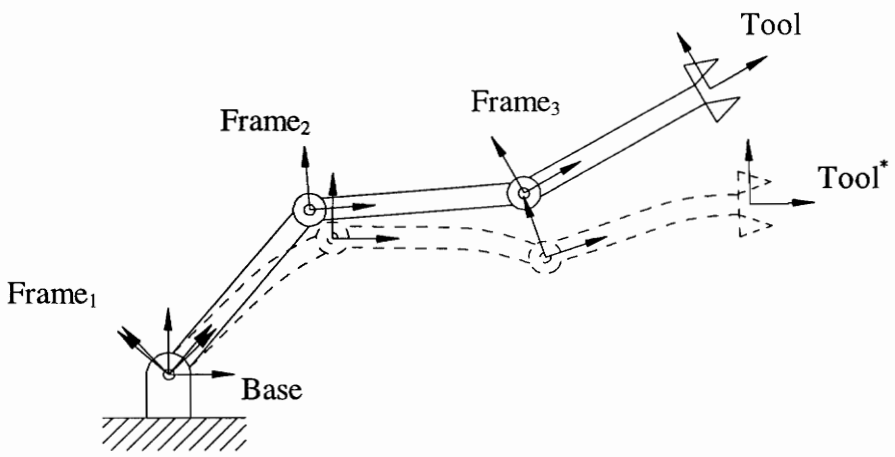

Figure 2.1 Deflection of a Planar Manipulator

Using the rigid kinematic representation of this manipulator, the forward kinematics is described as

$$ {}^{Base}_{Tool}T = {}^{Base}_{1}T \; {}^{1}_{2}T \; {}^{2}_{3}T \; {}^{3}_{Tool}T \tag{2.1} $$

where

${}^{i}_{i+1}T$ = transformation matrix from the *i+1* coordinate frame to the *i* frame.

The amount of deflection of each individual link coordinate frame is described by a transformation matrix, *D*. This matrix, referred to as the deflection matrix, represents the total rotational and positional deflection of the *i+1* link coordinate frame expressed in the link *i* coordinate frame. This deflection is a function of the loads on the components between link *i* and link *i+1* and the stiffnesses of those components. There are many ways to model the stiffness of a manipulator's components. The deflection matrix, however, is independent of the specific deflection model used for each component. In equation form this deflection matrix is

$$ D_{i,\,i+1} = f(\vec{F}_i, \vec{M}_i, \vec{w}_i) \tag{2.2} $$

where

i = index of the current deflected member

$D_{i,\,i+1}$ = positional and rotational deflection of link *i+1* relative to link *i*

$f()$ = general function of the applied loads

F_i = force applied to the end of link *i*

M_i = moment applied to the end of link *i*

w_i = distributed load due to the weight of link *i*

This deflection matrix is applied to each of the individual link transforms of the manipulator. The result is a set of link transforms representing the kinematics of the deflected manipulator. For any link, *i*, the deflected link transform is

$$ {}^{i}_{i+1}T^{*} = \left[D_{i,\,i+1} \right] {}^{i}_{i+1}T \tag{2.3} $$

The deflection of the second link of the planar manipulator shown in Fig. 2.1 is incorporated into the kinematics as

$$ {}^{2}_{3}T^{*} = \left[D_{2,\,3} \right] {}^{2}_{3}T \tag{2.4} $$

The deflected link transforms are multiplied together in the same way the rigid link transforms were multiplied,

$$ {}^{Base}_{Tool}T^{*} = {}^{Base}_{1}T^{*}\,{}^{1}_{2}T^{*}\,{}^{2}_{3}T^{*}\,{}^{3}_{Tool}T^{*} \tag{2.5} $$

The result is the forward kinematic representation of the manipulator including the effects of deflection.

2.2 Manipulator Component Loading Conditions

This section will present the method used to determine the forces and moments acting on each individual member of the manipulator. Since the location of the link coordinate frames changes when deflection occurs, it will be necessary to resolve the forces and moments into the deflected coordinate frames. An iterative scheme will be presented to address this issue.

2.2.1 Force and Moment Resolution Process

Each component of the manipulator is treated as a free body. To do this, the forces and moments applied to each component must be determined from the manipulator structural loading configuration using the following process. First, the location of each link coordinate frame is determined using forward kinematics. Second, the applied forces and moments are resolved for each member of the manipulator in the world coordinate system. The moment arm vectors in the world coordinate system are crossed with the force vectors to determine the additional moment vectors resulting from forces down the chain. Third, the force and moment vectors in the world coordinate system are transformed into each local link frame. This section will describe each step of this process.

The individual link coordinate frames are located relative to the world coordinate system using the manipulator forward kinematics. Given the Denavit-Hartenberg parameters for the manipulator, the individual link transforms are calculated using the general form of the manipulator link transform equation (Craig 1989). This will yield the transform from link coordinate frame $i+1$ to link coordinate frame i. Starting with the transform from the base of the manipulator to the world coordinate system origin, the individual link transforms are post multiplied to the previous transforms. This will yield the transform from any link coordinate frame to the world coordinate system. These transforms are used to resolve the applied loads to the individual members of the manipulator.

The second step, resolution of the link forces in the world coordinate frame, begins at the end of the serial chain. Each link of the chain is processed in descending order, until the forces and moments on the base are resolved. Figure 2.2 shows the notation used to denote the forces and moments relative to a link coordinate frame.

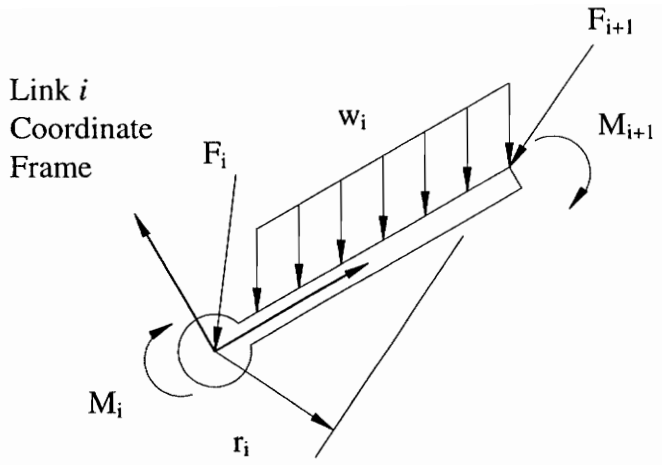

Figure 2.2 Notation Used For Loading in Link Coordinates

For any manipulator link, i, there is a total force and moment applied to the end of the member. This total force and moment is a result of the cumulative loading on the next link in the chain. These loads are called F_{i+1} and M_{i+1} because they are located at the $i+1$ coordinate frame. Given these end-loading conditions, it is necessary to determine the resulting force and moment at the origin of the link i coordinate frame. To do this, F_{i+1} is combined with the distributed loading along the link, and the joint weight at the origin of frame i. The combined force is

$$\vec{F}_i = \vec{F}_{i+1} + \vec{w}_i(L) + \vec{F}_{joint_i} \tag{2.6}$$

where

F_i = the force applied at the origin of link i

F_{i+1} = the force applied to the origin of link $i+1$

w_i = the distributed load applied along link i

L = the scalar distance between frame i and frame $i+1$

$F_{joint\,i}$ = the weight of the joint at the origin of frame i

M_i is determined by adding M_{i+1}, to the moment due to F_{i+1}, and the moment due to the distributed load w_i.

$$\vec{M}_i = \vec{M}_{i+1} + \left[(\vec{r}_i) \times (\vec{F}_{i+1})\right] + \left[\left(\frac{\vec{r}_i}{2}\right) \times (\vec{w}_i)(L)\right] \tag{2.7}$$

where

M_i = the moment applied about the origin of link i

M_{i+1} = the moment applied about the origin of link $i+1$

r_i = the vector from the origin of frame i to the origin of frame $i+1$

Once equations (2.6) and (2.7) have been applied to link i, the loading condition at the origin of coordinate frame i is known. The next link is processed by decrementing i, and re-applying equations (2.6) and (2.7). Once the entire chain is traversed, the loading conditions at each member of the manipulator are known in the world coordinate system.

The final step is to determine the forces and moments acting in the local frame of each link. This requires transforming all the vectors representing forces and moments calculated in the world coordinate system into the individual link coordinate systems. This is done by using the forward kinematics of the manipulator calculated during the first step of the force resolution process. The transform from each link of the manipulator to the world coordinate system is known. By inverting each of these transformation matrices, the transform from the world coordinate system to each link is obtained. This makes it possible to transform any of the individual loading vectors into any coordinate frame necessary to determine the deflection of the member.

2.2.2 Force and Moment Iteration Scheme

The procedure presented in the previous section described the forces and moments acting on each individual link and joint. These forces and moments are resolved into local link coordinate frames in order to calculate the corresponding link and joint deflections. Since the links of the manipulator are connected in a serial chain, the deflection of link i effects the location of link $i+1$. This means that the deflected link coordinate frames are in a new location relative to world coordinates. This shifting of the link coordinate frames changes the way forces are transmitted through the links. The procedure for determining the local loading configuration for each sub-component of the manipulator utilizes the link coordinate frames determined using the rigid, or non-deflected links. The deflection of the links, however, causes the links farther down the arm to assume a different position. The local loading information determined using the rigid kinematics is then only an approximation to the actual loading.

For small overall deflections, this approximation will be inconsequential. If, however, small link and joint deflections cause large tool deflections, it is necessary to determine the proper link coordinate frame locations and resolve the forces correctly. Resolving the loading into the proper coordinate frame requires knowing the location of that coordinate frame. The coordinate frame location is not precisely known, however, until the effects of deflection have been considered. This circular process requires iteration of the deflection model for a true solution. The steps of the modeling process are summarized in Table 2.1.

Table 2.1 Deflection Modeling Process

Step	Operation
1	Determine the link transforms using the manipulator kinematics.
2	Combine the link transforms.
3	Resolve the forces and moments applied to the manipulator into the link coordinate frames.
4	Calculate the deflection of the individual manipulator components.
5	Determine the deflected link transforms.
6	Combine the deflected link transforms.

In order to converge to the correct link coordinate frame locations, the operations in Table 2.1 are iterated. After completing step 6, the process is re-entered starting at step 3. The iterations continue between steps 3 and 6 until the end-effector position and orientation does not change appreciably from one iteration to the next.

Typically, the difference between the rigid coordinate frame locations and the deflected coordinate frame locations is larger near the end-effector. This is illustrated in Fig. 2.1 where the location of the third link's rigid coordinate frame is substantially different then the location of the third link's deflected coordinate frame. Using the iterative approach outlined above, the frames used to resolve the forces converge to the frames used to re-assemble the deflected manipulator. The number of iterations required to converge within a suitable tolerance should generally be small. The implementation of this model presented in Chapter 5, required only several iterations to converge within reasonable precision.

2.3 Manipulator Component Deflection Model

The deflected link transforms of the manipulator are determined by using equation (2.3) as described in previous sections. This equation incorporates a deflection matrix, $D_{i,\,i+1}$, that is a function of the loads applied to the corresponding manipulator component. Depending on the geometric and material properties of a component, there are many possible methods for modeling this deflection. This section will present one particular modeling scheme that was found to work well for the Cobra manipulator. This method uses pairs of beam and torsional stiffness elements to model the deflection in each link. The specific implementation of this approach is presented in Chapter 5.

Kinematically, a link is considered a rigid-body which defines the relationship between two neighboring joint axes (Craig 1989). The rigid-body assumption no longer applies, however, when deflections are considered. Instead, the deflection of the link must be determined and incorporated into the relationship between the joint axes. Previous manipulator deflection modeling efforts have treated each link as a cantilevered beam that deflects under the applied loads (Derby 1983, Bodur and Derby 1988, Lin, Chiang, and Cui 1990). This approach may be utilized by placing a single beam element along the link.

A single beam element is not always sufficient to model the complex link geometries found on many manipulators. For a complex case, each link may be divided into multiple links, providing the flexibility to assign moment of inertia and modulus of elasticity values for individual segments of the link. In this case, additional entries are added to the link parameter table, and the joint values of these intermediate entries remain fixed.

A manipulator joint exists anytime there is the possibility of relative motion between two links. This relative motion can consist of anywhere from one to six degrees of freedom.

For practical reasons most common manipulators employ either revolute or prismatic joints though all other types of joints can be modeled as a combination of the two basic joint types.

A prismatic joint is modeled using one or more variable-length beam elements. The length used for the beam is determined by the distance between the current link coordinate frame and the next link coordinate frame. Since torsional stiffness elements and beam elements are used in pairs, a prismatic joint may be modeled with a torsional stiffness element at the location of the joint bearings.

The deflection of rotational joints may be described using a single torsional stiffness element located at the origin of the corresponding link coordinate frame. The axis of rotation of a manipulator revolute joint will have some stiffness associated with the drive mechanism, and the remaining axes will have a stiffness associated with the bearings and joint housing. The torsional stiffness element may also be used to model the torsional deflection of the beam element that is attached to it.

2.3.1 Beam Element

The beam element used in this deflection model consists of a uniform material constant cross-section cantilever beam. This element was chosen because a cantilever beam is representative of the deflections commonly found in manipulator links (Derby 1983, Bodur and Derby 1988, Lin, Chiang, and Cui 1990). The deflection of each beam element is determined based on the loading conditions shown in Fig. 2.3, namely a force and a moment at the end of the beam, and a uniform distributed weight load along the beam. The deflections caused by these individual applied loads will be superimposed to determine the total positional and rotational deflection at the end of the beam shown in

Fig. 2.4 (Juvinall and Marshek 1983, Shigley and Mitchell 1993, Beer and Johnston 1981).

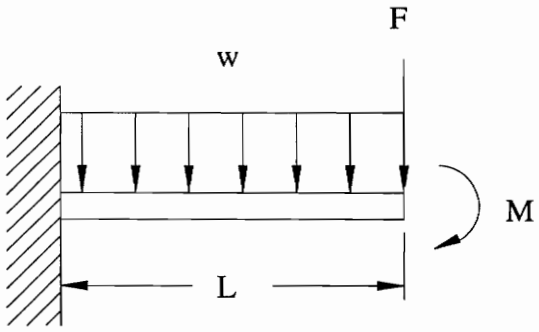

Figure 2.3 Beam Element Loading Configuration

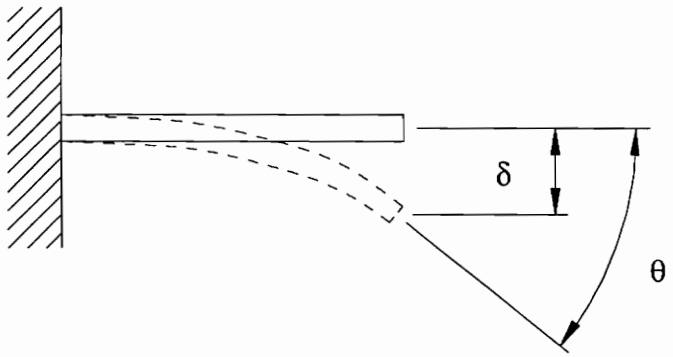

Figure 2.4 Beam Element Deflection

For a force at the end of a cantilever beam, the deflection is

$$\delta = \frac{FL^3}{2EI} \tag{2.8}$$

where

F = force applied to the end of the beam (Fig. 2.3)

δ = positional deflection of the end of the beam (Fig. 2.4)

L = length of the beam (Fig. 2.3)

E = modulus of elasticity of the beam material

I = moment of inertial of the cross section about the neutral axis

The slope at the end of the beam will be

$$\theta = \frac{FL^2}{2EI} \tag{2.9}$$

where

θ = rotational deflection of the end of the beam (Fig. 2.4)

When a moment is applied to the end of a cantilever beam, the deflection of the end is

$$\delta = \frac{ML^2}{2EI} \tag{2.10}$$

where

M = moment applied at the end of the beam (Fig. 2.3)

The resulting slope at the end of the beam will be

$$\theta = \frac{ML}{EI} \tag{2.11}$$

A distributed load applied along a cantilever beam will deflect the beam by

$$\delta = \frac{wL^4}{8EI} \tag{2.12}$$

where

w = distributed load along the beam (Fig. 2.3)

The resulting slope at the end of the beam will be

$$\theta = \frac{wL^3}{6EI} \tag{2.13}$$

When multiple loads are applied simultaneously to a beam, and the beam remains within the linear elastic range, the method of superposition may be applied (Juvinall and Marshek 1983, Shigley and Mitchell 1993, Beer and Johnston 1981). The deflections due to each individual load are summed to determine the total deflection. Summing equations (2.8), (2.10), and (2.12), the total positional deflection is

$$\delta = \frac{FL^3}{2EI} + \frac{ML^2}{2EI} + \frac{wL^4}{8EI} \tag{2.14}$$

By summing equations (2.9), (2.11), and (2.13), the total rotational deflection (slope) is

$$\theta = \frac{FL^2}{2EI} + \frac{ML}{EI} + \frac{wL^3}{6EI} \tag{2.15}$$

Equations (2.14) and (2.15) describe the deflection about a single axis of the beam. Due to the variable loading and changing geometry of a robotic manipulator, a particular beam element may experience deflection about each of its three coordinate axes. For this reason, the positional deflection equation (2.14), and the rotational deflection equation (2.15) are applied about each of the three axes of the beam element.

A torsional load applied to a beam causes an angular deflection proportional to the applied load (Juvinall and Marshek 1983, Shigley and Mitchell 1993, Beer and Johnston 1981). This deflection is

$$\theta = \frac{Tl}{GJ} \tag{2.16}$$

where

T = applied torque

l = length of the beam

G = modulus of rigidity

J = polar area moment of inertia

Since the angular deflection is proportional to the torque on the end of the beam, torsional deflection of a beam element may be modeled as deflection of a torsional stiffness element since the two elements are used in pairs. The next section will explain the torsional stiffness element model in more detail.

The stiffness parameters of each beam element may be determined analytically, or experimentally. An experimental approach employing optimization techniques was found to work well for the manipulators studied in this work. This approach will be presented in Chapter 3.

2.3.2 Torsional Stiffness Element

The torsional stiffness element is based on the deflections commonly found in manipulator revolute joints (Lin et al. 1994). This element has a stiffness about each of its coordinate axes. The rotational deflection about each of these axes is proportional to

the applied moment. The loading configuration of a torsional stiffness element may be applied to a manipulator revolute joint as shown in Fig. 2.5.

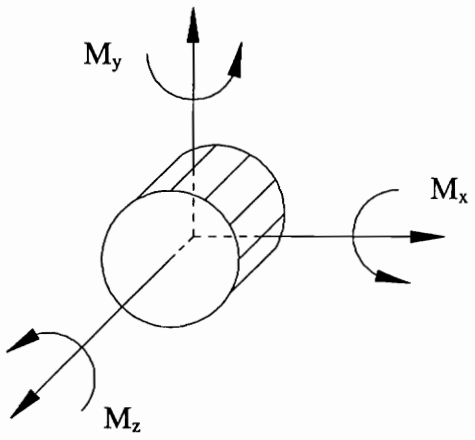

Figure 2.5 Torsional Stiffness Element Loading Configuration

The rotational deflection of the element is modeled as a linear relationship to the joint stiffness,

$$\theta = \frac{1}{K_\theta} M \tag{2.17}$$

where

θ = rotational deflection of the joint

K_θ = the torsional stiffness of the joint

M = the moment applied to the joint

Since moments may be applied about any of the axes of the element, equation (2.17) is applied to each joint axes to yield the resulting θ_x, θ_y, and θ_z.

2.3.3 Component Deflection as a Function of Link Loading

Figure 2.2 shows an individual manipulator link under general loading. To determine the bending of link i, F_{i+1}, M_{i+1}, and w_i are transformed into the local coordinate frame of link i. Equations (2.14) and (2.15) are used to determine the deflection at the end of link i and the slope at the end of link i. The deflection of joint $i+1$ is determined with equation (2.17). Since M_{i+1} is the moment applied at joint $i+1$, each component of the moment vector is used in conjunction with the stiffness about the corresponding axis to determine the rotational deflection of joint $i+1$. This rotational deflection due to the joint stiffness of joint $i+1$ is combined with the positional and rotational deflection due to the stiffness of link i. The result is the total deflection of the link $i+1$ coordinate frame relative to the link i coordinate frame. It is important to note that for both the link and joint deflection models, the basic equations, (2.14), (2.15), and (2.17), are applied to each axis of each joint or link where a load is applied. The corresponding deflections may be represented using x, y, z translations and fixed angle γ, β, α rotations in the deflection matrix,

$$D_{i,i+1} = \begin{bmatrix} & R(\gamma,\beta,\alpha) & & \vec{\delta} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

where

$R(\gamma,\beta,\alpha)$ = X, Y, Z fixed angle rotations due to deflections

δ = X, Y, Z translations due to deflections

The stiffness of each manipulator component is needed in order to determine the corresponding deflection. These stiffness values may be determined using several methods that will be presented in the next chapter.

3. Determination of Manipulator Stiffness Parameters

As explained in Chapter 2, the deflection model uses stiffness parameters for each component of the manipulator to characterize the deflections caused by applied loads. The stiffness parameters may be determined analytically, measured for each individual sub-component, or determined using measured robot position data and optimization techniques. For the manipulators studied in this work, optimization methods yielded the best results. This chapter will focus on an optimization technique developed to determine link and joint stiffnesses using the manipulator deflection model.

3.1 The Deflection Parameter Optimization Method

To characterize the stiffness of a manipulator, it is perturbed from static equilibrium by applying a known load to the end effector. Once the system returns to equilibrium, the deflection caused by the applied load is measured. This process is repeated for a variety of manipulator positions and applied loads. An optimization routine is then used to determine the set of stiffness parameters that best fit the measured end-effector deflection.

For each iteration of the optimization process, the stiffness parameters are used in the deflection model to determine the end-effector position and orientation with and without the applied load. It is important to note that both cases include the deflection of the manipulator under its own weight. This process is applied to all of the measured positions of the manipulator. If the modeled end-effector deflections were identical to the measured end-effector deflections for all the data points, the stiffness parameter set would be perfect, or optimal. For such a case, the flexible model of the manipulator would exactly predict all deflections. In practice, however, the model will not produce perfect

results at all positions. The optimization process is to determine the set of stiffness parameters that yield the least error for all manipulator positions.

3.2 Parameter Selection

The manipulator deflection model presented in Chapter 2 provides each link and joint pair with a minimum of six stiffness parameters. These are the moment of inertia about each axis of the link and the rotational stiffness about each axis of the joint. Additional stiffness parameters may be introduced when a link is modeled as several connected beams. For the minimum parameter set, however, optimizing for the stiffness of a 6 degree of freedom manipulator would require determining 36 unknown stiffness parameters. Though this is possible, it is not usually necessary. Most manipulators are primarily gravity-loaded, and operate in orientations that restrict the loading of the members to one or possibly two axes. All of the stiffness parameters that are determined to be insignificant should be assigned extremely stiff values and removed from the unknown parameter set.

3.3 Objective Function Formulation

The optimization procedure is applied to the set of significant unknown stiffness parameters using end-effector deflection data. There are several common non-linear optimization methods that may be used to solve this problem. The nuances of each method are not discussed here, as that is beyond the scope of this work. Instead, the formulation of the objective function is described, and the selection of the optimization method left to the user. One particular method used for the Cobra manipulator will be mentioned in Chapter 5.

The optimization problem may be stated as a minimization of the objective function,

$$\sum_{i=1}^{n} \left\| \vec{\delta}_i^{\,Measured} - \vec{\delta}_i^{\,Modeled} \right\|^2 \tag{3.1}$$

where

n = the total number of data points

δ_i = end-effector positional deflection vector

Given an initial estimate of the stiffness values, the deflection model is used to determine the loaded and unloaded end-effector location for each test point. The difference between the loaded and unloaded end-effector location is compared with the measured deflection. The sum of the squares of the differences between the modeled and measured deflection values is the objective function given in equation (3.1). For a system measuring only the positional deflection of the end-effector, three objective function contributors are added for each data point. These are the difference in the modeled and measured deflection in the x, y, and z direction. For optimization schemes requiring a single scalar objective function value, the component deflections are combined for each data point.

The objective function values are determined for each set of stiffness parameters that are tested in the optimization cycle. The optimization routine then varies the stiffness parameters until the objective function values are driven near zero. At this point, the deflection model predicts the amount of deflection that was actually measured, and the solution is optimal. It is important to note that while there are many non-linear optimization algorithms suited to this problem, none guarantee convergence to a global optimum solution. Many of these schemes employ multiple sub-optimizations using various starting locations in order to search for the global optimum. In this case, however, the stiffness parameters may be estimated using analytical approximations, yielding a reasonable starting parameter set.

3.4 Deflection Measurement Methods

The stiffness parameter optimization method is based on measuring the deflection caused by perturbing the manipulator from equilibrium with a known load. There are several methods that may be used to measure the deflection of the end-effector. An array of dial indicators could be positioned so as to directly measure the x, y, and z motion of the end-effector. Computer vision techniques could also be used to measure the relative position of the tool to a fixed object in the workspace. In addition, a spark gap acoustic system could be used to measure the positional change due to deflection. Basically, any system for measuring the positional change of an object could be used. Systems providing rotational information in addition to positional information are even more valuable in that each test point may be used to yield additional information. The use of position and rotation measurement systems will reduce the number of data points needed for an accurate solution.

Direct measurement of the end-effector deflection is an ideal way to collect data. The measurement systems needed to do this, however, are generally expensive devices requiring adaptation to a robot tool. In the case of dial indicators, for example, supporting brackets must be designed, and a flat surface must be available on the manipulator end-effector. In addition, the placement of the dial indicators is critical because it determines the axis along which deflections are measured. If a suitable measurement system is not available, the manipulator itself may be used as the measurement device. This approach is appealing since no additional measurement systems are required. The method of measuring robot deflection by using the manipulator itself will be referred to as touch-point optimization throughout this work.

In touch-point optimization, the end-effector is equipped with a pointing device or a sharp tool tip. The tip is driven to a fixed point in the workspace, and the manipulator joint

values are recorded. A known weight is applied to the end-effector, causing the tip to move away from the touch point. The manipulator is then driven back to the same point, and the new joint values are recorded.

The objective function for touch-point optimization is similar to that for direct deflection measurement schemes. The deflection model is given the unloaded set of joint values, and the corresponding tool position is determined. The model is then modified to include the applied load, and the new joint values. Again, the tool position is determined. The objective function contributors are the x, y, and z positional differences between the two modeled end-effector locations. When the difference is driven to near zero for all test points, the stiffness parameters are optimal.

Touch-point optimization has several advantages over direct measurement methods. It is a low-cost solution, requiring only that a pointed tip be placed on the tool. In addition, the manipulator itself is used as a measurement device. This is appealing since most manipulators already contain high-precision joint encoders. This method also partially compensates for errors that are not modeled, such as sensor or drive-train non-linearities.

Despite the advantages of touch-point optimization, measurement methods remain appealing. Less time would be required to obtain deflection data because the manipulator would simply be driven to the test point, and the weight applied. No re-positioning is necessary. In addition to quicker data collection, direct measurement methods may provide higher-precision deflection measurement. This is because the measurement is not performed using a model. Also, the direct deflection measurement methods do not depend on precise positioning of the tool-tip.

4. Real-Time Deflection Compensation

Manipulator controllers perform two basic kinematic operations, namely, forward and inverse kinematics. Given the manipulator joint values, the forward kinematic algorithm computes tool position and orientation. The inverse kinematic algorithm, on the other hand, determines the joint values necessary to yield a specified tool position and orientation. Both kinematic operations assume, however, that the links of the manipulator are rigid. The kinematics of a manipulator may be described as the set of functions that relate tool position to a set of joint values or conversely, a set of joint values to tool position. The kinematic functions of the manipulator change when deflection is considered, and the forward and inverse algorithms must be modified in order to incorporate the deflected link dimensions and joint twists.

To describe the implementation of the deflection model, two types of kinematic models will be used. The rigid kinematic model will be the standard kinematics of the manipulator not considering deflection. The flexible kinematic model, on the other hand, will be the kinematics of the manipulator that incorporate deflection as part of the manipulator kinematics. A set of joint angles will yield a different tool position in the rigid kinematic model then in the flexible kinematic model. The inverse solution will also vary for the rigid and flexible kinematic model.

Both the forward and inverse kinematic routines in a manipulator controller must be modified in order to use the deflection model. The forward kinematics are directly replaced by the deflection model's forward kinematics, since the deflection model is a more accurate representation of the manipulator. The inverse kinematics, however, require more involved modifications. This is because inverse kinematic solutions generally exploit geometric properties which are altered when deflection is considered. Introducing link deflections into the solution will generally make closed form techniques

intractable. This chapter will provide a general discussion of how the deflection model is incorporated into the control algorithms of a manipulator. This general scheme will need to be adapted to specific controllers, but the concepts will remain the same.

4.1 Forward Kinematics

As mentioned previously, the deflection model provides the forward kinematics of the manipulator. It determines the deflected tool position and orientation given a set of joint values, the manipulator's stiffness parameters, and loading information. In order to accurately represent the position of the end-effector for a given set of joint angles, the deflected tool position must be used. For this reason, the flexible forward kinematics of the manipulator should be used in place of the rigid forward kinematics.

The flexible forward kinematics use deflected link transforms as opposed to the standard Denavit-Hartenberg link transforms. These deflected link transforms are obtained by applying the rotations and translations caused by deflection to the standard link transforms. The transformation from end-effector coordinates to manipulator base coordinates is determined by combining all of the deflected link transforms. This process was described in detail in Chapter 2.

4.2 Inverse Kinematics

The complex problem of inverse kinematics is usually solved by exploiting certain geometric properties of the manipulator that simplify its solution to a polynomial of degree four or less (Cordle 1993). These properties include intersecting joint axes, links with no offsets, and 0 or 90 degree link twist angles. When deflection effects are

considered, however, these simplifying properties may disappear, making existing inverse kinematic algorithms inadequate.

For a specific goal point, both the flexible kinematic model and the rigid kinematic model of the manipulator will yield different inverse solution results. Since the effects of deflection are generally small, however, the two inverse solutions will be similar. As a result, the inverse solution obtained using the rigid model may be used as the initial guess for an iterative solution scheme applied to the flexible kinematic model. This will eliminate some common iterative kinematic solution problems, such as converging to a valid, though undesired solution.

The iterative solution of the inverse kinematics of a manipulator is based on testing sets of joint values using the forward kinematics until the correct end-effector position and orientation is achieved. The results of each successive joint value test are used to determine the next test set. There are several numerical methods that provide the solution in a small number of iterations. The Newton-Raphson root-finding method (Cheney and Kincaid 1985) is a common scheme used for solving for one unknown variable. This method may also be applied to multiple variable systems, such as the flexible inverse kinematics problem using the inverse of the manipulator's Jacobian matrix (Cordle 1993, Thomopoulos and Tam 1990, Goldenberg and Lawrence 1985, Xi and Fenton 1994).

The Jacobian of a manipulator is a multidimensional derivative. It relates joint velocities to Cartesian end-effector velocities. The Jacobian is commonly used to determine the joint velocities necessary to maintain a specified Cartesian velocity. It is also used to determine the joint torque (or force in the case of a prismatic joint) necessary to maintain a specified force at the end-effector (Craig 1989). This matrix may be determined analytically or numerically.

Numerical evaluation of the Jacobian, is straightforward, given the flexible forward kinematics of the manipulator. For a set of joint values, the tool position is determined. This is the bench-mark tool-position that will be used to determine the relative effect of the joint values on each of the output parameters. A small change is made to a single joint value while holding all other joint values constant. The flexible forward kinematics is used to determine the tool position resulting from the perturbed joint value. The difference between the perturbed tool position and the bench-mark tool position is divided by the amount of change in the joint value. The result is the derivative of tool position with respect to a particular joint value.

The components of the tool position and orientation derivative are entered as a columns in the Jacobian matrix. Each column corresponds to the variable that was perturbed to yield the derivatives. After determining a column of the Jacobian, the perturbed joint value is returned to its original state, and the next joint value is perturbed. The process is repeated until the entire matrix is populated.

Once the Jacobian matrix, *J*, has been determined, it is used to determine the next guess for each iteration of the inverse solution. The evaluation of the flexible forward kinematics yields the tool position and orientation for a test set of joint angles. This test set is compared to the desired goal location to yield the error vector,

$$\vec{E} = \vec{P}_{Test} - \vec{P}_{Goal} \tag{4.1}$$

This error vector becomes the right-hand side of the inverse kinematic equation system. The goal is to drive this vector to zero. In order to determine the next test set of joint angles, the following equation is used:

$$\delta\vec{x} = [J]^{-1}\,\vec{E} \tag{4.2}$$

The calculated change for each of the joint values, δx, is applied to the joint value set of the previous iteration to determine the next test set of joint values,

$$\vec{x}_{i+1} = \vec{x}_i - \delta\vec{x} \cdot \mu \tag{4.3}$$

where

μ = a scalar damping coefficient, $0<\mu<1$

Equation (4.3) yields the test set of joint values for the next iteration. This process is repeated until the right-hand side of the equation system, *E*, is driven suitably close to zero. At that point, the test set of joint angles yield the desired tool position and orientation, and the inverse kinematics problem is solved. Sometimes, however, there are difficulties in converging to the exact solution. To make the convergence more robust, the damping coefficient may be decreased. This damping effect may result in a higher number of iterations, but increases the robustness of the algorithm.

It is also important to note that the Jacobian matrix of the manipulator will become singular when the manipulator reaches a singularity in its workspace. This singular condition prohibits inversion of the Jacobian. There are several methods commonly used for identifying singular configurations of manipulators. Xi and Fenton (1991) applied the line cross product method to detect singular configurations when solving the kinematics of flexible manipulators. Salerno (1993) applied the method of singular-value decomposition to the control of truss-type manipulators which often yield ill-conditioned Jacobians. Generally speaking, manipulator controllers are designed to avoid singularities and will not attempt to pass through them. For this reason, deflection compensation is not necessary in singular regions. For a manipulator using a

sophisticated controller to navigate singular regions, the iterative solution of the flexible kinematics will require more advanced methods.

4.3 Trajectory Generation

There are several methods used for generating manipulator trajectories. These methods plan a path from a starting location to a goal location using multiple via points. These via-point specifications are converted into joint value specifications using the inverse kinematics of the manipulator. Once the via points have been determined, the joints are driven through the specified path by joint controllers.

The deflection model is easily implemented into the trajectory generation scheme of a manipulator controller. Several implementation methods are possible, and the selection of the proper method depends on the type of path desired. For many manipulator operations the accuracy of the manipulator is most important at the goal point. In this case, it is only necessary to perform the deflection analysis on the goal point itself. If, however, it is necessary to compensate for deflection at every point along the manipulator's trajectory, a different implementation scheme is required.

4.3.1 Goal-Point Deflection Compensation

To incorporate deflection compensation of only the goal point into trajectory generation, it is only necessary to modify the goal-point used in the existing trajectory generation algorithm. The actual user-specified goal point is used as an input to the flexible inverse kinematics described in Section 4.2. The flexible inverse kinematic algorithm determines the destination joint values for the flexible manipulator. These joint values are the values at which the manipulator must complete its path in order to arrive at the goal point.

Since only the destination joint values are of interest, these joint values are mapped into the rigid kinematic model of the manipulator. This is done by using the goal-point joint values from the flexible inverse algorithm as inputs to the rigid forward kinematic model. The result is the goal-point specification expressed in terms of the rigid kinematics of the manipulator. The rigid model goal-point is used as the input to the standard trajectory generation calculations of the manipulator. The via-points are then determined using the rigid kinematic model. The approach allows for the trajectory generation scheme of the manipulator to remain unchanged. Only the goal point specification input to the trajectory generation system is changed.

This process is attractive in that it adds only minimal computational burden to the manipulator controller. The existing trajectory generation module of the manipulator is still used, only its input is mapped from the flexible to the rigid kinematic model.

4.3.2 Via-Point Deflection Compensation

When it is necessary to compensate for static deflections at every point along a manipulator's trajectory, goal-point deflection compensation is not adequate. Instead, each point of the trajectory is determined using the deflection model.

The implementation of this scheme into trajectory generation only affects the mapping of the via points from world coordinates to joint space. This mapping is accomplished using the inverse kinematic solution of the manipulator. By replacing the rigid inverse kinematics of the manipulator with the deflected inverse kinematics described in Section 4.2, every via point will be determined based on the deflection of the manipulator at that particular configuration.

The computational burden of this approach is substantially larger then the goal-point approach in the preceding section. For each via point, the flexible inverse kinematics of the manipulator will be executed. Depending on the implementation of the flexible kinematics, this may require multiple iterations of the flexible forward kinematics, and inversion of the Jacobian matrix as discussed in Section 4.2. This approach will, however, compensate for static deflections for each point along the trajectory of the manipulator. Given the increased power of today's computing machines, this thorough approach is becoming increasingly viable.

5. Application to the Cobra Manipulator

The real-time deflection compensation method presented in this work was motivated by the development and refinement of the Cobra manipulator series. B&W Nuclear Technologies began development of the first Cobra manipulator in 1989. Cobra was developed to assist in the inspection and repair of nuclear power plant steam generators. The tasks performed by Cobra include eddy current inspection of steam generators, as well as repair of failed steam generator tubes via plugging, sleeving, and welding. The design and development of Cobra is described in detail by Tidwell et al. (1991). Figure 5.1 shows Cobra and its base inside a nuclear steam generator.

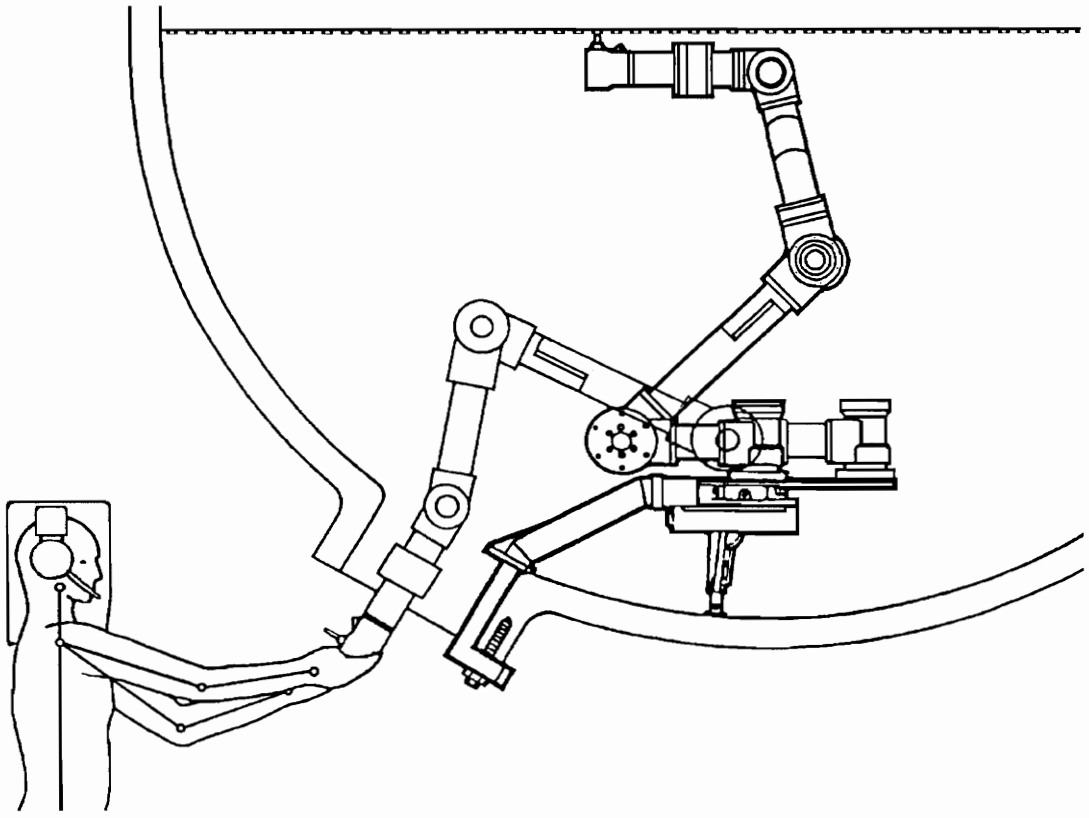

Figure 5.1 The Cobra Manipulator

Cobra is a four-degree-of-freedom, all-revolute manipulator. The joints consist of high-speed (10,000 RPM) motors and harmonic drive gear reduction units (10,000:1). The second link of the manipulator includes a latching device used to break the manipulator down for transportation and installation. The manipulator is installed into a steam-generator through an 18 inch diameter access porthole, called the manway. Installation is performed using a device called the mezzanine and a removable track.

Cobra was specifically designed to operate in the Limited Access Workspace (LAW) of a nuclear steam generator quarter sphere bowl. The LAW design process, as it pertains to the Cobra system, is described by Shooter, Reinholtz, and Dhande (1992). Since the manipulator was designed around its environment, the result was a relatively lightweight, portable, semi-modular manipulator.

Cobra is considerably more flexible and lightweight then common industrial manipulators. In fact, Cobra is a near worst-case scenario for deflection modeling. Cobra weighs less than 150 lb., and is capable of lifting over 100 lb. at a full extension of 6 ft. Given the high strength to weight ratio of Cobra, the stiffnesses of the links and joints are important factors in the positioning accuracy of the system. When using some of the heavier repair tools, the accuracy of Cobra is substantially degraded. In addition to the flexibility of the manipulator itself, the flexibility of the mezzanine causes significant end-effector deflections. The success of the deflection model for Cobra indicates that this model may be successfully applied to other manipulators.

The installation of Cobra into a steam generator requires first inserting the mezzanine into the steam-generator manway. After inserting the mezzanine and securing it to the manway, a track is attached to the mezzanine. Cobra is broken into two parts by releasing the latch on the second link. Next, the lower portion of the arm is attached to the track and propeled into the steam generator. Once the lower portion of Cobra is almost inside the steam generator, the rest of the manipulator is latched to the second link. The entire

manipulator then travels to the end of the track where the base of the manipulator is locked to the mezzanine platform using a series of air-actuated cam devices. After touching three known calibration points in the steam-generator cavity, Cobra is ready to begin performing its inspection and repair tasks.

The latching mechanism in Cobra's second link introduces a considerable amount of flexibility. Prior to this work, the latching mechanism was suspected to be the major contributor to Cobra's deflection. The results of the stiffness parameter optimization presented in this chapter confirm that the second link is considerably more flexible then the other members.

There are several models in the Cobra manipulator series. The deflection modeling and compensation approach developed in this thesis was applied to two of these manipulators. Tube Sheet Mounted (TSM) Cobra was designed with an additional degree of freedom and a longer reach than the original Cobra manipulator. TSM-Cobra was the first manipulator where the deflection compensation approach was implemented. The majority of the deflection work, however, was focused on the original Cobra manipulator. For this reason, the remainder of this chapter will focus on the implementation of deflection compensation into the original Cobra manipulator though a similar process was applied to TSM-Cobra.

This chapter presents the application of the deflection model to the Cobra manipulator and mounting platform. The characterization of Cobra's stiffness is described, and the implementation of the model into the existing control system is presented. The implementation includes goal-point deflection compensation for the trajectory generation algorithms, iterative inverse solutions for the goal-point specifications, and iterative applications of the flexible forward kinematics to the calibration process.

5.1 Cobra Kinematics

The kinematic description of the Cobra manipulator is represented by the Denavit-Hartenberg link parameter notation in Table 5.1, and shown in Fig. 5.2. The kinematic parameters are used to describe the location of the end-effector relative to the manipulator base that is mounted on the mezzanine platform.

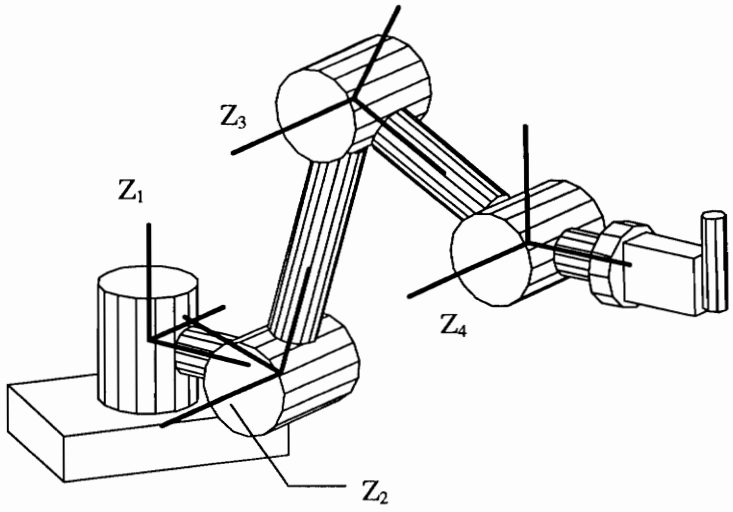

Figure 5.2 Kinematic Diagram of the Cobra Manipulator

Table 5.1 Denavit-Hartenberg Parameter Table for Cobra
(angles in degrees, lengths in inches)

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	90	13.202	0	θ_2
3	0	26.406	0	θ_3
4	0	19.400	0	θ_4

Cobra is controlled by a Hewlett Packard 700 series computer interfaced to custom joint controller cards via a serial data network. The control code was written in C under the

Unix environment using Motif and XWindows for the user interface. The deflection model was implemented directly into the kinematics and trajectory generation modules of the Cobra software.

5.2 Modeling Approach

The computer code written to accomplish the modeling scheme presented in Chapter 2 was developed as a general manipulator deflection package. The code was then applied to the Cobra system by specifying the link-parameters and deflection parameters in a data file. One example of this general approach is the kinematics used in the deflection model code. The Denavit-Hartenberg link parameters are stored as part of the deflection modeling file. The model then uses these parameters to construct the link transforms and perform the kinematics. Though this approach is more computationally intensive than hard-coding the link transformations, it does make the model extremely general. By changing the input file, the deflection model computer code will simulate deflections for any serial manipulator. The rest of this chapter will focus on determining the parameters that populate the deflection model input file for the Cobra manipulator.

The deflection model implemented for the Cobra system includes both the mezzanine and the manipulator. Passive link and joint elements are attached to the base of the manipulator to represent the stiffness of the mezzanine structure. The two-step optimization process used to determine this configuration is presented in Section 5.3.

The manipulator and mezzanine are analyzed separately for several reasons. First, it was initially unclear what the type of model would be necessary to describe the flexibility of the mezzanine. The manipulator flexibility, however, is described by the method presented in Chapter 2. Second, the Cobra manipulator will possibly be used on several alternate base platforms that are substantially different from the mezzanine. By modeling

only the manipulator, the result is useful for a wider variety of applications. Finally, optimization methods are generally simpler when less unknown parameters are considered. The two-step process reduced the amount of unknowns for each sub-optimization.

The mezzanine stiffness is incorporated into the deflection model of the manipulator system by first modifying the Denavit-Hartenberg link-parameter table for the manipulator. Two extra rows are inserted into the beginning of the table, representing the mezzanine members. The link-parameter values used to represent the mezzanine were determined using optimization methods that will be described in the next section. Figure 5.3 depicts the additional members, and the new coordinate frame locations. Table 5.2 details the Denavit-Hartenberg representation of the entire system.

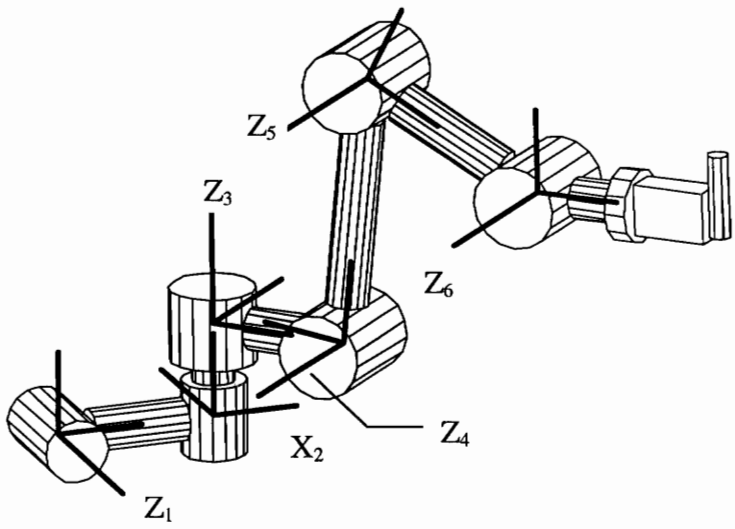

Figure 5.3 Kinematic Diagram of Cobra and the Mezzanine

Table 5.2 Denavit-Hartenberg Parameter Table for Cobra and the Mezzanine (angles in degrees, lengths in inches)

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	90	0	0	0
2	-91.37	9.36	0	0
3	0	0	4.01	θ_3
4	90	13.202	0	θ_4
5	0	26.406	0	θ_5
6	0	19.400	0	θ_6

The rigid kinematic model presented in Chapter 2 may be applied to the mezzanine and manipulator using the modified link parameter table. The flexible kinematic model of the manipulator, on the other hand, must utilize stiffness information pertaining to the mechanism described by the parameter table. The process used to determine the stiffness information for the sub-components of Cobra is described in the following section.

5.3 Stiffness Parameter Optimization

The initial implementation of the Cobra deflection model used link stiffnesses and joint stiffness that were calculated analytically based on link cross sectional areas. These calculations required many simplifying assumptions due to the complexities of Cobra's joints and links. Though use of the analytical stiffness values did improve the positioning accuracy of Cobra, the results were not adequate.

Direct measurement of the individual deflection of each link was also attempted. This was done by stretching the manipulator out and instrumenting each link with four dial indicators. A known load was applied to the end-effector, and the deflection seen at each

dial indicator was recorded. The stiffness of the link and joint was calculated by fitting a curve to the measured deflections and determining the stiffness constants. This approach improved positioning accuracy over the analytical methods, but did not yield satisfactory results.

Initially, optimization methods were selected only because the other methods did not yield satisfactory results. As it turned out, however, the optimization methods were less time consuming and far more accurate.

Since the manipulator and the mezzanine platform are analyzed separately, the coordinate frame numbering system described in Fig. 5.3 and Table 5.2 will be used to describe both processes. Note that the first active joint of the manipulator is joint 3. This numbering convention will simplify the process of combining the stiffness data into a model of the entire system.

5.3.1 Unknown Stiffness Parameters

In Chapter 3, the general approach for characterizing manipulator stiffness was presented. This approach is applied to the Cobra manipulator, beginning with the selection of the significant stiffness parameters.

The Cobra manipulator experiences gravity loading in the Z_0 coordinate direction. Usually, Cobra is mounted so that gravity acts in the negative Z_0 direction, although some installations require mounting the manipulator upside-down. The loads on the manipulator result from both the weight of its components and end-effector. Stiffness coefficients not affected by gravity loading are assumed to be negligible.

When a vertical load is applied to the end-effector, the last link of the manipulator is subject to a loading with components along the X_6 and Y_6 axes. This loading results in a moment about the Z_6 axis. The moment of inertia of the 6^{th} link about the Z_6 axis, I_{z6}, is used to determine the resulting deflection of the end-effector. In addition to I_{z6}, the rotational stiffness of the 6^{th} joint, K_{z6}, contributes to the resulting deflection.

The link coordinate frames of joints 6, 5, and 4 are all related in that their Z axes are parallel. This means that gravity loading will cause deflection described by the same coordinate axis stiffness parameters for each of the links and joints attached to frames 4, 5, and 6. For this reason, the additional stiffness parameters, I_{z5}, K_{z5}, I_{z4}, and K_{z4} are all added to the significant parameter list.

Link coordinate frame 3, however, does not share the common Z axis of the previous three links. Instead, the Y axis of coordinate frame 3 is parallel to the Z axes of frames 4, 5, and 6, as seen in Fig. 5.3. The stiffness parameters pertaining to loading about the Y_3 axis, I_{y3} and K_{y3}, are added to the significant parameter list. At this point, the important stiffness characteristics of the manipulator have been taken into account.

The unknown stiffness parameters resulting from the mezzanine must also be determined. These parameters were selected after several iterations of the optimization process. Once the proper configuration was selected, the location and orientation of the additional passive links was optimized in addition to the unknown stiffness parameters. Figure 5.3 illustrates the location of the passive links and joints that were finally used to describe the mezzanine. The significant set of unknown stiffness parameters is determined using the same process used for the parameters of the manipulator.

The rotational stiffness about the X_2 axis, namely K_{x2}, is used to model the torsional stiffness of the platform assembly. The beam along the X_1 axis, with moment of inertia, I_{z1}, models the bending of the mezzanine structure. There is an additional torsional

stiffness about the Z_1 axis, K_{z1}, modeling pure rotation about the origin of the combined manipulator coordinate system. The selection of these parameters was based on knowledge of the construction of the mezzanine device, and the location of the joints and links was determined using optimization methods.

The unknown stiffness parameter set is K_{z1}, I_{z1}, K_{x2}, I_{y3}, K_{y3}, I_{z4}, K_{z4}, I_{z5}, K_{z5}, I_{z6}, and K_{z6}, as shown in Table 5.3. These 11 unknowns are determined using the touch-point optimization approach described in Chapter 3.

Table 5.3 Unknown Stiffness Parameters for Cobra and the Mezzanine
(blank entries indicate parameters not needed for this geometry and loading)

	Moment of Inertia, I			Rotational joint stiffness, K		
Link	x	y	z	x	y	z
1			I_{z1}			K_{z1}
2				K_{x2}		
3		I_{y3}			K_{y3}	
4			I_{z4}			K_{z4}
5			I_{z5}			K_{z5}
6			I_{z6}			K_{z6}

5.3.2 Deflection Measurement and Data Collection

Chapter 3 presented the process used to collect deflection data. This data is used to determine the stiffness parameters of the manipulator. The idea is to apply a known load to the end-effector, and measure the resulting deflection. By repeating this process over a

wide range of manipulator positions, and applied loads, enough data is obtained to determine the stiffness of the individual links and joints of the manipulator.

An accurate position measurement device that could be easily adapted to the Cobra end-effector was not available. Instead, the touch-point optimization process described in Chapter 3 was used to measure the deflection. Since the goal of touch-point optimization is to return the tool to the same position in the workspace, a pointed tip was attached to the end-effector.

In order to determine the stiffness of the individual manipulator sub-components, it was necessary to decouple the deflection of the manipulator from the deflection of the mezzanine. To do this, the manipulator was removed from the mezzanine, and bolted to a rigid platform. Another rigid platform was constructed above the manipulator, and marked with a series of points.

The pointed tip of the Cobra tool was driven to a marked point on the rigid platform, and the joint values were recorded. Next, a 20 lb. weight was attached to the tool tip, and the manipulator deflected away from the touch-point. The Cobra tool was driven back to the same point, and the joint values were recorded. This process was repeated for the entire series of points using 20, 30, and 40 lb. weights. The result was a set of 31 data points.

In addition to obtaining deflection data, it is necessary to determine the weight of the individual sub-components of Cobra. The deflection model uses these weights to determine the deflection of Cobra under its own weight in addition to the weight of the attached end-effector. In the case of Cobra, the weight of each individual joint is treated as a lumped mass located at the origin of the corresponding link coordinate system. For link i, the lumped end-load is the weight of joint i+1. This is a reasonable assumption since the joints are circular, and symmetric about the center of rotation. The weight of each link, however, is treated as a uniform distributed load applied along the link. This

assumption is also reasonable since the links of Cobra are relatively uniform. The load applied to the last link due to the end-effector is modeled as a distributed load applied along the length of the tool. During the data collection process, a weight is added to the end of the tool. For this case, the applied load is modeled as the lumped weight at the end of the last link.

The weights of the individual sub-components of the manipulator, including links, joints, cables, and assorted hardware were measured. The distributed link loads were determined by dividing the total distributed weight by the length of the corresponding link. For the passive links that describe the mezzanine, the weights are not considered. This is because they cause deflection that does not change as the manipulator moves. Since the position and orientation of the mezzanine is determined using a calibration process, the static equilibrium offset position of the mezzanine is not important. The measured loading data for Cobra is summarized in Table 5.3.

Table 5.4 Lumped and Distributed Loading for Cobra

Link	**Lumped End Load** (lb.)	**Distributed Load** (lb./in)
1	0	0
2	0	0
3	26.7	0.152
4	20.0	0.403
5	10.4	0.126
6	0	1.2062

Once the data for the manipulator arm was taken, the system was re-assembled and placed in a steam-generator mockup. The tube-sheet in the mockup facility was used at the touch surface, and several points were selected. The same data collection procedure employed for the arm was used for the mezzanine, and a series of 14 data points were obtained. Less data is needed for the mezzanine stiffness optimization because there are only three unknown stiffness parameters.

5.3.3 Optimization Process

The optimization process described in Chapter 3 was implemented using the general deflection model code to evaluate the objective function. For the optimization of the arm stiffness parameters, 8 unknowns were determined using a total of 31 data points. Initially, a Hooke and Jeeves optimization scheme was used (Rao 1979). Though simple and in some cases less efficient then other more sophisticated techniques, this straightforward approach is a powerful optimization tool requiring only minimal implementation code (Reinholtz 1983). For this application, the Hooke and Jeeves optimization yielded good results, though it required approximately one hour to run. Later, another method was employed utilizing the direction of steepest descent in conjunction with line-searching techniques. This method converged to an optimum more quickly, but involved sophisticated algorithms and was found to be less robust than the Hooke and Jeeves method. The results presented in this chapter are those from the Hooke and Jeeves optimization technique.

The mezzanine stiffness determination process was similar to that for the arm. The difference was the deflection model input file that contained additional links and joints and the unknowns that included not only stiffnesses, but link lengths and twists. Due to the complex geometry of the mezzanine, the location of the mezzanine stiffness members

was also an unknown. The stiffness values for the arm, however, were fixed at the values determined from the arm optimization.

The optimization process yielded both the stiffness values for each link in the model and the link-parameters used to model the mezzanine. The link-parameters of the mezzanine determined using the optimization process were given previously in Table 5.2. The calculated stiffness values for both the arm and the mezzanine are given in Table 5.4.

Table 5.4 Stiffness Values Determined Using Optimization

	Moment of Inertia, I (in^4)			Rotational joint stiffness, K (in-lb./deg)		
Link	x	y	z	x	y	z
1			0.426			272,000
2				6,543		
3		$1(10)^9$			$37(10)^{15}$	
4			0.394			$1(10)^{15}$
5			$1(10)^9$			207,000
6			$1(10)^9$			5,588

If the objective function value were zero for the parameters given in Table 5.4, the resulting model would accurately predict all of the deflections in the manipulator for each of the data points. Due to the approximations made in the model and the measurement errors introduced into the data set, the objective function value did not reach zero. When it was determined that the objective function value did not change significantly after many iterations, the optimization loop was stopped. At this point, the parameters were said to be optimal despite the remaining errors. For the arm optimization, the individual

errors at each touch-point are given in Table 5.5, and the touch-point errors for the mezzanine parameter optimization are given in Table 5.6.

Table 5.5 Cobra Arm Model Touch-Point Error at Optimum
(inches)

0.015461	0.034366	0.041367	0.015440	0.026947	0.065858	0.026121
0.019088	0.005295	0.026862	0.009850	0.013486	0.022496	0.029101
0.018785	0.016528	0.012403	0.001181	0.007778	0.024610	0.018524
0.049513	0.029370	0.049654	0.014296	0.025922	0.018081	0.018745
0.035384	0.025021	0.041279				

Average = 0.0245 inches

Table 5.6 Cobra Arm and Mezzanine Model Touch-Point Error at Optimum
(inches)

0.025741	0.054024	0.037818	0.027512	0.034029	0.048146	0.017627
0.033851	0.010944	0.016305	0.050814	0.014880	0.012998	0.035348

Average = 0.0300 inches

The errors calculated at the conclusion of the optimization process are well within the expected range. Since the touch-point optimization procedure was used, the expected positioning error in the tool is on the same order of magnitude as the data point errors.

The optimal parameters given in Table 5.4 show a reasonable correspondence to the expected stiffness of the manipulator and its platform. The fourth link, for example, has a low moment of inertia value and is therefore very flexible relative to the other

parameters. This link was thought to be responsible to a large amount of the deflection because it contains the latching device designed to allow Cobra to be separated for transportation. Also, the first link, or the mezzanine beam element, is relatively flexible. This is expected because the deflection of the mezzanine heavily contributes to the overall deflection.

5.4 Controller Implementation

The general deflection modeling code was written in ANSI-C as defined by Kerninghan and Ritchie (1988). This modular software package was implemented directly into the controller code for the Cobra manipulator. A brief description of the software architecture is given in Appendix A along with a header file that describes the central data structures.

A goal-point deflection compensation scheme as described in Chapter 4 was implemented into the trajectory generation module of Cobra. The nature of the repair work that Cobra performs requires positioning the end-effector directly beneath a specific tube of the steam-generator. It is therefore only necessary to incorporate deflection into the final destination point for the end-effector.

Calibration is performed using several known tool touch-points. By touching the known locations, and storing the corresponding joint values, the base position and orientation may be determined as described by Shooter and Reinholtz (1992). This calibration process relies heavily on the kinematic description of the manipulator. By using the flexible forward kinematics module of the deflection model, the proper tool transformation matrix may be determined. This provides for significantly more accurate touch-calibrations.

In addition to goal-point compensation and touch-calibration, the deflection model was also used to provide accurate operator feedback. Twice a second, the forward kinematics of the manipulator are called using the current joint values. The forward routine then calculates the position and orientation of the end-effector and displays it on the operators user interface. The deflection model was implemented into this module of the controller to provide more accurate position representation.

The model was implemented into the Cobra controller in a manner that is transparent to Cobra's operators. The kinematics have been heavily modified, but the user-interface remains the same. In fact, after the algorithms were implemented into forward kinematics, inverse kinematics, and trajectory generation, there were no noticable time delays.

To better understand the computational burden of the deflection model, execution tests were performed on a 486 DX personal computer running at 33 MHz. A single iteration of the deflection model will execute at a rate of approximately 109 Hz, or 9.2 milliseconds. This rate was determined by running the model through 10,000 iterations and measuring the elapsed time Though the deflection model may normally iterate several times for each call, this execution speed was more then adequate for this application.

5.5 Results

The positioning accuracy of Cobra was substantially improved by the implementation of the deflection model. Though the end-effector deflected as much as 1.5 inches due to the manipulator's own weight, deflection compensation allowed the manipulator to perform tasks requiring a positioning accuracy of 90 mills. This improvement is due in part to the increased touch-calibration accuracy. The compensation of the end-effector deflection at the goal point also contributed significantly to the positioning accuracy improvements.

In order to quantify the improvements in positioning accuracy, a series of calibration and testing sequences were performed both with and without the deflection model activated. The tests were performed with the manipulator mounted on the mezzanine, installed in the steam-generator mockup. The tubesheet of the mockup contains approximately 5,000 0.760 inch diameter drilled holes. The centers of these holes are well known since the tube-sheet was built to simulate the precisely machined tube-sheets found in nuclear power-plants. The performance of the manipulator was evaluated based on its ability to accurately position beneath specific tubes. A calibration tool-tip, shown in Fig. 5.4, was placed on the end-effector, allowing Cobra to insert the tip into tubes to evaluate positioning accuracy.

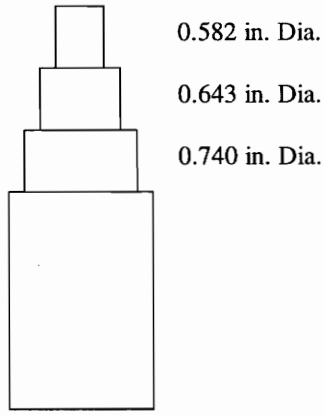

Figure 5.4 Measured Diameters of the Calibration Tool-Tip

The manipulator was calibrated by inserting the third step of the calibration tool into three different calibration holes. Since the holes are 0.760 inches in diameter and the third step of the tool is 0.740 inches in diameter, the calibration tool is centered in the tube within 0.010 inches. The tip was inserted without applying additional loading to the arm since this would degrade the performance of the deflection model. The tool transforms determined using the flexible forward kinematics were used in the existing calibration

algorithms to calculate the position and orientation of the mezzanine relative to the world coordinate system.

The resulting accuracy of the model was evaluated by attempting to center the calibration tip under a set of 10 holes distributed throughout the tube-sheet. The approximate locations of the calibration holes are shown in figure 5.5 along with the locations of the test holes. The calibration holes were also used as test holes.

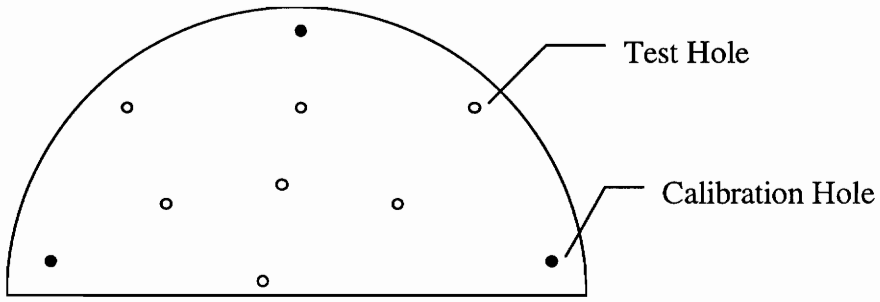

Figure 5.5 Tube-Sheet Test Locations

The accuracy of the manipulator is given in terms of the number of calibration tool steps that could be inserted into the test holes without re-aligning the tool-tip or excessively loading the arm. The four possible insertion cases are shown in Fig. 5.6. Table 5.7 shows the total number of test holes that corresponded each of the insertion cases both with and without deflection compensation. If the deflection model had yielded perfect performance, all three steps on the insertion tool would have inserted into every test hole. If none of the calibration tool steps inserted into a hole, the error is at least the difference in radius of the hole and the first step. For many of the cases where the first tip would not insert, however, the error was substantially larger. This was verified using a camera mounted on the tool. Additionally, if the calibration tool contacted the hole during insertion, the errors may be larger then the tool tolerance due to the compliance of the manipulator.

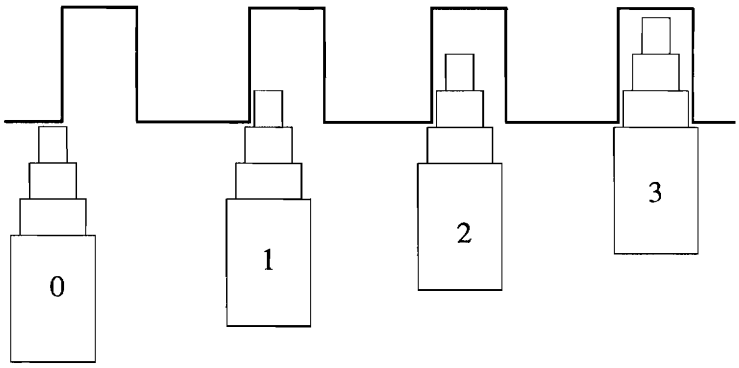

Figure 5.6 Deflection Model Test Insertion Cases

Table 5.7 Deflection Test Results

Case	Centering Error	Number of Test Holes (10 total)	
(Fig. 5.6)	(in)	Without Deflection Compensation	With Deflection Compensation
0	> 0.089 (no steps inserted)	7	0
1	0.089 (1 step inserted)	2	4
2	0.058 (2 steps inserted)	1	4
3	0.010 (3 steps inserted)	0	2

Deflection compensation significantly improved the positioning accuracy of Cobra for the test holes. Many additional tests were performed using different calibration hole locations, and different test hole locations. These all demonstrated results similar to those in Table 5.7. They were not included here, however, because most of the tests were performed only with deflection compensation on and therefore could not be compared to rigid kinematic results.

5.5.1 Numerical Examples

This section will provide several numerical examples to illustrate the deflection in the Cobra manipulator. The optimal stiffness parameter set was used in the deflection model for the mezzanine and manipulator. For each of the manipulator configurations shown below, three tests were performed. First, the rigid kinematics of the manipulator were used to determine the tool tip location. Next, the flexible kinematics were used to determine the tool tip location with the manipulator deflected under its own weight. Finally, an additional 30 lb. was added to the tool weight to simulate the deflection during the touch-point optimization process. The manipulator joint values used for each example configuration are given in Table 5.8. Figure 5.7 shows the first example with Cobra elongated to illustrate a near worst-case deflection scenario. Figure 5.8 shows Cobra retracted to illustrate a stiffer configuration. For both figures, the tool coordinate frame is draw for each of the loading cases, but the tool tip is not drawn for clarity.

Table 5.8 Joint Values for Examples

Joint	Example 1 Fig. 5.7 (deg.)	Example 2 Fig. 5.8 (deg.)
θ_1	-36.688	73.312
θ_2	46.138	126.138
θ_3	-35.856	-95.856
θ_4	-11.127	-31.127

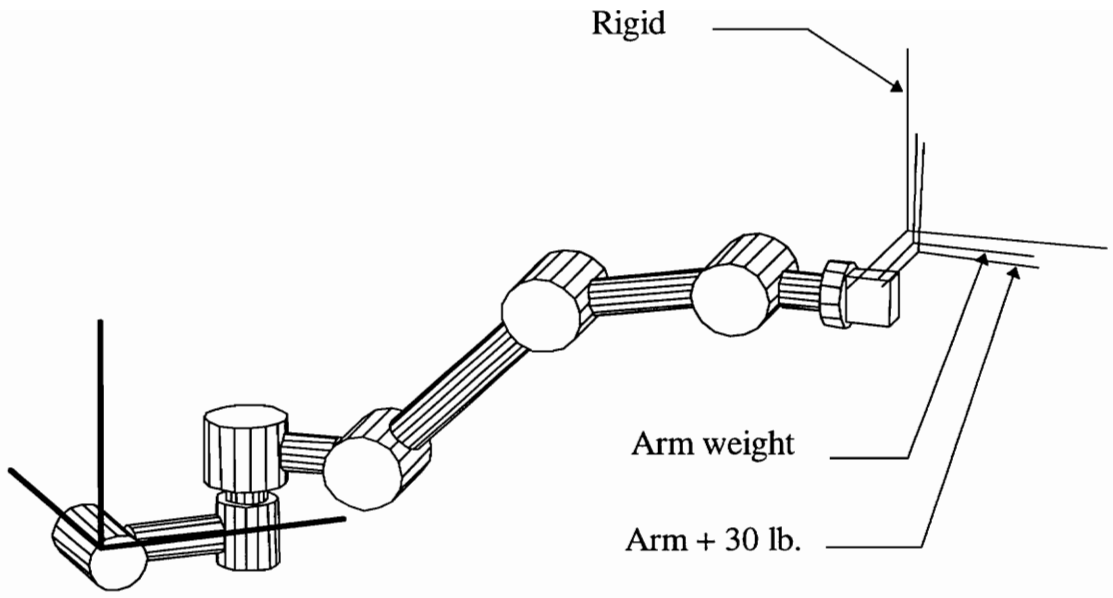

Figure 5.7 Cobra and Mezzanine for Example 1

For the configuration shown in Fig. 5.7, the modeled deflection of Cobra is given in Table 5.9. The position and orientation componenents of the transform from tool to base coordinates are given as a reference, and the changes in the comonents are given for each loading scenario.

Table 5.9 Deflection Data for Example 1

Example 1 Fig. 5.6	Position Change (inches)				Orientation Change (degrees)		
	X	Y	Z	Mag.	γ	β	α
T_{Tool}^{Base} Ref.	68.688	-39.055	39.994		88.901	0.027	-36.696
Cobra wt.	0.612	-0.322	-1.333	***1.502***	-0.172	1.248	-0.015
Cobra+30 lb.	1.019	-0.573	-2.315	***2.594***	-0.255	2.275	-0.030

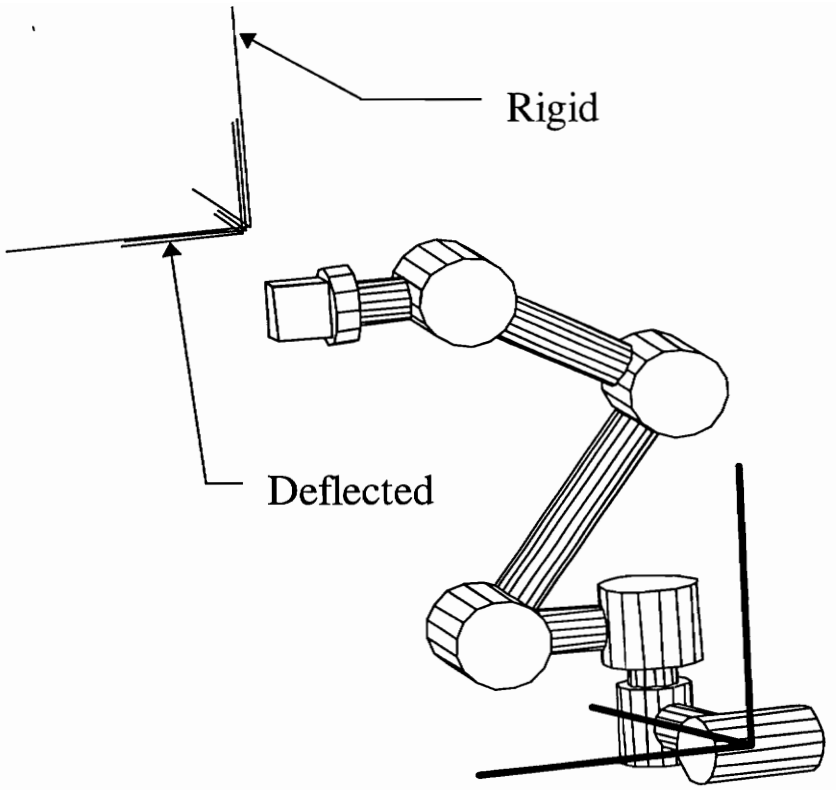

Figure 5.8 Cobra and Mezzanine for Example 2

For the configuration shown in Fig. 5.8 the modeled deflection of Cobra is given in Table 5.10. Again, the position and orientation componenents of the tool transform are given as a reference, and the changes are given for each loading scenario.

Table 5.10 Deflection Data for Example 2

Example 2 Fig. 5.7	Position Change (inches)				Orientation Change (degrees)		
	X	Y	Z	Mag.	γ	β	α
T_{Tool}^{Base} Ref.	23.829	30.582	46.937		89.606	2.158	73.302
Cobra wt.	0.185	0.181	-0.226	***0.343***	0.153	0.369	0.006
Cobra+30 lb.	0.312	0.441	-0.506	***0.740***	0.211	0.933	0.008

The primary task of the Cobra manipulator is positioning tools at the center of a steam generator tube. The predicted deflections shown in these examples are relatively large when compared to the 0.09 inch positioning accuracy necessary to locate steam generator tubes.

6. Conclusions

This chapter will briefly summarize the significant contributions of this research. It will also present specific recommendations for further improvement of Cobra's accuracy. Finally, general recommendations and future applications will be discussed.

6.1 Contributions of This Research

This thesis has presented several significant contributions to the field of robotics. First, a general approach for modeling the deflection of any serial manipulator was developed. The second major contribution of this work is the stiffness parameter characterization method. The use of numerical optimization to calculate manipulator stiffness parameters is a powerful innovation. This method is more accurate than analytically based approximations and does not require disassembling the manipulator to measure the component stiffnesses. The third contribution of this work is the controller implementation strategy. This strategy makes possible the real-time compensation of static deflections using only modest computer power with general computer code, written in ANSI-C. Finally, this theory was successfully applied to a nuclear service manipulator that suffered from unacceptable positioning errors due to deflections. The model was able to compensate for the majority of the deflection errors in real-time, thus dramatically increasing the accuracy and overall efficiency of the manipulator.

6.2 Specific Recommendations for the Cobra Manipulator

To further improve the positioning accuracy of the Cobra manipulator, the algorithm used to perform the calibration calculations should be investigated. Possibly, more than three calibration points could be incorporated using an optimization scheme. This would base the calibration on better sample of the overall workspace. In addition, a more precise method for locating the calibration points would increase the accuracy of the calibration. Computer vision techniques could be implemented to inspect the steam generator tube-sheet and obtain calibration data without actually touching the tube-sheet. This would assure that the arm was not under any loading aside from gravity at the time the calibration data was taken.

In addition, the values of the stiffness parameters used the characterize the system should be improved. This could be done by taking more precise deflection data for the stiffness parameter optimization. A direct measurement methods such as computer vision systems or spark-gap acoustic systems could be used to measure the end-effector deflection. With more accurate measurement methods, and possibly a higher number of data points, the optimizations for both the arm and the mezzanine stiffness parameters could be improved. If enough data were taken, it would also be possible to optimize for the arm parameters and the mezzanine parameters together. This would allow for collecting data with the arm attached to the mezzanine. Also, given enough test data, it would be possible to incorporate more torsional stiffness and beam element pairs into the model. Should this approach decrease the optimization residuals, better overall performance would be expected.

6.3 General Recommendations and Future Applications

Successful implementation of deflection compensation allows for accurate operation of low-speed flexible manipulators. The designer may choose to build a light-weight manipulator without sacrificing positioning accuracy. Many common industrial manipulators are heavy relative to their maximum payloads in order to maintain stiffness and therefore accuracy using only rigid kinematics. It is realized, however, that high-speed flexible manipulators may encounter problems associated with dynamics that are not addressed in this work.

This deflection modeling approach should be applied during the initial design phases of manipulator development. The stiffness of the members could be determined using analytical methods as an approximation. The designer can use the deflection model to determine the flexure of the proposed manipulator throughout its workspace. A sensitivity analysis could be wrapped around the deflection model to determine the links that should be stiffened to increase performance.

The iterative approach that is used to solve the flexible inverse kinematics of the manipulator could be replaced with a more sophisticated method. There are methods that employ a 16^{th} degree univariate polynomial in order to solve the inverse kinematics of a general six degree of freedom manipulator (Cordle 1993). This approach could be adapted to solve the flexible inverse kinematics. The polynomial method has many advantages over the iterative approach described in Chapter 4. All of the solutions are obtained, allowing obstacle avoidance schemes to select optimum manipulator configurations. For the implementations described in this work, the iterative methods were adequate, though the polynomial methods are more elegant.

Optimization methods were used to determine the stiffness parameters of the manipulator. This approach could be expanded to include the standard kinematic parameters of the manipulator as well. The determination of the as-built kinematic parameters of manipulators has been suggested (Mooring and Tang 1984), but never performed in conjunction with stiffness parameters. By including the flexibility coefficients of the manipulator with the standard kinematic parameters, the entire set may be determined using the data collection and optimization schemes presented here. A limited set of kinematic parameters were determined in the optimization of the mezzanine stiffness model described in Chapter 5.

7. References

Beer, F. P., and Johnston, E. R., 1981, *Mechanics of Materials*, McGraw-Hill, New York.

Bodur, I. N., and Derby, S. J., 1988, "Analysis and Modeling of the Positioning Inaccuracy of Industrial Manipulators in Off-line Programming Part 1: Due to Static Forces and Joint Clearances," *Proceedings of the 20th Biennial Mechanisms Conference*, Kissimmee, FL, DE-Vol. 15, No. 3 (Trends and Developments in Mechanisms, Machines, and Robotics), pp. 383-391.

Chang, L., and Hamilton, J. F., 1991, "The Kinematics of Robotic Manipulators With Flexible Links Using and Equivalent Rigid Link System (ERLS) Model," *Journal of Dynamic Systems, Measurement and Control*, Vol. 113, pp. 48-53.

Cheney, W., and Kincaid, D., 1985, *Numerical Mathematics and Computing*, Brooks/Cole, Monterey.

Cibiak, N., and Lipkin, H., 1994a, "Asymmetric Cartesian Stiffness for the Modelling of Compliant Robotic Systems," *Proceedings of the 23rd Biennial Mechanisms Conference*, Minneapolis, MN, DE-Vol. 72 (Robotics: Kinematics, Dynamics and Controls), pp. 197-204.

Cibiak, N., and Lipkin, H., 1994b, "Centers of Stiffness, Compliance, and Elasticity in the Modeling of Robotic Systems," *Proceedings of the 23rd Biennial Mechanisms Conference*, Minneapolis, MN, DE-Vol. 72 (Robotics: Kinematics, Dynamics and Controls), pp. 185-194.

Cordle, W. H., 1993, *Numerical Inverse Kinematics for a Six-Degree-of-Freedom Manipulator*, Masters Thesis, Virginia Polytechnic Inst., and State U., Blacksburg.

Craig, J. J., 1989, *Introduction to Robotics Mechanisms and Control*, Addison Wesley, New York.

Denavit J., and Hartenberg, R. S., 1955, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Journal of Applied Mechanisms*, Vol. 22, pp. 215-221.

Derby, S., 1983, "The Deflection and Compensation of General Purpose Robot Arms," *Mechanism and Machine Theory*, Vol. 18, No. 6, pp. 445-450.

Goldenberg, A. A., and Lawrence, D. L., 1985, "A Generalized Solution to the Inverse Kinematics of Robotic Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 107, pp. 103-106.

Juvinall, R. C., and Marshek, K. M., 1983, *Fundamentals of Machine Component Design*, John Wiley & Sons, Inc., New York.

Kerninghan, B. W., and Ritchie, D. M., 1988, *The C Programming Language*, Prentice Hall, New Jersey.

Lee, J. D., and Wang, B., 1988, "Optimal Control of a Flexible Robot Arm," *Computers and Structures*, Vol. 29, No. 3, pp. 459-467.

Lin, C. Y., Chew, M., Everett, L. J., and Juang, J., 1992, "Identification for Manipulators with Joint Compliances and Link Deflections," *Proceedings of the 23 Biennial Mechanisms Conference*, Minneapolis, MN, DE-Vol. 72 (Robotics: Kinematics, Dynamics and Controls), pp. 169-176

Lin, P. P., Chiang, H., and Cui, X. X., 1990, "An Improved Method for On-Line Calculation and Compensation of the Static Deflection at a Robot End-Effector," *Journal of Robotic Systems*, Vol. 8, No. 2, pp. 267-288.

Lipkin, H., and Patterson, T., 1992a, "Geometric Properties of Modelled Robot Elasticity: Part 1 - Decomposition," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 179-185.

Lipkin, H., and Patterson, T., 1992b, "Geometric Properties of Modelled Robot Elasticity: Part 2 - Center of Elasticity," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 187-193.

Mabie, H., and Reinholtz, C. F., 1987, *Mechanisms and Dynamics of Machinery, Fourth Edition*, John Wiley & Sons, New York.

Mooring, B. W., 1983, "The Effect of Joint Axis Misalignment on Robot Positioning Accuracy," *Proceedings of the ASME Conference on Computers in Engineering*, Chicago, IL, pp. 151-155.

Mooring, B. W., and Tang, G. R., 1984, "An Improved Method for Identifying the Kinematic Parameters in a Six-Axis Robot," *Proceedings of the ASME Computers in Engineering Conference*, Las Vegas, NV, pp. 79-84.

Naganathan, G., and Soni, A. H., 1987, "Coupling Effects of Kinematics and Flexibility in Manipulators," *International Journal of Robotics Research*, Vol. 6, No. 1, pp. 75-84.

Nissley, L., 1983, "Understanding Positioning Errors in Your Robotic Arc Welding System," *Welding Journal*, Vol. 62, Nov., pp. 30-37.

Patterson, T., and Lipkin, H., 1990a, "Structure of Robot Compliance," *Proceedings of the 21st Biennial Mechanisms Conference*, Chicago, IL, DE-Vol-26 (Cams, Gears, Robot, and Mechanism Design), pp. 315-322.

Patterson, T., and Lipkin, H., 1990b, "A Classification of Robot Compliance," *Proceedings of the 21st Biennial Mechanisms Conference*, Chicago, IL, DE-Vol-26 (Cams, Gears, Robot, and Mechanism Design), pp. 307-314.

Rao, S. S., 1979, *Optimization: Theory and Applications*, John Wiley & Sons, New York.

Reinholtz, C. F., 1983, *Optimization of Spatial Mechanisms*, PhD Dissertation, University of Florida.

Salerno, R. J., 1993, *Positional Control Strategies for a Modular, Long-Reach, Truss-Type Manipulator*, PhD Dissertation, Virginia Polytechnic Inst., and State U., Blacksburg.

Serna, M. A., and Bayo, E., 1990, "Trajectory Planning for Flexible Manipulators," *IEEE International Conference on Robotics and Automation*, Vol 2, pp. 910-915.

Shigley, J. E., and Mitchell, L. D., 1993, *Mechanical Engineering Design*, McGraw-Hill, Inc., New York.

Shooter, S. B., Reinholtz, C. F., and Dhande, S. G., 1992, "On the Kinematic Design of Manipulators for Limited Access Workspaces (LAWS)," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 485-491.

Shooter, S. B., and Reinholtz, C. F., 1992, "Extrinsic Calibration of Portable Manipulators," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 501-506.

Sun, C. H., Griffis, M. W., and Duffy, J., 1994, "Methodology for Comparing Stiffness Mappings Using Geometric Invariants," *Proceedings of the 23rd Biennial*

Mechanisms Conference, Minneapolis, MN, DE-Vol. 72 (Robotics: Kinematics, Dynamics and Controls), pp. 177-183.

Thomopoulos, S., and Tam, R., 1990, "An Iterative Solution to the Inverse Kinematics of Robotic Manipulators," *Mechanism and Machine Theory*, Vol. 26, No. 4, pp. 359-373.

Tidwell, P. H., Glass, S. W., Hildebrand, J. J., Reinholtz, C. F., and Shooter, S. B., 1991, "Cobra - Design and Development of a Manipulator for Steam Generator Maintenance," *Proceedings of the 2nd National Applied Mechanisms and Robotics Conference*, Vol. 1, pp. IVB1-1 to IVB1-10.

Xi, F., and Fenton, R. G., 1991, "A Quasi-Static Motion Planner for Flexible Manipulators Using the Algebra of Rotations," *Proceedings of the 1991 IEEE International Conference On Robotics and Automation*, Sacramento, CA, Vol. 3, pp. 2363-2368.

Xi, F., and Fenton, R. G., 1992, "Inverse Kinematics Analysis Using the Algebra of Rotations for Flexible Link Manipulators," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 209-216.

Xi, F., and Fenton, R. G., 1994, "On Inverse Statics and Dynamics Problems of Flexible Manipulators," *Proceedings of the 23rd Biennial Mechanisms Conference*, Minneapolis, MN, DE-Vol. 72 (Robotics: Kinematics, Dynamics and Controls), pp. 169-176.

Yang, Z., and Sadler, J., 1992, "Finite Element Analysis of Revolute Manipulators with Link and Joint Compliance by Joint-Beam Elements," *Proceedings of the 22nd Biennial Mechanisms Conference*, Scottsdale, AZ, DE-Vol. 45, (Robotics, Spatial Mechanisms, and Mechanical Systems), pp. 619-625.

Appendix A: Software Architecture

The deflection model was coded in C on a PC platform. It was then ported to a workstation to be used in the actual manipulator control software. To make the implementation more straightforward, the model was developed with a degree of open-architecture. All of the intermediate calculation results are available in the results structure, and may be viewed by the calling module.

There are three basic data structures which must be defined globally in the calling module. These hold the miscellaneous parameters, *Misc*, the link parameters, *Link*, and the results of the deflection analysis, *Results*. These three structures are passed to the main deflection modeling routine, *Bend-Bot*.

The stiffness parameter optimization module is a standalone program which reads touch-point data from a file. The routine uses the same modeling code, *Bent-Bot*, used by the controller, to evaluate the objective function in the optimization.

The contents of the data structures is presented below in the form of the deflection model header file. This code segment was included to give the reader the sense of the general approach used in the deflection model code. The actual deflection modeling code is under 700 lines including comments.

```
/*************************************************************
BEND-BOT                        Deflection modeling software
Written by:  Joe Calkins        MAIN HEADER FILE
Language ::  ANSI-C
*************************************************************/
typedef struct /* general variables required for deflection model. */
{
   int links;      /* set to the number of links on the arm */
   BEND_VECTOR
      gravity,     /* vector in the base frame: direction of gravity*/
      tool_tip,    /* vector to tool-tip in last link frame */
```

```
      dR,          /* tool rotations due to deflection */
      dP;          /* tool translations due to deflection */
   double
      rigid_tool[4][4], /* holds the tool transform without deflection*/
      Tolerance;   /* tolerance for force vector iterations */
}   BEND_MISC;

typedef struct /* Link definition structure. Must be populated. */
{
   double
      alpha,    /* D&H link parameters... alpha i-1 */
      a,        /*                        a i-1 */
      d,        /*                        d i */
      theta,    /*                        theta i */
      Ixx,      /* moment of intertia (in^4) about the link's X axis */
      Iyy,      /* moment of intertia (in^4) about the link's Y axis */
      Izz,      /* moment of intertia (in^4) about the link's Z axis */
      E,        /* modulus of elasticity */
      w,        /* link weight (lbs/in) */
      joint_wt;   /* weight of joint at the END of the link (lbs) */
    BEND_VECTOR   /* a structure containing data elements x, y, z */
      K;   /* joint housing deflection factor about the x,y,z axis*/
}   BEND_LINK;

typedef struct /* holds the calculated variables for each link */
{
   BEND_VECTOR
      F,           /* force vector at the end of the link */
      w,           /* uniform load vector */
      r,           /* vector along the X axis of the link */
      M,           /* moment at the end of the link */
      Fl,          /* force in local (link) coords */
      wl,          /* vector in direction of link-weight */
      Ml,          /* moment in local (link) coords */
      Ml_joint,    /* moment in joint coordinates of previous link */
      Rl,          /* total rotations about each axis */
      Rl_joint,    /* the rotation about each joint axis */
      dl;          /* positional deflection in local (link) coords */
   double
      len,         /* length of vector r */
      t[4][4],  /* individual link transform */
      tw[4][4];  /* transform from current link to the world */
}   BEND_RESULTS;

/* function prototypes */
extern void bend_load(BEND_MISC  *,BEND_LINK *);
extern void bend_bot(BEND_MISC *,BEND_LINK *, BEND_RESULTS *);
```

In order to implement the deflection model, instances of the data structures must be declared. Next, the *bend_load* function is called to load the deflection model data file into the structures. Once the joint values in the *Link* structure are set, the *bend-bot* function is called to analyze the deflection. The *results* structure may be inspected to determine the tool transform with and without deflection, and intermediate values of interest.

Vita

Joseph M. Calkins was born in Baltimore, Maryland on December 16, 1970. During grade-school, his family moved to Lancaster, Pennsylvania where Joe lived until he graduated from high school in 1989. During high school, Joe worked vacations and weekends as a computer programmer for Kalas Mfg. Inc., located in Denver, Pennsylvania.

Joe chose to pursue Mechanical Engineering at Virginia Polytechnic Institute and State University. This decision was based not only on the schools excellent reputation in engineering, but on the beautiful location of the Virginia Tech campus. Joe spent much of his undergraduate career hiking, biking, and camping throughout the beautiful Blue-Ridge mountains. During semester breaks, Joe worked as a project engineer at Kalas Mfg. Virginia Tech is also where Joe met his wife Kara, whom he married in January of 1994.

Upon graduation in May of 1993, Joe took a summer position as a robot control software developer at B&W Nuclear Technologies, located in Lynchburg, Virginia. It was there Joe became interested in studying the deflection of robots. This provided an excellent transition into his graduate studies at Virginia Tech.

Joe plans to continue his career as a graduate student at Virginia Tech, pursuing the degree of Doctor of Philosophy in Mechanical Engineering.