Chapter 5: Automatic Plant Description

This chapter details the systematic design and implementation of an automatic physical plant description system for robots. The plant description is essentially a dynamic model of the robot. In its current form, this plant description system calculates performance criteria values, the joint forces or torques, and the deflection of the robot's end-effector. The system is automatic because it produces its numerical outputs given its parametric inputs with no further user intervention. There are many uses for an automatic plant description system and many reasons for preferring a systematic design process.

Designing a robot requires evaluating many different design alternatives. The automatic plant description system can be an important part of the evaluation process. Choosing a monolithic robot for a given task also requires evaluating robots. Configuring a modular robot for a given task requires evaluating a large number of different configuration alternatives. The automatic plant description system will certainly play an important role in modular robot configuration. The automatic plant description system is very much tied to generalized inverse kinematics. A generalized inverse kinematics algorithm must incorporate performance criteria. Joint torques or forces and end-effector deflections are important performance criteria. The generalized inverse kinematics algorithm generates these criteria values using the plant description system. Many of the benefits associated with a general inverse kinematics algorithm will be lost if the plant description is not general and automatic. An automatic plant description

system could also be a part of a commercial software package. Finally, incorporating a plant description improves the performance of robot controllers.

This chapter details a systematic design process that loosely follows the method of Pahl and Beitz (1988). This design method leaves a structured record of the goals, constraints, and specifications that influenced the design. The design begins by identifying the system's inputs and outputs. These inputs and outputs combine with the intended uses of the automatic plant description system to determine a set of specifications. The design then divides the automatic plant description system into three subsystems to satisfy the specifications. A database subsystem stores and retrieves information associated with different robot modules. The model formulation subsystem accepts a description of the robot as its input. This subsystem combines the robot's description with information from the database to formulate a model of the robot. The final subsystem actually calculates the performance criteria values, joint torques or forces, and end-effector deflections. The model formulation subsystem communicates with the calculation subsystem via datafiles. This allows automatic downloading of the robot's model to any computer using standard file transfer protocols. The calculation subsystem may reside on a high-speed processor for real-time control applications.

5.1 Inputs and Outputs

Good systematic design demands the complete understanding and clear specification of all requirements at the outset of the design process. This facilitates an understanding of the task at hand and discourages unnecessary efforts that do not lead towards a solution. It is important that the specifications

do not imply any specific implementation as this might bias the outcome of the design and adversely affect the results. A clear specification sheet also provides a yardstick by which to judge the final design. An understanding of the automatic plant description system's inputs and outputs facilitates the development of a specification sheet. Fig. 5.1 shows these inputs and outputs graphically. This figure is a preliminary description of the system intended to facilitate the development of a set of specifications. As the design progresses, it will detail the structure of the data.

The robot's description is a fundamental input to the automatic plant description system. This description is also somewhat problematic since the robot may be modular or it may be monolithic. In the case of a modular robot, the automatic plant description system requires a list of the modules, the order in which the modules are arranged, and the geometry of the connections between the modules. For a monolithic robot, the system requires the robot's geometry as well as other physical data, such as mass and compliance parameter distribution. The description of the robot is specific to each particular robot, but independent of the task. The path description and the externally applied loads, however, are task-dependent inputs.

The path description is communicated to the automatic plant description system at the joint level. Describing the path at the joint level maintains a separation between the automatic plant description system and the inverse kinematics method. This separation allows for the system's use in applications not requiring generalized inverse kinematics. The separation also prevents the system from being tied to any one inverse kinematics method.

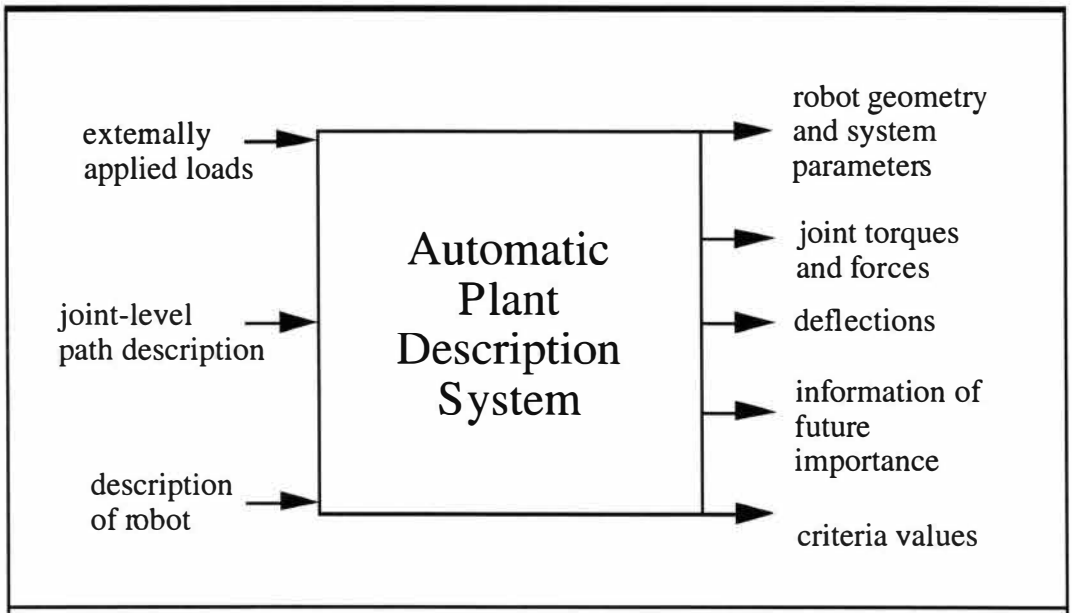

Figure 5.1 The gross inputs and outputs of the automatic plant description system.

The externally applied loads are specific to the task and the environment. Gravity, for instance, is an externally applied load that is of little concern to a robot working in outer-space. Other externally applied loads include the various forces and torques at the robot's end-effector necessary to perform a given task.

Calculating the torques and forces at the joints of the robot is a primary function of the automatic plant description system. The torques and forces represent the load on the robot's actuators.

End-effector deflection refers to deviations between the desired and actual end-effector displacement due to deformations of the robot's structural and drive components. Both applied and inertial loads cause end-effector deflections. Gravitational forces, when applicable, typically cause significant deflections at the

end-effector. Vibrations, particularly if the arm is excited at a resonant frequency, may also cause significant deflections.

Performance criteria values are another output of the automatic plant description system. The performance criteria are mathematical measures with a physical interpretation. The literature review describes a number of criteria for evaluating the robot's performance.

The robot's geometry and other physical parameters are outputs of the system available for downloading onto a high-speed processor for real-time control. This output of the automatic plant description system must contain all the physical plant information required by the real-time controller.

The final output is information of future importance. The exact nature of this output is unknown, but undoubtedly in the future there will be requirements for the automatic plant description system that cannot be foreseen at this time. These requirements might concern thermal effects, reliability, or power demands. By including a provision for future requirements, the current capabilities of the system will not become embedded constraints and limit the usefulness of the system in the future.

5.2 SPECIFICATIONS

At this point in the design process, it is possible to develop a set of specifications for the system. Table 5.1 shows the specification sheet. The right column contains a brief description of the specification and the left column indicates whether the specification is a demand or a wish. The design is not satisfactory if it does not meet a demand. Wishes are incorporated if possible, but

the design will still fulfill its intended uses if that specification is not met. The intended applications of the automatic plant description system determined these specifications. This discussion will therefore focus on each specification as it affects the intended applications.

Table 5.1 A specification sheet for the automatic plant description system

Demand or Wish	Specifications
D	formulate robot geometry and system parameters
D	calculate joint torques
D	calculate deflections at the end-effector
D	expandable to include new robot modules
D	calculate performance criteria values
D	portable among many different computing platforms
D	automatic (requires no intervention to produce outputs given inputs)
D	expandable to include new modeling and analysis capabilities
D	general with respect to the robot's geometry
W	use only University of Texas Robotics Research Group software

Generality with respect to the robot's geometry is important for several of the intended applications. Robot design involves a large number of design parameters that may vary independently. The definition of a general robot structure is that all the physical parameters may vary. A general automatic plant description system would also be useful when choosing from among the different available monolithic robots or configuring modular robots given a particular application. Finally, generality is important in an automatic plant description system embedded in a commercial software package. Essentially, a more general formulation has more market value.

Portable software is machine independent. Basically, the automatic plant description system should be compatible with as many computers as possible. Because computer technology is advancing at a rapid pace, tying the automatic plant description system to a specific computer platform would virtually guarantee obsolescence within just a few years. Portable software also has much more market value. Installed computer hardware varies from industry to industry and even a single customer might have several different computer platforms within the same factory. Software that runs on all of their computers and allows them to upgrade their computers without changing software will be very attractive to these customers. Maintaining software compatibility on different computer platforms is also more expensive and difficult than writing portable software from the outset.

Restricting this effort to software developed by the University of Texas Robotics Research Group's software is not a rigid requirement, but rather falls under the "wish" category. Doing so, however, ensures maintaining maximum flexibility as far as modifying and updating the software. This approach also acknowledges the Group's large inventory of available software and continuing work in software development.

Expandability to include new robot modules and new modeling techniques will allow the automatic plant description system to remain useful as the state of the art in modular robotics advances. New robot modules are being developed at the University of Texas and elsewhere. The automatic plant description system's structure should allow for the addition of these new modules. Work is also continuing in the area of generalized compliance modeling and dynamic

modeling. The system's design should allow for the inclusion of these new modeling techniques as well.

Several of the intended applications of the automatic plant description system may involve calculating forces and torques at the joints. During the robot's design, calculated forces and torques are useful when sizing the actuators. The robot's actuators must be powerful enough to achieve the payload and motion specifications. Distributing the torques or forces such that the actuators nearest the base of the robot support as much of the load as possible may be important for a given design. The required joint torques or forces are extremely important when evaluating a monolithic robot's ability to perform a given task. The most fundamental concern is that the robot must have actuators with a maximum capability greater than what is required at all points along the path. For a modular robot, the forces or torques at the joints are but one of the criteria used to assemble modules in response to a task specification. Finally, an inverse kinematics strategy for redundant robots can make use of calculated forces or torques to choose joint-level paths that optimize their distribution among the actuators.

Deflection calculation during the design of a robot measures the effects various design options have on compliance. A robot designed to withstand end-effector disturbances without significant deflections is essential for precision operation. Applied and inertial loads cause deflections that ultimately limit a robot's precision. With compliance modeling and deflection calculation, a monolithic robot capable of performing the desired task within the required precision may be intelligently chosen. When configuring modular robots,

compliance modeling and deflection calculation can form part of the module selection process. Properly choosing and arranging the modules will minimize the total deflection or, if the task requires, allow deflections in certain directions but not others. An inverse kinematics strategy for redundant robots can make use of deflection calculation to choose a joint-level path that minimizes undesirable deflections.

The performance criteria developed by the University of Texas Robotics Research Group (described in the literature review) are very high-level mathematical formulations for evaluating a robot as it performs a task. Robot design involves a large number of choices. Even a simple robot with six degrees of freedom has eighteen geometric, forty-two mass, thirty-six deformation, and eighteen actuator parameters, for a total of 114 parameters (Tesar and Butler, 1989a). The different combinations of these design parameters represent a huge number of options available to the designer. This large number of design parameters recommends the use of performance criteria to give a "big" picture of how the different design alternatives affect the capabilities of the robot. Choosing a monolithic robot for a given application might involve performance criteria calculation. This is not, however, a very good application of the high-level performance criteria. Most industrial robots are of a similar design. These performance criteria are more appropriate for choosing from among a larger number of options. On the other hand, configuring modular robots involves choosing from among a very large, yet finite, number of options. The performance criteria are ideally suited to this approach to design. Performance criteria are a fundamental part of generalized inverse kinematics. A generalized

inverse kinematics strategy must make intelligent choices to provide enhanced performance. Performance criteria form the fundamental basis for these decisions.

The specification that the plant description system be automatic requires that the system produce the desired operational outputs given the parametric inputs without further intervention by the user. Most of the intended uses of the automatic plant description system involve evaluating a large number of robots. An automatic system is much more convenient than one requiring user intervention for each different set of robot parameters. Also (assuming the algorithms are correct) an automatic system is much less likely to make errors. Finally, a Bachelors-level engineer, or even a trained salesperson, could operate an automatic plant description system. This gives an automatic system much more market value than one requiring intervention by a robotics expert.

5.3 SUBSYSTEM IDENTIFICATION

Several of the specifications call for specific calculations. These calculations are based on a model of the robotic system. Formulation of the model, in turn, requires physical parameters specific to each robot. Thus it can be seen that the automatic plant description system may be divided into three main subsystems: calculation, model formulation, and information storage (database).

The calculation subsystem is responsible for generating the numerical values for the joint torques or forces, the deflections, and the performance criteria. This subsystem could reside on the same computer as the other subsystems. This would likely be the case in a simulation environment. For real-time control, the

calculation subsystem would reside on a dedicated high-speed processor. Computer code for performing these calculations already exists and is available for use in the automatic plant description system. Unfortunately, the existing code does not have a unified format for the robot model. The structure for the model input has taken two distinct formats. The earliest and most general code accepts the Denavit and Hartenberg parameters as a description of the robot's geometry. The newer code uses a modular description of the robot's geometry (essentially the specific modules and the geometry of the connections). In both cases, the mass and compliance distributions are referenced to the local frame. There are positive and negative qualities associated with both modeling methods and the automatic plant description system should accommodate both approaches.

Table 5.2 The inputs and outputs of the three subsystems which comprise the automatic plant description system.

Subsystem	**Input**	**Output**
calculation	system model in either a DH or a modular format task information, including the joint displacements, joint speeds, joint accelerations, and end-effector loads.	deflections at the end-effector joint torques performance criteria values
formulation	description of the robot in either a DH or modular format pertinent data from the database	system controlling model in either a DH or a modular format
database	robot or module of interest parameters entered into the database	requested operational parameters

The model formulation subsystem assembles the physical parameters of the robot into a format which can be understood by the calculation subsystem. The design of the model formulation subsystem is complicated by the two different model formats required by the calculation subsystem. If the robot of interest is modular and the calculation software requires the Denavit and Hartenberg parameters, a processing step will be necessary to extract these parameters from the modular robot description. Several researchers have proposed procedures for performing this extraction process. Unfortunately, their procedures do not allow for a general modular geometry. A process which is general with respect to the modular robot is developed as part of this design process.

The final subsystem performs the data storage and retrieval. This is a database where the physical parameters of each module, such as the geometry or mass, are stored. Database management is a common use for computers and there are many developed techniques available for this database application. This database is somewhat unusual because it will be updated infrequently – only when new modules are added – but accessed often. The information that needs to be associated with each module is determined by the needs of the calculation subsystem. The raw data in the database is assembled by the model formulation subsystem and then communicated to the calculation subsystem.

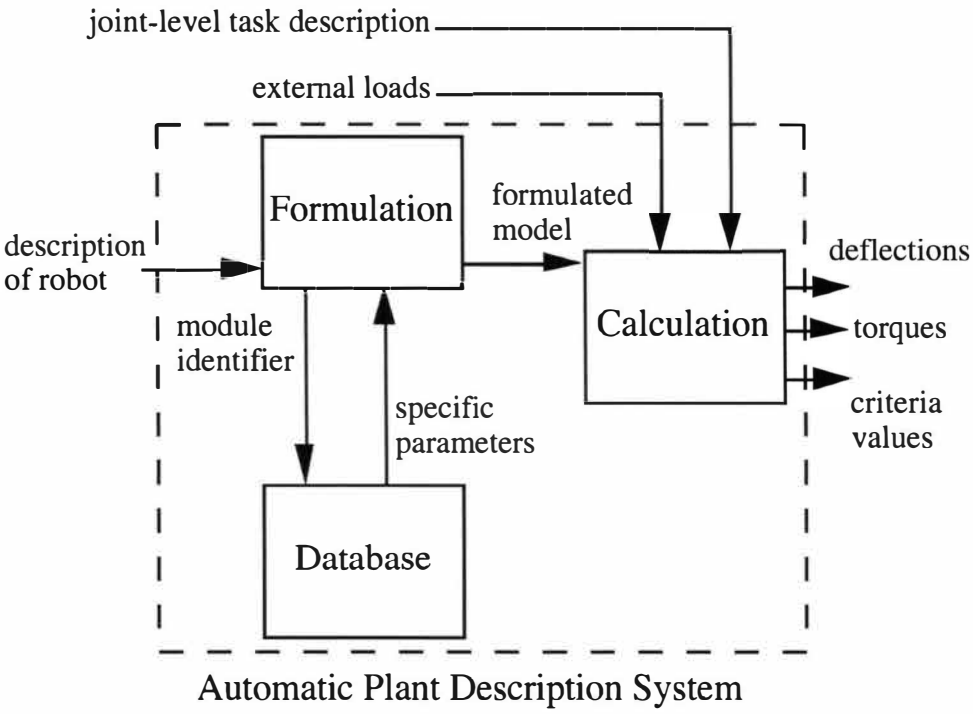

Figure 5.2 A schematic showing the information exchange between the three major subsystems of the automatic plant description system.

Figure 5.2 shows the interconnections between the subsystems. Each of the subsystems will now be discussed in detail and several options for fulfilling their functions will be identified and developed. As the subsystems are being discussed, the inputs and outputs of each subsystem will be explicitly formulated. By carefully defining the structure of the interface between the subsystems, functional independence of each subsystem is maintained. Thus it will be possible to upgrade an individual subsystem without requiring changes to the rest of the automatic plant description system. For instance, a calculation subsystem

developed for a specialized real-time controller will be able to access to the model formulation subsystem.

5.3.1 Calculation Subsystem

The calculation subsystem is where several of the outputs of the automatic plant description system are actually generated. The input to this system is a parametric description of the robot and the task that the robot is to perform. The outputs are the joint torques or forces, the end-effector deflections, and performance criteria values. This subsystem could reside on a general-purpose computer for simulation applications, or it could reside on a specialized computer for real-time control applications. The important issue is the identification and description of the subsystem's inputs. The proper inputs can then be communicated to the calculation subsystem wherever it resides. There are two general types of inputs. The robot-specific inputs describe physical parameters of the robot and the task-specific inputs describe the task.

The calculation subsystem essentially performs kinematic and dynamic analysis of robotic systems; a research area in which the University of Texas Robotics Research Group has made a significant investment. This research has built upon the kinematic influence coefficients first developed for the analysis of planar Assur groups by Benedict and Tesar (1978). Various researchers have extended the use of kinematic influence coefficients to include the analysis of: torque requirements for general serial manipulators (Thomas and Tesar, 1982), end-effector deflections of general serial manipulators (Fresonke et al., 1988), torque requirements for modular manipulators (Kang et al., 1992), and

performance criteria for general serial manipulators (Van Doren and Tesar, 1992). This effort has resulted in a large stock of computer software that will satisfy the requirements for the calculation subsystem.

Table 5.3 Some functions and attributes of robot analysis software available for use in the calculation subsystem.

Software	**Functions and Attributes**
NDOF serial analysis software	entirely general
	Denavit and Hartenberg-style inputs
	calculates joint torques
	calculates deflections referenced to the end-effector
Module-Based Manipulator Analysis Package	calculates joint torques
	modular-style inputs
	seven different types of modules are addressed
Performance Criteria Code	calculates the values for twenty-nine different criteria
	general with respect to the robot's geometry
	Denavit and Hartenberg-style inputs

Work in the area of kinematic and dynamic analysis of robots is continuing at the University of Texas Robotics Research Group and elsewhere. The deflection and compliance analysis for modular manipulators is but one example of this continuing work. It is also likely that kinematic and dynamic analysis code for real-time control algorithms will be specialized for the controller hardware. Given that the calculation subsystem will continue to evolve, the input format of the analysis code that already exists will be identified and standards suggested so that future code can maintain compatibility with the existing software.

The foundation of the computational effort in kinematic and dynamic analysis is the NDOF serial modeling code (Wander and Tesar, 1987). This code was originally written for execution on a specialized array processor but was later rewritten in the portable "C" programming language. This code is entirely general in the sense that it will calculate the joint torques or forces for a serial robot with any number of degrees of freedom. The robot-specific inputs are the geometry and the mass distribution of the robotic system. The geometry is expressed in terms of the Denavit and Hartenberg parameters. The mass distribution is given by the individual masses, the location of the mass-centers and the inertia tensors; all referenced to the local coordinate system. The robot-specific inputs are communicated via datafiles. The task-specific inputs are the applied end-effector loads and the joint states. The applied loads are referenced to the base frame coordinates. The joint states are simply the displacements, speeds, and the changes in speeds with respect to time.

The NDOF compliance code calculates the deformations expressed at the end-effector due to applied and inertial loads acting upon structural and drive flexibilities (Hernandez and Tesar, 1989). Local deformations are transferred to the end-effector via kinematic influence coefficients. This code was also originally written for execution on an array processor and later rewritten in its portable form in the "C" programming language. The compliance code is general in the sense that it applies to any serial robot with any number of degrees of freedom. The robot specific inputs to this code are the compliance distribution, the mass distribution and the robot geometry. The robot geometry is described using Denavit and Hartenberg parameters. The mass distribution is given by the

individual mass values, the location of the mass centers, and the inertia tensors; all referenced to the local coordinate frames. The compliance is described in six general directions, three translations and three rotations, also referenced to the local frames. The robot specific parameters are communicated via datafiles. The task-specific inputs are the applied loads and the individual joint states. The applied loads are referenced to the base frame and the joint states are the displacements, speeds, and changes in speeds with respect to time.

The module-based manipulator analysis package written by Kang (1992) is significantly different from the NDOF code in the way that the robot-specific inputs are formulated. The modular analysis package is tailored towards modular robots, rather than robots described by Denavit and Hartenberg parameters. The module-based manipulator analysis package will analyze any modular robot composed of nine basic modules: constant link, prismatic joint, base joint, elbow joint, 2 DOF knuckle, 3 DOF wrist/shoulder, 1 DOF parallel elbow, 2 DOF five-bar, and 2 DOF seven bar. The dynamic analysis is based on the recursive Newton-Euler formulation. The robot specific inputs are: the robot modules, the connection geometry between the modules, the ordering of the modules, and the mass distribution within the modules. The robots modules are simply specified by their names. The connection geometry is specified via *XZX* Euler angles. The ordering of the modules is serial; meaning they are simply listed from first to last. The mass distribution is specified by the individual mass values, the location of the mass centers and the inertia tensors referenced to local frames. For the serial modules the mass distribution is specified just as in the NDOF code. A convention is developed to specify the geometry and mass distribution for the

parallel modules. For the larger modules, such as the five-bar and seven-bar, this represents a significant amount of information. There are over thirty parameters which describe these module's geometry and mass distribution. The module-based manipulator analysis package is written in the "C" programming language and the robot-specific inputs are communicated via datafiles. The task-specific inputs are the joint states and are also communicated via datafiles. The module-based manipulator analysis package, as it is currently written, does include end-effector loads.

A module-based manipulator compliance analysis effort is currently under way. Though still in development, several aspects of this system are known. The computer code is written in "C". The analysis is based upon a known set of modules connected at rigid interfaces and the information pertinent to each module is expressed in local coordinates. This module-based manipulator compliance analysis package is mentioned to emphasize that continuing work is expected in the area of module-based kinematic and dynamic analysis.

A generalized inverse kinematics implementation calculates performance criteria values using the automatic plant description system. The University of Texas Robotics Research Group has developed software that will calculate criteria values for at least twenty-nine different performance criteria (Van Doren and Tesar, 1992). This code is unique to the University of Texas Robotics Research Group and there are no alternatives available from outside sources. The code is written in the "C" programming language. The literature review in this dissertation contains a listing of the performance criteria values which may be calculated using this code. The code is general in the sense that it can calculate

the criteria values for any serial robot with any number of degrees of freedom. The robot-specific inputs are the mass distribution, the compliance distribution and the robot geometry. The mass distribution is given by the individual mass values, the location of the mass centers, and the inertia tensors; all referenced to the local coordinate frames. The compliance is described in six general directions, three translations and three rotations, also referenced to the local coordinate frames. The robot geometry is described using Denavit and Hartenberg parameters. The robot-specific parameters are communicated via datafiles. The task-specific parameters are the joint displacements, speeds, and changes in speeds with respect to time. The task-specific parameters are communicated to the code either through datafiles, or as arguments to a function call which also identifies the specific criterion of interest.

The main difference in all the previously mentioned software packages is the format for describing the robot's geometry. The NDOF code and the performance criteria code use Denavit and Hartenberg parameters, while the module-based manipulator analysis package and the (in-development) module-based compliance modeling package use a sequence of known modules. In both approaches, the mass and compliance distribution is described relative to the local frames. There are positive and negative qualities associated with both methods of specifying the robot's geometry.

Denavit and Hartenberg parameters are suited for monolithic robots. The Denavit and Hartenberg parameters are commonly used by most researchers in the robotics field and their use facilitates the importation of outside technology. Denavit and Hartenberg parameters do not, however, provide enough information

for some types of analysis, especially those associated with modular robots. For instance, the Denavit and Hartenberg parameters do not give information about where one module ends the next begins. It is not possible to extract which modules comprise the robot simply from the Denavit and Hartenberg parameters. This is because different modules could be arranged so as to have the same Denavit and Hartenberg parameters.

The modular approach to kinematic and dynamic analysis is much simpler with regard to the robot-specific inputs. A graphical user interface, such as that in the robo_cad software package, makes it quite easy to assemble the modules and generate a file describing the modules and their connectivity (Hooper and Tesar, 1990). Assigning Denavit and Hartenberg parameters, on the other hand, requires some skill in the art of robotics. The modular approach also allows parallel modules to form hybrid chains with both serial and parallel structures. The standard Denavit and Hartenberg parameters do not include hybrid chains. Finally, a modular description is rather clumsy for monolithic robots, though it is possible using very general modules.

The commonalties among the various software packages suggest standards so that future modeling and analysis efforts can maintain compatibility. The most important standard is communicating the robot-specific inputs with a datafile. All of the existing University of Texas software accepts the robot-specific inputs in this fashion and files provide a convenient interface that allows software modules to maintain functional independence. File transfer protocols exist for easy automatic downloading to high-speed processors for real-time control applications. Another standard is that the task specific inputs should include, as a

minimum: the joint displacements, speeds, and changes in speeds with respect to time. Dynamic analysis code should also include the loads at the end-effector referenced to the base frame coordinates. All of the kinematic and dynamic analysis software, with the exception of the module-based manipulator analysis package, adheres to this standard. The inability to include end-effector loads is a shortcoming of the module-based manipulator analysis package. A final standard is that the mass and compliance distribution should be referenced to the local frames. For purely serial structures the protocol of Wander, Hernandez and Tesar should be used, and for in-chain parallel structures the protocol of Kang and Freeman should be used. No clear standard as to whether the geometry of the robot should be represented by Denavit and Hartenberg parameters or as a sequence of modules has evolved. There are benefits to both approaches. The development of a method for extracting Denavit and Hartenberg parameters from a modular sequence will, to a large extent, alleviate difficulties that might arise from failing to set a standard in this area.

5.3.2 Database Subsystem

A database keeps records on a permanent storage medium that is accessed by various users at different times. The automatic plant description system's database keeps a record of the available modules and associates with each module a variety of information. The modeling subsystem accesses the database and gathers the information required by the calculation subsystem. Other processes may access the database as well. For instance, the automatic configuration of a modular robot in response to task requirements requires information about the

available modules. The database should be designed to facilitate the addition of new information.

As a first step, the design of this subsystem identifies the information that will be placed into the database. This information includes an identifier, such as a name, for each of the currently available modules. The information associated with each of the modules is determined primarily by the requirements of the calculation subsystem. All calculations require the geometry of the module. For torque and force calculations, the mass, location of the center of mass, and the inertia tensor must be specified relative to the local frames of each module. The calculation of deflections at the end-effector requires the compliance distribution as well. This information satisfies the current requirements of the calculation subsystem. There is also other information that may be important in the future. This information includes: gear ratios, speed limits, torque limits, thermal time constants, frequency response, and cost.

Even if there were hundreds of available modules and thousands of parameters associated with each module, this would not be a complex database application. For instance, the number of available modules will undoubtedly be much less than the number of names in the phone book. Because the amount of information in the database is relatively small, the data can reside in directly-addressable computer memory. This eliminates the need for repeatedly swapping data to and from the permanent storage device. Also, the information in the database will be updated infrequently. This prevents conflicts between processes accessing and updating the database. The resolution of these conflicts is a principle function of most database software. Because the demands are limited,

there are many options available for satisfying the database requirements. These options include: a simple database of lists, a hierarchical database, a relational database, and an object-oriented database.

A simple list is one example of a database. A list can be ordered, such as the residential pages of the phone book. A list can be indexed, such as the chapters in a book, or a list can be both ordered and indexed, such as the yellow pages in the phone book. The list-type of database is sufficient for the automatic plant description system application. The list associates parameters with each module. The classification of the parameters, such as geometric or inertial, indexes the list. The data for each module can simply be typed into a file. Since conflicts between processes updating and reading the database are not expected, the code for reading the database will be very straightforward. Finally, modules can be added to the database by creating new files and entering the appropriate data. New information can be associated with each module by simply adding the new information at the bottom of the list. It is important that new information be added to the end of the list, rather than in the middle, so as not to cause errors in previously written software.

A hierarchical database is slightly more complicated than a simple database of lists. The hierarchical database is particularly effective when the data can be arranged in a tree structure. Information can be manipulated at the highest level while the associated data beneath that level is handled automatically. This type of database could be used for the automatic plant description system application, though it is not particularly well-suited. This is because there is no hierarchy among the modules.

The most common database is the relational database. The relational database assigns references, dependencies, and commonalties to the information in the database. This type of database is useful when changes in one piece of information imply changes to information stored elsewhere. This database technique could be applied to a complete modular robot system. For instance, a given set of modules might all use the same actuator and therefore have the same torque capabilities. In a relational database, a dependency between the actuator and the torque capabilities could be established. Substituting a different actuator in the specification for the modules would automatically enact a change in the record for torque capabilities. Another example is that changes in material specifications might also relate to changes in stiffness. This level of sophistication is not, as yet, found in modular robots. Because the automatic plant description system is designed with the database as an independent subsystem, a relational database could be incorporated when warranted by sufficient advances in robotics technology.

Object-oriented databases are becoming increasingly popular. These databases adhere to the object-oriented programming paradigm (Zdonik and Maier, 1990). The data are considered objects and are categorized according to the class in which they belong. In object-oriented programming jargon, each datum object is called an instance of its class. Each class is defined by certain properties that apply to all of the data objects in the class. Combining old classes creates new classes. In the jargon, a new class created from old classes is said to inherit their properties. The object-oriented paradigm is useful for creating very flexible software modules that can be used in a variety of applications. The

flexibility is derived from a very careful definition of the classes and the inheritance structure. An object-oriented database is certainly capable of meeting the requirements of the automatic plant description system. At this time, however, the benefits do not warrant the additional complexity.

Table 5.4 Some factors associated with application of the four different database types as part of the automatic plant description system.

Database	**Positives**	**Negatives**
list	software to create and manage is straightforward matches the modest demands of the automatic plant description system	changes in one part of the database don't automatically cause changes in other parts of the database
hierarchical	effective when data can be arranged in a tree structure data is manipulated at a high-level	database architecture doesn't match the modular robotics architecture
relational	changes in one part of the database automatically cause changes in other parts of the database may have future application in the automatic plant description system	software for implementation is complex
object-oriented	expansion of the database is easy	software for implementation is complex "overkill" for the automatic plant description system

Each of these database techniques is able to satisfy the database requirements of the automatic plant description system. This is because the automatic plant description system is a very simple database application with a relatively small amount of infrequently-updated information. Of these database

techniques, the relational database and the indexed list are most applicable. A relational database is somewhat complex and implementation will require either purchasing software or a significant development effort. An indexed list is a database technique that provides a good match with the automatic plant description system's database requirements. Data for the indexed list can be entered directly with any editor. Portable software to read the data can be written with minimal effort.

5.3.3 Model Formulation Subsystem

The model formulation subsystem is the heart of the automatic plant description system. This subsystem's input is a description of the robot. This description may be of either a modular or monolithic robot. Using the robot's description and information in the database, this subsystem assembles the parameters and formulates a model of the robot. The model of the robot is generated and transferred to the calculation subsystem for analysis. The model must be expressed in either a modular or Denavit and Hartenberg format depending upon the analysis software. There are four combinations of input and output formats: a modular robot with module-based analysis, a modular robot with Denavit and Hartenberg-based analysis, a monolithic robot with module-based analysis, and a monolithic robot with Denavit and Hartenberg-based analysis.

The input of the modeling subsystem is a description of either a monolithic or a modular robot. Denavit and Hartenberg parameters are the standard method of describing the geometry of a monolithic robot. The defacto

standard for specifying a modular robot configuration (within the University of Texas Robotics Research Group) is the robo_cad file (Hooper and Tesar, 1990). Robo_cad files specify the robot's modules, their sequence, and their connection geometry. These files are created interactively by choosing and scaling modules through the robo_cad graphical user interface. Robo_cad files have also been written by optimization routines configuring a limited number of modules in response to simple task priorities (Ambrose and Tesar, 1992). In any event, a description of the robot geometry is available via Denavit and Hartenberg parameters for monolithic robots or as robo_cad files for modular robots.

Table 5.5 The combinations of input and output formats possible in the the modeling subsystem.

Input	**Output**
Modular	Modular
Modular	Denavit and Hartenberg
Denavit and Hartenberg	Modular
Denavit and Hartenberg	Denavit and Hartenberg

The format for the output of the modeling subsystem is determined by the kinematic and dynamic analysis software. Again, two basic methods have been identified, the module-based approach and the Denavit and Hartenberg-based approach. For both cases, the available kinematic and dynamic analysis software packages require that the model be communicated via a datafile. Communication of the robot-specific inputs via datafiles has also been suggested as a standard during the course of this design. Communication via datafiles facilitates

functional independence and modularity of the software and is ideally suited for automatic downloading of system parameters via standard file transfer protocols.

A modular robot description is inherently compatible with a module-based analysis package. The input is a robo_cad file and the output file is written in the module-based manipulator analysis package format. The modeling subsystem reads the robo_cad file to identify the sequence of modules and the geometry of the connections. It then accesses the database to acquire the parameters associated with each module. Finally, each module's identifier along with the corresponding geometry and mass distribution is written sequentially into a datafile. The datafile forms the input to the module-based manipulator analysis package.

The most challenging combination of input and output for the modeling subsystem is the modular robot description with the Denavit and Hartenberg-based analysis software. Addressing this combination requires extracting the Denavit and Hartenberg parameters from the modular robot description. There are several motivations for developing this capability. It will allow the NDOF torque and compliance software to be used with a modular robot. The performance criteria software is also based on a Denavit and Hartenberg description of the robot. Finally, the bulk of manipulator theory uses a Denavit and Hartenberg-based description of the robot's geometry. Other research institutions have developed procedures for extracting Denavit and Hartenberg parameters from a modular robot description. The most notable of them are from Carnegie Mellon University by Kelmar and Khosla (1990), and the University of Toronto by Benhabib (1989). Though the procedures seem to work for simple

module geometries and constrained connection methods, the development is lacking in generality. A brief discussion of these extraction procedures will be followed by the derivation of a general procedure. This derivation is a specific development of this dissertation.

Carnegie Mellon University's procedure for extracting Denavit and Hartenberg parameters from a modular robot geometry relies on very simple module geometries with constrained connection methods. The links are designed as a single linear translation along the Z-axis. The rotational joints only allow rotation about the Z axis and translations along the Y and Z axes. A twist about the Z axis specifies the connections between modules.

Benhabib's procedure for extracting the Denavit and Hartenberg parameters is inefficient because it generates two sequential Denavit and Hartenberg transformations, rather than one, for each joint axis. Also, the extraction method is only valid for modular robots with strictly Denavit and Hartenberg geometries. That is, the modular geometry must be expressed as $\text{Screw}_x(a_{i-1}, \alpha_{i-1})$ then $\text{Screw}_z(d_i, \Phi_i)$.

The methodology of both of these extraction procedures is essentially the same. By restricting the module and the connection geometry, the geometric problem of extracting parameters from general homogeneous transformations is reduced to simply adding parameters together as modules are added to the robot chain. Though this procedure is valid for simple cases, there are reasons for preferring a more general approach. For instance, a metrology effort may find that a joint axis for a specific module is skewed by some amount from the nominal design. The above-mentioned extraction procedures are invalid in this

case. Also, the previous extraction procedures constrain the design of the modules as well as the configuration possibilities for the modular robot. This is contrary to the goal of designing the modules based on engineering principles and configuring them to best perform a given task. Clearly, a general extraction procedure is superior.

In the general case, each link module may be represented by a single homogeneous transform from its input to its output. Each joint may be represented by one homogeneous transform from its input to its axis of motion and by another homogeneous transform from its axis of motion to its output. Concatenating all the transforms between the two joint axes generates a single homogeneous transform from one axis to the next. Without loss of generality, two local reference frames, {i-1} and {i}, are defined with their Z axes coinciding with the joint axes. The Denavit and Hartenberg parameters to be extracted from the homogeneous transform between these two successive joint axes, $\hat{Z}_{i-1}$ and $\hat{Z}_i$, are defined as follows

α_{i-1} = The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured in a right-hand sense about their mutual perpendicular, $\hat{x}_{i-1}$.

a_{i-1} = The distance along $\hat{x}_{i-1}$ between $\hat{Z}_{i-1}$ and $\hat{Z}_i$.

d_i = The distance along $\hat{Z}_i$ between $\hat{x}_{i-1}$ and $\hat{x}_i$.

Φ_i = The angle between $\hat{x}_{i-1}$ and $\hat{x}_i$ measured in a right-hand sense about $\hat{Z}_i$.

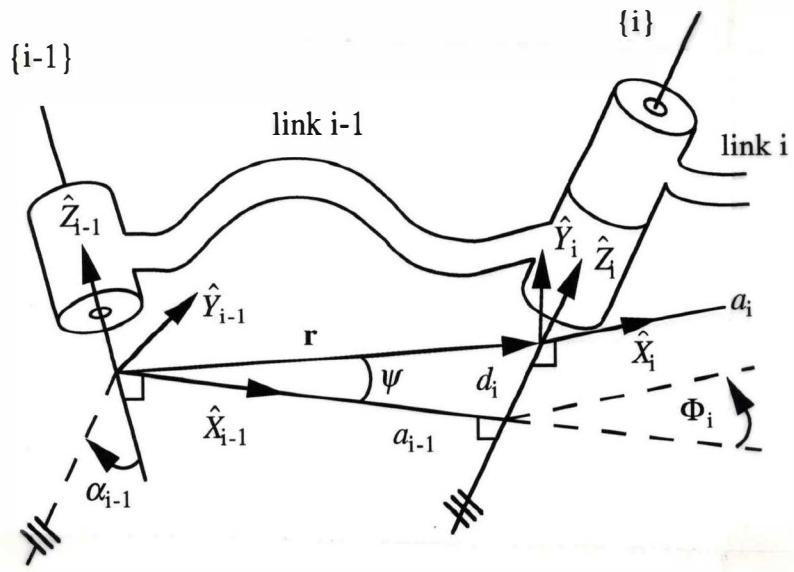

Figure 5.3 Then Denavit and Hartenberg parameters relating frame {i-1} to frame {i}. The vector **r** and the angle ψ are defined for use in the extraction procedure.

Figure 5.3 illustrates these quantities. The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ is α_{i-1}, which can be found using the inner product.

$$\cos\alpha_{i-1} = \hat{Z}_{i-1} \bullet \hat{Z}_i \tag{5.1}$$

$$\alpha_{i-1} = \cos^{-1}\left(\hat{Z}_{i-1} \bullet \hat{Z}_i\right) \tag{5.2}$$

Though the inner product gives the magnitude of α_{i-1}, there is an ambiguity concerning the direction. The ambiguity is because there are two solutions for Θ in the equation

$$\cos\Theta = b \tag{5.3}$$

which are

$$\Theta = \cos^{-1}(b) \tag{5.4}$$

and

$$\Theta = -\cos^{-1}(b) \tag{5.5}$$

The Atan2 function will resolve the ambiguity. Lipkin and Duffy (1985) suggest

$$\cos\alpha_{i-1} = \hat{Z}_{i-1} \bullet \hat{Z}_i \tag{5.6}$$

and

$$\sin\alpha_{i-1} = \hat{Z}_{i-1} \times \hat{Z}_i \bullet \hat{x}_{i-1} \tag{5.7}$$

then

$$\alpha_{i-1} = \mathrm{A}\tan 2(\sin\alpha_{i-1}, \cos\alpha_{i-1}) \tag{5.8}$$

There are some problems with the Atan2 function. For instance, the solution to $\mathrm{A}\tan 2(0,0)$ is ambiguous and may vary among computer platforms. Assuming a sign and then performing the rotation by the assumed α_{i-1} as a check will also resolve the ambiguity in Equation (5.2). If the $\hat{Z}_{i-1}$ and $\hat{Z}_i$ axes are in the same direction, the assumption was correct. If they are in opposite directions, the assumed α_{i-1} is multiplied by -1 to produce the correct direction. Given that a modern computer can check this assumption in a small fraction of a second, using the assume and check method may be preferable. In any case, the remainder of this derivation assumes that similar ambiguities in direction are resolved either by the $\mathrm{A}\tan 2$ function or by checking.

Figure 5.3 defines a vector **r** to find the two lengths, a_{i-1} and d_i. The vector **r** appears explicitly as the translational portion of the homogeneous transform between the frames {i-1} and {i}. The angle, ψ, defines the angle

between $\hat{x}_{i-1}$ and $\mathbf{r}$. The unit vector, $\hat{x}_{i-1}$, defines the line perpendicular to both $\hat{Z}_{i-1}$ and $\hat{Z}_i$.

$$\hat{x}_{i-1} = \hat{Z}_{i-1} \times \hat{Z}_i \tag{5.9}$$

The angle ψ is then found as

$$\cos\psi = \frac{\mathbf{r} \bullet \hat{x}_{i-1}}{|\mathbf{r}|} \tag{5.10}$$

$$\psi = \cos^{-1}\left(\frac{\mathbf{r} \bullet \hat{x}_{i-1}}{|\mathbf{r}|}\right) \tag{5.11}$$

The lengths a_{i-1} and d_i are

$$a_{i-1} = |\mathbf{r}|\cos\psi \tag{5.12}$$

and

$$d_i = |\mathbf{r}|\sin\psi \tag{5.13}$$

The joint angle, Φ_i, (the angle between $\hat{x}_{i-1}$ and $\hat{x}_i$) is the final Denavit and Hartenberg parameter to be extracted. The angle Φ_i is found as

$$\cos\Phi_i = \hat{x}_{i-1} \bullet \hat{x}_i \tag{5.14}$$

$$\Phi_i = \cos^{-1}(\hat{x}_{i-1} \bullet \hat{x}_i) \tag{5.15}$$

This procedure for extracting the four Denavit and Hartenberg parameters from a homogeneous transform matrix is general and allows the NDOF analysis software and the performance criteria software to be used with a modular robot description.

Another combination of robot description and analysis software is the monolithic robot and the Denavit and Hartenberg-based analysis software. With this combination, the modeling subsystem simply queries the user for the pertinent information. Once the modeling subsystem has obtained the necessary data, the subsystem simply writes the information into a datafile. The datafile can

be saved in the database for future use. Eventually the database will include a library of monolithic robots.

The final combination is the monolithic robot and the module-based analysis software. This is, however, an unreasonable combination. The monolithic robot does not have interfaces that allow it to be decomposed into a set of joint and link modules. The automatic plant description system must use the Denavit and Hartenberg-based analysis software to simulate a monolithic robot.

5.4 IMPLEMENTATION

A working version of the automatic plant description system described in this chapter was written in the "C" programming language and implemented on a Unix computing platform, though it will run on most modern computers. As with any design, implementation of the automatic plant description system required choosing from among several options. Options were identified for each of the three main subsystems: calculation, model formulation, and information storage (database). This discussion focuses on the options that were chosen and the reasoning behind the choices.

The main result in the implementation of the modeling subsystem was the procedure for extracting the Denavit and Hartenberg parameters from a homogenous transformation matrix. The translation of this procedure into computer code was not difficult. The extraction process for a robot with seven degrees of freedom runs on a Silicon Graphics Indigo R4000 in less than one second. Other than this extraction procedure, the modeling subsystem simply must read and write datafiles.

Four types of databases were identified as options for use in the automatic plant description system (list, hierarchical, relational, and object-oriented). Of these options, the list structure was chosen for implementation. There were several reasons for making the choice. The simple list structure matches the very modest demands of the automatic plant description. There will be a relatively small amount of data which can easily be handled by a list structure. Also, because the database will be updated infrequently, managing conflicts between processes simultaneously attempting to read from and write to the database is not difficult. The list database can easily be updated to incorporate new information associated with the existing modules and also allows the addition of new modules. Finally, the software to manage the list database could be written entirely in-house and with only a modest effort.

Four different software packages were identified for use in the analysis subsystem of the automatic plant description system. These software packages are: the NDOF joint force and torque calculation software, the NDOF end-effector deflection calculation software, the performance criteria calculation software, and the module-based manipulator analysis package. The first three of these software packages use a Denavit and Hartenberg-based description of the robot geometry. The module-based manipulator analysis package generates the robot geometry from the sequence of known modules. All of these software packages were addressed in the implementation. The crucial factor is that the modeling subsystem communicates the robot-specific inputs to the calculation subsystem via datafiles. Communication of the robot-specific inputs via datafiles facilitates functional independence and modularity of the software. Data files are also

ideally suited for automatic downloading of system parameters to specialized and real-time processors via standard file transfer protocols.

5.5 EXAMPLE

This section presents an example of using the Automatic Plant Description System for a modular robot with seven degrees of freedom. This example shows the steps to the user for running the appropriate software and entering the necessary parameters. There are essentially three distinct steps:

1. Assemble the robot graphically using the **robo_cad** software,
2. Extract the D. H. parameters using the **extract** program,
3. Run the dynamic modeling software (**dyn** in this example).

This example also includes representative plots of the output data and run times for an Analogic AP500 array processor and an Indigo personal workstation.

Robo_cad is a copyrighted, interactive assembly program for modular and reconfigurable robots. Through a graphical user interface, the program allows the operator to assemble a modular robot on the computer screen and then immediately animate the model thus providing an impression of the simulated robot's dexterity. The program is based on a set of palettes, each containing a variety of one, two, and three degree of freedom joint modules and generic links. By selecting from among these modules and supplying the appropriate parameters, the operator assembles the modular robot.

After running the robo_cad program (simply type "robo_cad"), choose a base for the robot. This is accomplished by selecting "add base" from the "build" menu. The program will display the palette of base modules. This example uses

the slanted mounting pedestal (Figure 5.4). Choose the color for the module and then click in the "O.K." box, and the slanted mounting pedestal appears on the computer screen.

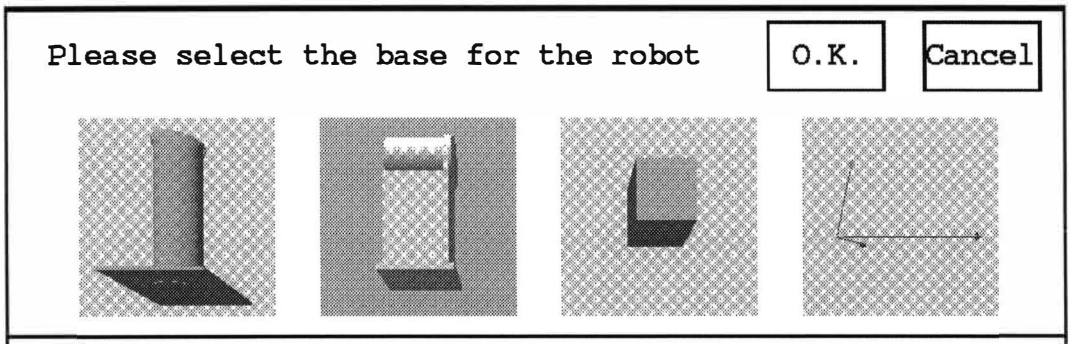

Figure 5.4 A palette of base modules from the robo_cad software.

The next step is to choose a joint for the robot. Select "add joint" from the build menu and the palette of joint modules appears on the screen. Choose a pivot module and click in the "O.K." box. A dialogue box requiring several input parameters will appear on the screen (Figure 5.5). Specify a length for the pivot joint, a twist angle, and the color. Then click in the "O.K." box and the pivot module appears on the screen connected to the base module.

Continue selecting and adding modules until the simulated robot is complete. There is essentially no limit as to the number of modules that may be added to the model, though this example gives timing results for a robot with seven degrees of freedom. After the robot is complete, save the model by choosing "save" from the "file" menu. The program will ask for a name for the model. Any name is fine. Simply type the name into the dialogue box.

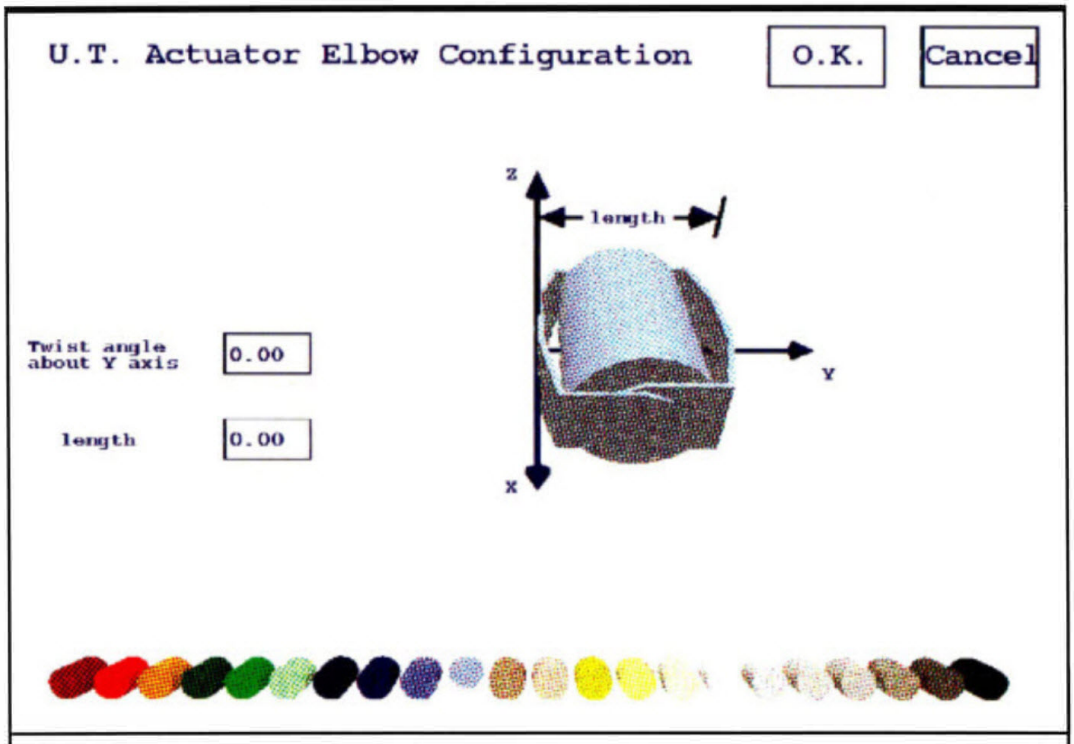

Figure 5.5 A dialogue box from the robo_cad software that shows the input parameters for a pivot joint module.

The next step in using the Automatic Plant Description System is to extract the Denavit and Hartenberg parameters using the **extract** program. This program is essentially the extraction procedure given in section 5.3.3 translated into computer code. It also includes a simple protocol to access a database for the physical parameters associated with each module. To run this program, type "extract". The program will ask for the name of the robo_cad file and for the name of a file for the D. H. and physical parameters. After entering these file names, the program runs without any further intervention. For a robot with seven degrees of freedom, the extraction procedure takes less than one second on an Indigo computer.

Once the Denavit and Hartenberg parameters are available, the next step is to run the dynamic modeling software. Running this software requires one component that is not part of the Automatic Plant Description System, and that is a path for the robot. The path for the robot may come from an off-line programming environment, or it may be part of a real-time robot control system. For the off-line operation, the **dyn** program will generate the joint torques and end-effector deflections. Figure 5.6 shows the joint torques and the end-effector deflections calculated by **dyn** for a representative path. On an Indigo R4000 personal workstation, the joint torques and end-effector deflections together take 2.4 milliseconds to calculate for a robot with seven degrees of freedom. For real-time operation, the Analogic AP500 is currently the choice for a processor. This array processor will calculate the joint torques and end-effector deflections in 3.7 milliseconds for a robot with seven degrees of freedom.

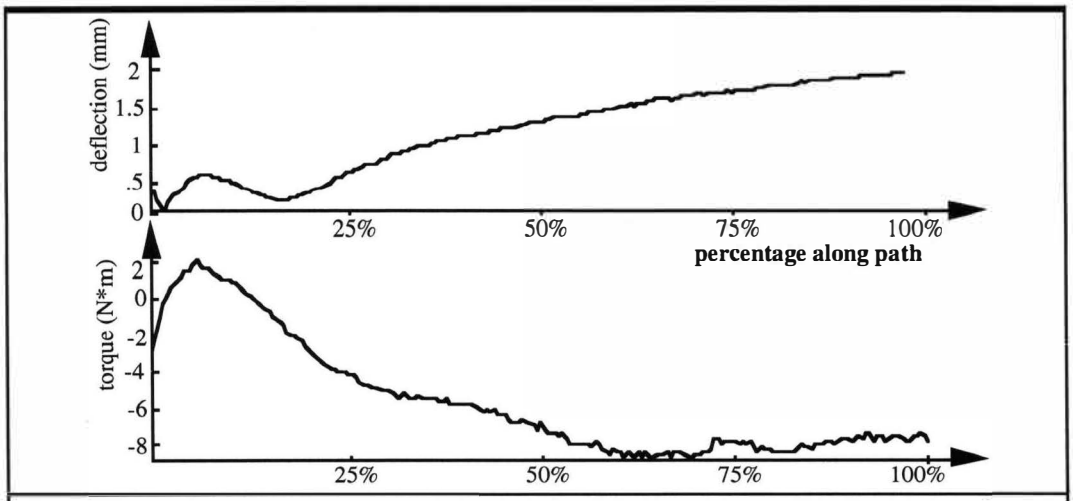

Figure 5.6 Magnitude of end-effector deflection and torque demand on the first joint of a 7 DOF robot following a representative path.