Int. J. Appl. Engng Ed. Vol. 7, No. 3, pp. 197–206, 1991
Printed in Great Britain.

0742–0269/91 $3.00+0.00

Computer Assisted Teaching on Robot Pose Error Modeling*

N. VIRA†
L. THIGPEN‡

Department of Mechanical Engineering, Howard University, Washington, DC 20059, U.S.A.

This paper presents a concept for teaching students on robot pose error modeling and compensation methods with the aid of character strings manipulation program developed on a DEC VAX-750 minicomputer under VMS operating system. A software package is designed which combines generality, user interaction and user-friendliness with the systematic usage of symbolic computation and artificial intelligence techniques. It utilizes MACSYMA, a LISP-base symbolic algebra language to generate automatically closed-form expressions representing end-effector's pose errors in the world coordinate system for N degree-of-freedom industrial robots where N being the number of joints. The goal of such a package is to aid faculty and students in the robotics course by removing burdensome tasks of mathematical manipulations. Several commercially available robotic configurations are tested to verify the program accuracy. The worst case takes 174.8 seconds to generate three-dimensional error equations in world coordinates for the PUMA 600 series robot, which is insignificant compared to the time required for accurate manual derivation.

INTRODUCTION

ROBOTIC manipulators have been used in many automated manufacturing processes like painting, welding, seam tracking, and pick/place type of operations. Recently, they have been employed in more difficult situations such as assembly of parts and insertion of precision components into circuit boards. In many of these processes the exact position and orientation (pose) of the robot's wrist with respect to the part being processed is crucial to the success of the operation and the quality of the product. Although the desired pose is communicated to the robot controller through teaching with a teach pendant or programming, the actual pose of the wrist will often be somewhat different from that desired. The pose inaccuracy is due to many errors present in the robotic system ranging from control point of view through structural consideration [1].

There are two approaches to improve robots' positioning accuracy. One would be to manufacture robots with greater precision accuracy; however, this is quite costly because the production cost increases rapidly with requirements of high accuracy standards. Oftentimes, this is unattainable because current machines and manufacturing processes have not yet evolved to the stage necessary to satisfy the required tolerance limits. The second approach to solving this problem would be to consider software corrections in which the systematic errors of a robot are first determined and then corrected in the robot controller. Note that the systematic errors can be represented by some sort of functional relationship between the robot's controller output readings and true wrist positions. This paper is devoted to issues of robot pose error modeling, error compensation through software correction, and a method of teaching such topics with the aid of symbolic computation.

Brief review on pose error modeling

In recent years, researchers have proposed numerous error calibration and compensation models for the improvement of robots' pose accuracy. Some of the modeling techniques are based on modifications of the well known Denavit-Hartenberg kinematic models (DH model), while others are based on alternative methods of kinematic representation (non-DH models). The majority of these models deal primarily with kinematics of the robot which address errors in geometrical link parameters. For example, models proposed by Wu [2, 3], Hayati [4], and Hsu and Everett [5] are based on DH formalism and are modified to handle parallel or near parallel robot joints. Other models proposed by Whitney [6], Stone [7], Driels and Pathre [8] and Chen [9] are based on coordinate transformations but do not follow the DH notations. Mooring [10] has proposed a model based on Rodrigues equation that does not utilize successive coordinate transformations. Vaishnav and Magrab [11] have employed the theory of skewed coordinate systems and formulated the pose error corrections by modifying Cartesian coordinate frames attending to each joint of the robot. Non-orthogonality of the joint axes and origin shift errors are included in the

* Paper accepted 14 September 1990.
† Naren Vira is Associate Professor.
‡Lewis Thigpen is Professor and Chairman.

modeling. A host of other positioning error models (DH and non-DH) have been developed and reported in the literature. A comprehensive literature review discussing these models can be found in Ziegert and Datseris [12]. A number of the contributions have not only introduced error model formulations, but have also implemented various robot calibration schemes. Although a wide variety of modeling approaches have been used, no single approach has yet been accepted as standard.

Robotic education

Presently, robotics education is in its early stages and educators have encountered several difficulties in designing suitable courses [13–16]. This is due in part to the interdisciplinary nature of robotics. That is, robotics encompasses three major disciplines: mechanical engineering, electrical engineering and computer science. In fact, many robotics related courses are included as required technical electives in mechanical, electrical and computer engineering undergraduate curricula. However, the emphasis placed on certain topics may be weighted according to the department through which the courses are being offered. As a result, each group of students has strengths not possessed by students in other disciplines. For example, mechanical students may have a good understanding of concepts in kinematics and dynamics, but may not feel as comfortable with control theory and electrical actuators as their counterparts in electrical engineering. Likewise, computer science students may have a better understanding of microprocessors and real-time computation than either mechanical and electrical engineering students. With such a heterogeneous group of students at different levels of education, it becomes difficult to design a course to satisfy the needs of all students without too much repetition of topics to one group of students.

Currently, two types of introductory course are being offered by several universities. One is technology-related; it stresses industrial robot applications and the operation of robotic accessories such as actuators, sensors, encoders, vision systems, microprocessors, and drive mechanisms. The second type covers the theoretical fundamentals of robotic kinematics, dynamics, and control. Based on these fundamentals, students are taught the design of the controller and its function as the 'robot brain'.

Advanced level graduate courses in robotics cover a wide spectra of topics that are general in nature and sometimes specialized for the particular research application. For example, kinematics, dynamics, and control topics can normally be found in courses given at various universities whereas topics such as robotic vision, end-tooling design, computer simulation and pose error modeling are uncommon since special expertise of the faculty is required to offer them. As we indicated earlier, robotics applications are growing and more accurate robots are needed in industry. Thus, the knowledge of pose error modeling and error compensation techniques become vital. To provide training to future graduates, we at Howard University have included such a topic in our advanced level robotics course. This paper is written to share our experience and demonstrate the feasibility of using symbolic computation as an aid in the teaching process.

Difficulties

We have experienced numerous difficulties in teaching by conventional techniques, the underlying concept on robot error modeling and compensation. First, there exists a variety of models and formulations of which are to be included in the course content. Inclusion of too many models would make ineffective presentation since the time devoted for the topic is limited whereas the coverage of too few models will provide students with insufficient background. Second, model derivation in the classroom is laborious since the equations are very lengthy. A single equation may fill several text book pages. It is obvious that manual derivation of the equations for a large number of models and robot configurations is a time-consuming and error-prone undertaking. Third, in order to communicate effectively with students, several exercises are required. By solving different problems students gain experience necessary to understand the theoretical concepts. Equations and solutions developed by students are applicable for different robot configurations. Hence, it becomes almost impossible (certainly time consuming) for the professor to check solution accuracy and grade each and every derivation. Furthermore, the problem intensifies if a large number of students are enrolled in the course. Last but not least, to discuss effectiveness of a model and to describe the influence of an error parameter on overall model validity, several models must be demonstrated and compared in the class room. In other words, to compare n models on r robots, a total of $n \times r$ model equations representing three dimensional configurations must be developed. This presents a major bottleneck in the process of model comparison. In addition, numerical results are required to make any quantitative sense of the relative importance of the model parameters and their effects on the model accuracy (sensitivity analysis).

This paper demonstrates an effective way to cover sufficient material while providing students with the appropriate fundamentals. Our approach uses a software package which eliminates the burdensome task of deriving equations and allows students to develop a thorough understanding of a broader range of subject matter. Such is the case with well known software packages like STRUDL for structural analysis, and SPICE for electrical circuit analysis; surely the concept can be extended to the analysis of robotic manipulator's errors.

In general, the theoretical analysis of robotic pose error modeling consists of mathematical operations requiring knowledge of matrix manipulations, trigonometry, calculus, differential geometry and solution of differential equations (if dynamic error modeling is considered). Students in introductory courses learn to apply these mathematical tools together with proper practical consideration. Therefore, symbolic computation seems to be the most logical candidate on which to base any computer-aided analysis software for use in robotics education. The interactive software package presented here was developed with this in mind, and the paper is devoted to a discussion of how to use the package to benefit such a course topic. Figure 1 depicts a conceptual framework representing union of robot error modeling and symbolic computation. Such union becomes a powerful tool useful for robotic education and enhancement of the state-of-the-art research. The symbolic manipulation software package named AREEM (Automatic Robot Error Equations Modeler) automatically generates geometrical error model equations applicable for robotic configurations with N degrees of freedom [17].

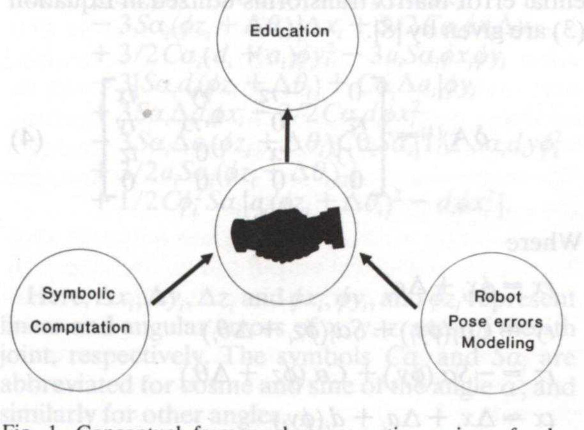

Fig. 1. Conceptual framework representing union of robot error modeling and symbolic computation.

BACKGROUND THEORY

An industrial robot is considered to be an open-loop manipulator comprising a series of links connected together by lower pair joints. Each link is assumed to have only one joint that is either revolute (rotational) or prismatic (sliding). A robot is moved to perform a specific task by driving various joints via its controller. Thus, the robot hand or wrist takes positions as required by the desired task. Due to the presence of errors, the hand positions are usually different from the desired ones. These differences in hand positions are the robot pose errors. Before one considers any scheme to model inaccuracy associated with the robot hand, its pose errors must be known *a priori*. The robot pose errors are determined using some sort of measurement device. A typical measurement configuration is depicted in Fig. 2 where the measurement device, robot controller, and a host computer are interfaced for on-line data acquisition and analysis. The measurement device shown is a contact type of system because it directly connects to the robot wrist. Detailed description of the measurement system and its usage procedures can be found from Ref. [18]. The measured robot pose errors are classified into two categories—systematic and random. The systematic errors are reproducible and can be represented by mathematical functional relationship. Errors of this kind can be compensated for by algorithms (software) when the functional relationship is known. Random errors are difficult to relate and compensate for. Statistical tools may be used provided the variance of their probability distribution is reduced. Most of the models described in the literature deal with systematic errors and so does our discussion.

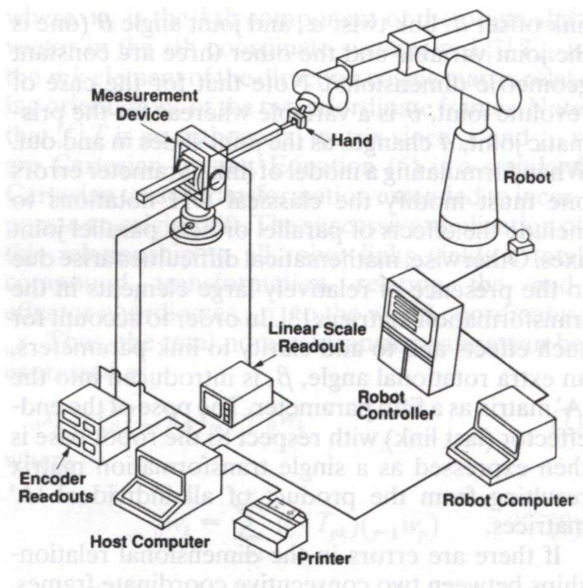

Fig. 2. A typical measurement system configuration for a robot pose error determination.

At present, the AREEM program incorporates three models of which two are based on DH formulation (five and ten error parameters) and one uses non-DH formulation. These models are commonly discussed by researchers. However, the selections in the AREEM program are still arbitrary. Additional models, if desired, can easily be included because AREEM is structured in a modular fashion. The robot pose error correction equations for each of the three models are presented below.

Five-parameter DH model. According to the DH notations, the relative translations and rotations between robotic links are represented by assigning a coordinate frame to each joint. The relationship between adjacent coordinate frames is then expressed by a 4×4 homogeneous transformation matrix called an 'A' matrix [19]. The 'A' matrix is a function of four linked parameters—link length a, link offset d, link twist α, and joint angle θ (one is the joint variable and the other three are constant geometric dimensions). Note that for the case of revolute joint, θ is a variable whereas for the prismatic joint, d changes as the joint slides in and out. When formulating a model of link parameter errors one must modify the classical DH notations to include the effects of parallel or near parallel joint axes. Otherwise, mathematical difficulties arise due to the presence of relatively large elements in the transformation matrices [3]. In order to account for such effects and to add clarity to link parameters, an extra rotational angle, β, is introduced into the 'A' matrix as a fifth parameter. The pose of the end-effector (last link) with respect to the robot base is then expressed as a single transformation matrix resulting from the product of all individual 'A' matrices.

If there are errors in the dimensional relationships between two consecutive coordinate frames, there will be differential change, $d\mathrm{A}_i$, between the two joint coordinates. The correct relationship between the two successive joint coordinates will then be equal to.

$$\mathrm{A}_i^c = \mathrm{A}_i + d\mathrm{A}_i \tag{1}$$

where A_i is the 'A' matrix of the ith joint and $d\mathrm{A}_i$ is the differential change in joint coordinates $i-1$ and i due to errors in the five kinematic parameters $(\Delta\theta, \Delta d, \Delta a, \Delta\alpha, \Delta\beta)$. The superscript c denotes the correct value. This differential change of the transformation matrix A is estimated by a Taylor series [20]. Considering the A_i^c matrix of Equation (1), the correct pose of the end-effector ${}^0T_N$ with respect to the world coordinate frame is

$$^0T_N^c = {}^0T_N + dT = \pi_{i=1}^{N} (\mathrm{A}_i + d\mathrm{A}_i). \tag{2}$$

The symbol π represents the product of terms considered and dT is the total differential change of the end-effector pose due to the geometrical errors. Simplification of Equation (2) yields

$$dT = {}^0T_N * \delta T = {}^0T_N * [\delta T^{(1)} + \delta T^{(2)}] \tag{3}$$

where

$$\delta T^{(1)} = \sum_{i=1}^{N} L_i^{(1)} = \sum_{i=1}^{N} ({}^iT_N)^{-1} * \delta\mathrm{A}_i^{(1)} * ({}^iT_N)$$

$$\delta T^{(2)} = \sum_{i=1}^{N} ({}^iT_N)^{-1} * \delta\mathrm{A}_i^{(2)} * ({}^iT_N) + \sum_{i=1}^{N-1} \sum_{j=i=1}^{N} L_i^{(1)} * L_j^{(1)}.$$

The accuracy of dT is up to second order since $d\mathrm{A}_i$ includes terms of second-order only. The $\delta\mathrm{A}_i^{(1)}$ and $\delta\mathrm{A}_i^{(2)}$ are, respectively, first-order and second-order differential error matrix transforms with respect to coordinate frame $i-1$. Note that $d\mathrm{A}_i$ is substituted as $\{d\mathrm{A}_i = \mathrm{A}_i * [\delta\mathrm{A}_i^{(1)} + \delta\mathrm{A}_i^{(2)}]\}$ [See Ref 20 for details]. Equation (3) relates the change in the A matrices to the resultant change in the end-effector pose (cumulative error in world coordinates). Also, $5N$ error parameters are at our disposal which are required to be estimated from measured values of dT. Once error parameters of Equation (3) are found from measured pose errors for a given robot configuration, it can be used to correct the positioning error of the robot, and thus can be incorporated in the robot controller. Equation (3) is written in a compact notation. When it is spelled out for a robot geometry, it can extend over several notebook pages.

Ten-parameter DH model. The second kinematic error compensation model included in the AREEM program is based on ten error parameters per robot joint axis. The model is a reconciliation of the perturbed four-parameter DH representation with a generalized six-parameter model developed and used in the analysis of coordinate measuring machines. The additional six parameters accounts for errors due to robot axes motion (i.e., one error along a prime axis of motion, two out-of-straightness errors, and three rotational errors (roll, pitch, and yaw) of a joint axis). For this case, the differential error matrix transforms utilized in Equation (3) are given by [8].

$$\delta\mathrm{A}_i^{(1)} = \begin{bmatrix} 0 & -rz & ry & tx \\ rz & 0 & -rx & ty \\ -ry & rx & 0 & tz \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Where

$$rx = \phi x_i + \Delta\alpha_i$$

$$ry = C\alpha_i(\phi y_i) + S\alpha_i(\phi z_i + \Delta\theta_i)$$

$$rz = -S\alpha_i(\phi y_i) + C\alpha_i(\phi z_i + \Delta\theta_i)$$

$$tx = \Delta x_i + \Delta a_i + d_i(\phi y_i)$$

$$ty = C\alpha_i(\Delta y_i) + S\alpha_i(\Delta z_i + \Delta d_i) - C\alpha_i(d_i\phi x_i) - S\alpha_i(a_i\phi y_i) + a_iC\alpha_i(\phi z_i + \Delta\theta_i)$$

$$tz = -S\alpha_i(\Delta y_i) + C\alpha_i(\Delta z_i + \Delta d_i) + S\alpha_i(d_i\phi x_i) - C\alpha_i(a_i\phi y_i) - a_iS\alpha_i(\phi z_i + \Delta\theta_i)$$

and

$$\delta\mathrm{A}_i^{(2)} = \begin{bmatrix} ux & vx & wx & ttx \\ uy & vw & wy & tty \\ -uz & vz & wz & ttz \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

where

$$ux = 3/2[\phi y_i^2 + (\phi z_i + \Delta\theta_i)^2] - 6\phi x_i\phi y_iC\theta_iS\theta_i$$

$$uy = 3(1 + 2C\theta_i^2)C\alpha_i\phi x_i\phi y_i + 1/2S\alpha_i\phi y_i^2 - 1/2\,C\alpha_i(\phi z_i + \Delta\phi_i)^2$$

$$uz = 3(2C\theta_i^2 - 1)S\alpha_i\phi x_i\phi y_i + 1/2\,C\alpha_i\phi y_i^2 + 1/2S\alpha_i^2(\phi z_i + \Delta\theta_i)^2$$

$$vx = 3S\alpha_i(\phi z_i + \Delta\theta_i)(\phi x_i + \Delta\alpha_i) + 3C\alpha_i\Delta\alpha_i\phi y_i - 1/2S\alpha_i\phi y_i^2 + 1/2C\alpha_i(\phi z_i + \Delta\phi_i)^2$$

$$
\begin{aligned}
vy &= -3/2(\phi x_i^2 + \Delta\alpha_i^2) + 3C\alpha_i S\alpha_i(\phi z_i + \Delta\theta_i)\phi y_i \\
&\quad - 3\alpha_i\phi x_i - 3/2\,C\alpha_i^2(\phi z_i + \Delta\theta_i)^2 \\
vz &= -1/2(\phi x_i^2 + \Delta\alpha_i^2) + 3/2\,C\alpha_i S\alpha_i(\phi z_i + \Delta\theta_i)^2 \\
&\quad - 3S\alpha_i^2(\phi z_i + \Delta\theta_i)\phi y_i \\
wx &= 3C\alpha_i(\phi z_i + \Delta\theta_i)(\phi x_i + \Delta\alpha_i) - 3S\alpha_i\Delta\alpha_i\phi y_i \\
&\quad - 1/2\,C\alpha_i\phi y_i^2 + 1/2\,S\alpha_i(\phi z_i + \Delta\theta_i)^2 \\
wy &= 1/2(\phi x_i^2 + \Delta\alpha_i^2) + 3/2\ C\alpha_i S\alpha_i\ [(\phi z_i + \Delta\theta_i)^2 \\
&\quad - \phi y_i^2] + 3C\alpha_i^2(\phi z_i + \Delta\theta_i)\phi y_i \\
wz &= -3/2(\phi x_i^2 + \Delta\alpha_i^2) - 3/2(\phi z_i + \Delta\theta_i)^2 \\
&\quad + 3/2\,C\alpha_i^2(\phi z_i + \Delta\theta_i)^2 - 3/2\,C\alpha_i^2\phi y_i^2 \\
&\quad - 3C\alpha_i S\alpha_i(\phi z_i + \Delta\theta_i)\phi y_i - 3\Delta\alpha_i\phi x_i \\
ttx &= 3(\phi z_i + \Delta\theta_i)(d_i\phi x_i - \Delta y_i) - 1/2\ S\theta_i^2\,[d_i\phi y_i^2 \\
&\quad + 3a_i(\phi z_i + \Delta\theta_i)^2] - 1/2\ C\theta_i S\theta_i[a_i(\phi z_i \\
&\quad + \Delta\theta_i)^2 - d_i\phi x_i^2] \\
tty &= -1/2\,C\theta_i S\theta_i[C\alpha_i d_i\phi y_i^2 + 3a_i C\alpha_i(\phi z_i \\
&\quad + \Delta\theta_i)^2] - 1/2\,C\theta_i^2[a_i C\alpha_i(\phi z_i + \Delta\theta_i)^2 \\
&\quad - C\alpha_i d_i\phi x_i^2] - 3/2\,S\alpha_i d_i\phi x_i^2 - 3C\alpha_i\phi x_i(\Delta d_i \\
&\quad + \Delta z_i) + 5/2\,S\alpha_i\phi x_i y_i - 3S\alpha_i\phi y_i\Delta x_i \\
&\quad + 1/2\,S\alpha_i\phi x_i\Delta x_i + 3C\alpha_i(\phi z_i + \Delta\theta_i)\,\Delta x_i \\
&\quad + 3/2\,S\alpha_i\phi y_i^2(d_i + a_i) + 3a_i C\alpha_i\phi x_i\phi y_i \\
&\quad + 3C\alpha_i d_i(\phi z_i + \Delta\theta_i)\phi y_i - 3S\alpha_i\Delta a_i\phi y_i \\
&\quad + 3C\alpha_i\Delta a_i(\phi z_i + \Delta\theta_i) \\
ttz &= [3S\alpha_i\phi x_i - 3C\alpha_i\phi y_i + 1/2\,C\alpha_i\phi x_i \\
&\quad - 3S\alpha_i(\phi z_i + \Delta\theta_i)]\Delta x_i + 5/2\,C\alpha_i\phi x_i\Delta y_i \\
&\quad + 3/2\,C\alpha_i(d_i + a_i)\phi y_i^2 - 3a_i S\alpha_i\phi x_i\phi y_i \\
&\quad - 3[S\alpha_i d_i(\phi z_i + \Delta\theta_i) + C\alpha_i\Delta a_i]\phi y_i \\
&\quad + 3S\alpha_i\Delta d_i\phi x_i - 3/2\,C\alpha_i d_i\phi x_i^2 \\
&\quad - 3S\alpha_i\Delta a_i(\phi z_i + \Delta\theta_i)C\theta_i S\alpha_i[1/2\ S\alpha_i d_i y\phi_i^2 \\
&\quad + 3/2\,a_i S\alpha_i(\phi z_i + \Delta\theta_i)_2] \\
&\quad + 1/2\,C\phi_i^2 S\alpha_i[a_i(\phi z_i + \Delta\theta_i)^2 - d_i\phi x_i^2].
\end{aligned}
$$

Here, $\Delta x_i, \Delta y_i, \Delta z_i$ and $\phi x_i, \phi y_i$, and ϕz_i represent linear and angular errors of x, y, z axes of the ith joint, respectively. The symbols $C\alpha_i$ and $S\alpha_i$ are abbreviated for cosine and sine of the angle α; and similarly for other angles.

Non-DH model. Non-DH models employ alternative link coordinate frame assignment procedures and thus, do not include the usual DH parameters. The most interesting point about the non-DH model considered in AREEM is that it employs the coordinate system representation commonly considered in engineering analysis subjects like solid mechanics, fluid mechanics, dynamics, etc. In other words, the kinematics of the robot motion is studied by attaching a Cartesian reference frame to each link at the joint axis, and then relating two consecutive frames by an origin shift vector and a direction cosine matrix whose elements represent tilts between the coordinate axes. Ideally, the coordinate axes are mutually orthogonal and intersect at a point. For an N-link open-loop manipulator with a nominal Cartesian coordinate system ${}_i x_k$ $(k = 1, 2, 3)$, it can be shown that two successive coordinate systems, i and $i+1$, are related as [11]

$$ {}_i x_k = ({}^{i+1}_{i}T_{mk})({}_{i+1}x_m) + {}_i w_k \tag{6} $$

where ${}_i w_k$ is the kth component of the origin shift vector in the ith coordinate system, and ${}^{i+1}_{i}T_{mk}$ is the mk element of the direction cosine matrix relating orientations of the two coordinate frames. Note that ${}^{i+1}_{i}T$ is an orthogonal matrix since ${}_i x$ and ${}_{i+1}x$ are Cartesian frames. Equation (6) is a standard Cartesian tensor transformation extended to incorporate an origin shift. The successive application of this relationship to all robot links yields a total compound transformation relating the end-effector coordinates, ${}_N x$, to the world coordinates, ${}_0 x$. Thus, the total nominal transformation can be expressed as

$$ {}_0 x_k = ({}^{N}_{0}T_{mk})({}_N x_m) + {}_0 w_k \tag{7} $$

where

$$ {}_0 w_k = \sum_{j=1}^{N} ({}^{j-1}_{0}T_{pk})({}_{j-1}w_p) \tag{8} $$

$$ {}^{N}_{0}T_{mk} = ({}^{N}_{N-1}T_{mp})({}^{N-1}_{N-2}T_{pq}) \ldots ({}^{i}_{0}T_{tk}). \tag{9} $$

Equations (7–9) represent an ideal case in which the end-effector position is related to the robot world coordinates.

When considering an inaccurate robot, the above formulation needs modification. The coordinate axes are no longer considered orthogonal and intersecting at a point. True robot link coordinate axes will be shifted and skewed relative to their assumed positions. Vaishnav and Magrab [11] have developed a scheme to analyze the problem considering 'slightly skew' frames using general tensor algebra. It was shown that the correct Cartesian coordinates, ${}_i x_k^c$ $(k = 1, 2, 3)$ of a desired point resulting from axis shifts and tilts is expressed in the ith frame as

$$ {}_i x_k^c = (\delta_{km} - {}_i H_{km})[({}^{i+1}_{i}T_{pm})({}_{i+1}x_p^c + {}_i w_m - {}_i\sigma_k)] \tag{10} $$

where ${}_i H_{km}$ represents angular errors and ${}_i\sigma_k$ are origin shift errors of the nominal ${}_i w_k$. The symbol δ_{km} is the Kronecker delta. Equation (10) is the counter part of Equation (6) which relates two successive frames, i and $i+1$, when the geometrical errors are incorporated. Note that products of ${}_i H_{km}$ and ${}_i\sigma_k$ were neglected since they produce higher order contributions. The model includes nine geometrical errors per joint axis of which six are represented by ${}_i H_{km}$ $(k = m)$ and three are ${}_i\sigma_k$. The total compound transformation incorporating the effects of all link errors can be represented by

$$
\begin{aligned}
{}_0 x_k{}^c = [\delta_{km} - \sum_{j=1}^{N+1} {}^{j-1}_{0}H_{km}]\,({}_0 x_m) - \sum_{j=1}^{N+2} ({}^{j-1}_{0}T_{km}) \\
({}_{j-1}\sigma_k) + \sum_{j=2}^{N+1}\sum_{g=2}^{j} ({}^{j-1}_{0}H_{kp})({}^{g-2}_{0}T_{rp})({}_{g-2}w_r)
\end{aligned}
\tag{11}
$$

where

$$ {}^{j-1}_{0}H_{km} = ({}^{j-1}_{0}T_{kp})({}_{j-1}H_{pq})({}^{0}_{j-1}T_{qm}) \tag{12} $$

$$ {}_0 x_m = ({}^{N}_{0}T_{1m})({}_N x_1) + {}_0 w_m \tag{13} $$

For consistency with Equation (3), the robot positioning error in world coordinates, dT, is represented by

$$dT = {}_0x^c - {}_0x \tag{14}$$

Equation (14) is valid for all k axes. Since the errors of each link relate to the link itself rather than to its relationship to neighboring links, it becomes easy to physically identify them [11]. Furthermore, no ill-conditioned matrices exist due to near parallel joint axes. These are two prime reasons for consideration of non-DH formulation.

PROGRAM DESCRIPTION

The AREEM program has been developed based on the models described in the previous section. The program uses MACSYMA to generate the three-dimensional change in the end-effector position dT due to robot geometrical errors. The use of MACSYMA provide the added flexibility to output results in algebraic form or as FORTRAN code. The FORTRAN representation of dT can directly be coupled with other robot calibration software. The program has been developed and tested on a VAX/750 computer under the VMS operating system.

There are two choices in selecting a programming language to code algorithms designed for generating equations in symbolic form. One choice is to use a general purpose language such as FORTRAN or BASIC to write programs that manipulate character strings. The other choice is to use an algebraic manipulation system designed specifically for handling symbolic expressions. Two well known algebraic manipulation systems are MACSYMA and REDUCE. MACSYMA can handle polynomials, matrix manipulation, symbolic solution of algebraic and differential equations, simplification, and substitution with symbolic expressions. REDUCE is similar, but provides less built-in functions and mathematical operations than MACSYMA. In developing the AREEM program, we have chosen MACSYMA for its powerful trigonometric and algebraic simplification capabilities and ease of its use.

Input. Inputs to run the AREEM program are: (i) number of robot links (degrees-of-freedom), (ii) type of each joint (revolute or prismatic), (iii) geometric parameters. After input items (i) and (ii) are received, a menu is provided allowing selection of a DH or non-DH model as indicated in the flow diagram of Fig. 3.

When a DH model is selected, the user is required to enter input item (iii), namely, the DH link parameters. Once the input has been interactively entered and interpreted, a link parameter table is displaced. Subsequently, the A_i and ${}^iT_N$ matrices as well as the end-effector pose, ${}^0T_N$, are computed.

If the non-DH model is selected, input item (iii) consists of the Cartesian axis of rotation of each revolute joint (x, y or z), and the three dimensional origin shift vector component (i.e. link offsets). For a prismatic joint, one of the origin shift vector components become the joint variable. After the user enters these data, they are reiterated on screen in tabular form. AREEM uses these data to generate the direction cosine matrices, ${}_i^{i+1}T$ ($i = 0, 1, \ldots N-1$), from which the total transformation, ${}_0^NT$, is computed using Equation (9). Finally, the end-effector position ${}_0x$ is computed.

Output. Once error equations are computed, AREEM provides the user with the option to output the forward kinematic solution (${}^0T_N$ for the DH models and ${}_0x$ for the non-DH models) if previously unknown. Then the positioning error dT is displayed as three equations representing the Cartesian components (dx, dy, dz) of the positioning error vector in world coordinates. For robots with many degrees-of-freedom, output expressions are often lengthy and difficult to analyze. To handle this problem the AREEM program, via MACSYMA, includes an option to perform simplifications and substitutions on program output. This is implemented with an exit from AREEM to the top level of MACSYMA and using MACSYMA commands such as TRIGSIMP and SUBST [21]. After the output is simplified, control is returned over to AREEM and the user is prompted with the option to display output in MACSYMA algebraic format, or as FORTRAN assignment statements. The desired output is displayed on screen and hard copies are obtained as needed.

In addition to using the program output in symbolic form, numerical output can be useful as well. If the error parameters are known, their values can be used to compute the magnitude of the actual end-effector positioning error at any location in the robot work volume. This gives a numerical indication of how far off the end-effector is relative to the commanded position. Numerical output can also be used for error model sensitivity analyses to study the effect of individual link error parameters on the total error in world coordinates. This can be done, in the forward sense, by numerically varying one error parameter at a time while the remaining error parameters are kept constant, and observing the corresponding effect on dT. One such study has been done and is discussed in the next section.

EXAMPLE AND RESULTS

To demonstrate the usefulness of the AREEM program, we have applied it to the 3-link planar robot illustrated in Fig. 4. The DH link parameters are listed in Table 1. Upon selecting the 5 parameter DH model from the screen menu, the positioning error output may be displayed as shown in Fig. 5. The orientation error output as well as other models' output may be displayed. These results are

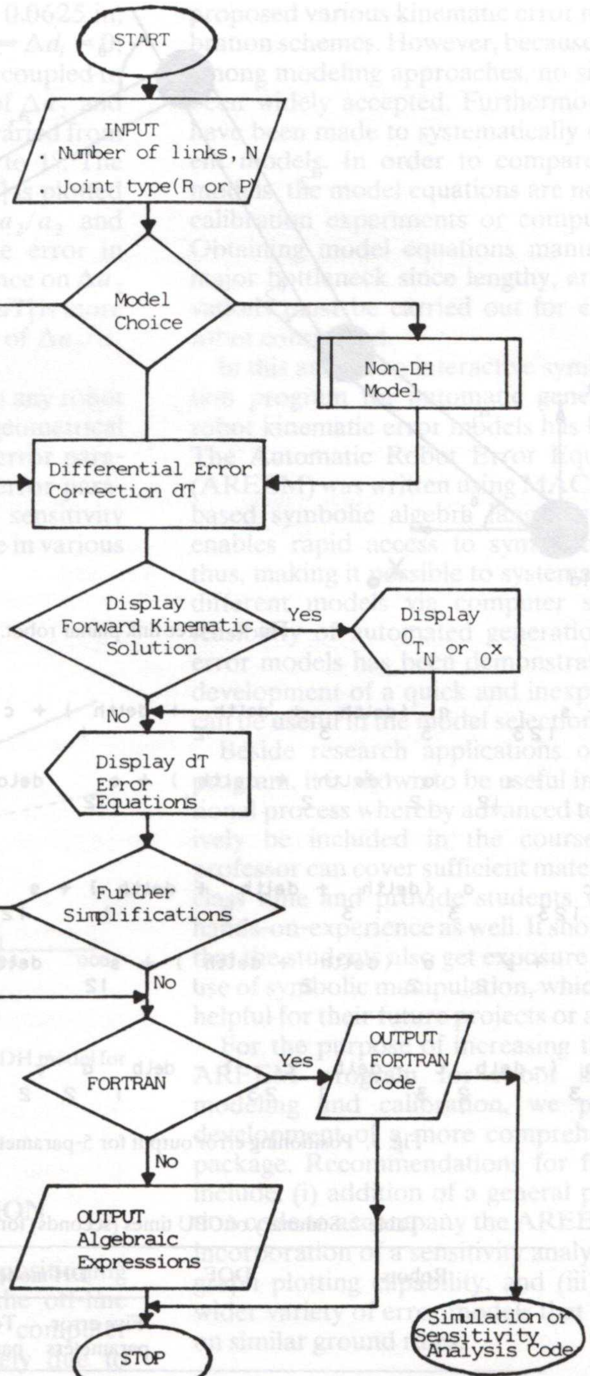

Fig. 3. Overall flow diagram for AREEM Program.

Table 1. DH link parameters for a 3-link planar robot

Link	*Variable*	α	a	d
1	$\theta_1 = 40°$	0	20 in	0
2	$\theta_2 = 20°$	0	20 in	0
3	$\theta_3 = 70°$	0	20 in	0

not shown here because of space limitation (the expressions are very lengthy).

The program has been applied to four additional robot configurations and CPU time values in seconds to generate model error equations are listed in Table 2. These CPU values do not include the time required for algebraic and trigonometric simplifications. Such simplification time is, generally, on the order of the time spent in generating the error equations. Note that the CPU time required for symbolic manipulations is not totally dependent upon the number of addition and multiplication operations. It also depends on the amount of virtual memory being used; this is particularly true on multi-user computer operating systems. Nevertheless, significant time savings are realized when

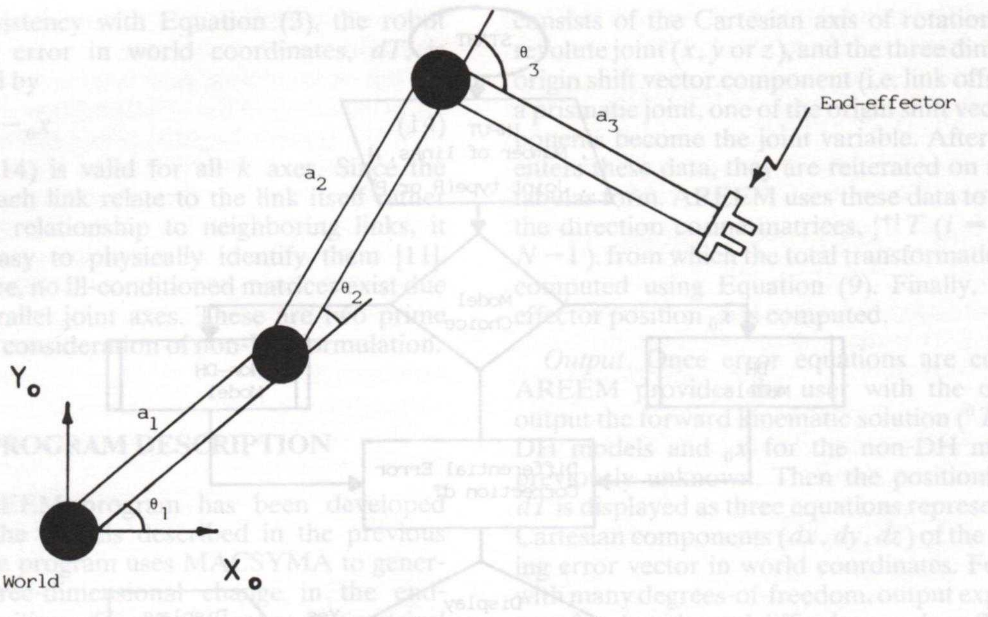

Fig. 4. Three link planar robot.

$$dx = -s_{123}\,a_3\,(delth_3 + delth_2 + delth_1) + c_{123}\,dela_3 - s_{12}\,a_2\,(delth_2 + delth_1) + c_{12}\,dela_2 - a_1\,delth_1\,s_1 + c_1\,dela_1$$

$$dy = c_{123}\,a_3\,(delth_3 + delth_2 + delth_1) + s_{123}\,dela_3 + s_{12}\,a_2\,(delth_2 + delth_1) + s_{12}\,dela_2 + dela_1\,s_1 + a_1\,c_1\,delth_1$$

$$dz = a_3\,(-delb_2\,c_3 - delb_1\,c_{23}) - delb_1\,a_2\,c_2$$

Fig. 5. Positioning error output for 5-parameter DH model.

Table 2. Summary of CPU times (seconds) for typical robots

Robots	DOF	DH models		Non-DH models
		Five error parameters	Ten error parameters	Nine error parameters
3-Link Planar	3	41.33	43.78	50.52
Minomover Microrobot	5	50.42	54.92	123.48
Stanford Arm	6	83.23	87.00	161.23
Elbow Manipulator	6	84.22	88.07	163.92
PUMA 600	6	85.27	89.37	174.80

using the AREEM program. The rapid access to kinematic error models demonstrated by this program facilitates implementing comparisons of different models' performance.

To further describe the applicability of AREEM, we make use of the positioning error output to investigate the sensitivity of the total Cartesian error (in world coordinates) with respect to individual geometric error parameters. For the sake of demonstration, we arbitrarily selected the first-order five-parameter DH model for the 3-link planar robot. The link parameters are given in Table 1.

The DH parameters θ, α, a, d in Table 1 are joint angle, link twist, link length and link offset, respectively. Link 2 was varied to study the effect of a length error, Δa_2, and an orientation error, $\Delta\theta_2$, on the magnitude of the positioning error in world coordinates. $|dT|$. All geometric error parameters are assumed to be known except Δa_2, and $\Delta\theta_2$. The assumed error values are $\Delta a_1 = \Delta a_3 = 0.0625$ in, $\Delta\theta_1 = \Delta\theta_3 = 0.1^0$, $\beta_i = 0.05^0$, $\Delta\alpha_i = \Delta d_i = 0$; $i = 1, 2, 3$. The FORTRAN output was coupled to a separate program where the effects of Δa_2 and $\Delta\theta^2$ on dT were studied. The Δa_2 was varied from 0 to 1 in. and $\Delta\theta^2$ was varied from 0^0 to 1^0. The results are shown in Fig. 6 in which $|dT|$ is plotted against nondimenzionalised values $\Delta a_2/a_2$ and $\Delta\theta_2/\theta_2$. The Figure indicates that the error in world coordinates has a linear dependence on Δa_2 and $\Delta\theta_2$ for the range considered. The $|dT|$ is more sensitive to Δa_2 than $\Delta\theta_2$ as the ratio of $\Delta a_2/a_2$ and $\Delta\theta^2/\theta_2$ increases.

Similar analyses can be carried out on any robot to gain insight into possible sources of geometrical error and to investigate the effects of error parameters on dT. In general most of the error parameters should be included in the sensitivity analysis; and the analysis should be done in various regions of the robot work volume.

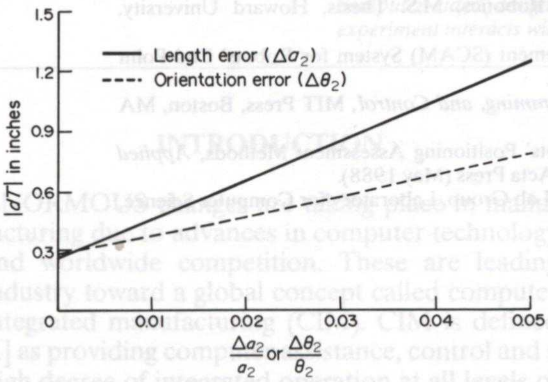

Fig. 6. Sensitivity plot of first-order 5-parameter DH model for Link 2.

SUMMARY AND CONCLUSION

Improvement in present robot positioning accuracies is essential for achieving the off-line programming capability required for computer integrated manufacturing. Unfortunately due to inherent difficulties in manufacturing of robots' links and joints, geometrical errors arise which contribute to pose inaccuracy. In an effort to improve robot accuracy, many researchers have proposed various kinematic error models and calibration schemes. However, because of the diversity among modeling approaches, no single model has been widely accepted. Furthermore, no attempts have been made to systematically compare different models. In order to compare various error models, the model equations are needed for use in calibration experiments or computer simulation. Obtaining model equations manually presents a major bottleneck since lengthy, error-prone derivations must be carried out for each model and robot considered.

In this article, an interactive symbolic manipulation program for automatic generation of three robot kinematic error models has been presented. The Automatic Robot Error Equation Modeler (AREEM) was written using MACSYMA, a LISP-based symbolic algebra language. The program enables rapid access to symbolic error models, thus, making it possible to systematically compare different models via computer simulation. The feasibility of automated generation of kinematic error models has been demonstrated through the development of a quick and inexpensive tool that can be useful in the model selection process.

Beside research applications of the AREEM program, it is shown to be useful in robotic educational process whereby advanced topics can effectively be included in the course content. The professor can cover sufficient material in minimum class time and provide students with reasonable hands-on-experience as well. It should be indicated that the students also get exposure to the powerful use of symbolic manipulation, which in turn can be helpful for their future projects or assignments.

For the purpose of increasing the utility of the AREEM program for robot kinematic error modeling and calibration, we propose further development of a more comprehensive software package. Recommendations for further research include: (i) addition of a general purpose calibration code to accompany the AREEM program, (ii) incorporation of a sensitivity analysis module with graph plotting capability, and (iii) inclusion of a wider variety of error models that are comparable on similar ground rules.

Acknowledgements—Authors wish to acknowledge Mr Edward Tunstel, Jet Propulsion Laboratory, Pasadena, California for programming effort. This research was partially supported by Howard University Faculty Research Program, No. 300273.

REFERENCES

1. N. Vira, Robots' End Point Sensing: Hardware and Software Techniques, *Computer in Industry*, **13** (11), pp. 1–12 (March 1989).
2. W. Veitschegger and C. Wu, A Method for calibrating and compensating robot kinematic errors, *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, 39–44 (April 1987).
3. W. Veitschegger and C. Wu, Robot Accuracy Analysis Based on Kinematics, *IEEE J. Robotics and Automation*, RA-2, No. 3, 171–179 (September 1986).
4. S. Hayati, Robot Arm Geometric link parameter Estimation, *Proceedings of the 22nd IEEE Conference on Decision and Control*, 1477–1483 (December 1983).

5. T. Hsu and L. Everett, Identification of the kinematic parameters of a robot manipulator for position accuracy improvement, *Proceedings of Computers in Engineering Conference and Exhibition*, 263–267 (1985).
6. D. Whitney, C. Lozinski and J. Rourke, Industrial Robot Forward Calibration method and Results, *J. Dynamic Systems, Measurement, and Control*, **108**, 1–8 (1986).
7. H. Stone, A. Sanderson and C. Neuman, Arm Signature Identification, *Proceedings of IEE International Conference on Robotics and Automation*, 41–48 (April 1986).
8. M. Driels and U. Pathre, Generalized Joint Model for Robot Manipulator Kinematic Calibration and Compensation, *J. Robotic Systems*, **4**, 66–114 (1987).
9. J. Chen and L. Chao, Positioning Error Analysis for Robot Manipulators with All Rotary Joints, *Proceedings of IEEE International Conference on Robotics and Automation*, 1011–1016 (April 1986).
10. B. Mooring, The effect of Joint Axis Misalignment on Robot Positioning Accuracy, *Proceedings of ASME Computers in Engineering Conference*, 151–155 (August 1983).
11. R. Vaishnav and E. Magrab, A General Procedure to Evaluate Robot Positioning Errors, *Int. J. Robotics Research*, **6**, 59–74 (1987).
12. J. Siegert and P. Datseris, Basic Considerations for Robot Calibration, *IEEE International Conference on Robotics and Automation*, 932–938 (1988).
13. R. Indigo and J. Angulo, University Robotics Education: Theory and Practice, *Recent Trends in Robotics: Modeling, Control and Education*, pp. 355–364. Elsevier Science Publication Co. (1986).
14. E. Muth, A Graduate Course in Industrial Robotics, *Recent Trends in Robotics: Modeling, Control and Education*, pp. 365–371. Elsevier (1986).
15. R. Toogood, A Project Oriented Introductory Robotics Course, *Proceedings of ASME Computers in Engineering Conference and Exhibition*, 5–8 (July 1986).
16. A. Bennett and J. Luh, Robotics Research Reflecting Practical Education, *Recent Trends in Robotics: Modeling, Control and Education*, pp. 441–444. Elsevier (1986).
17. E. Tunstel, Applied Symbolic Computation in Robotics, M.S. Thesis, Howard University, Washington, DC (August 1989).
18. N. Vira, Spherical Coordinate Automatic Measurement (SCAM) System for Robots' End-Point Sensing, *Precision Engng*, **11**, 151–164 (July 1989).
19. R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Boston, MA (1981).
20. N. Vira and V. Goodwyn, Developments in Robots' Positioning Assessment Methods, *Applied Simulation Modelling '88, IASTED*, pp. 194–198. Acta Press (May 1988).
21. MACSYMA Reference Manual, Version 10, Math Lab Group, Laboratory for Computer Science, MIT (1983).