Engineering Journal of Qatar University, Vol. 4, 1991, p. 141 - 156.

MICROCOMPUTER AIDED SELECTION OF ROBOT MANIPULATORS

Tarek M. Abdel-Rahman and Nabeel H. Salem
Mechanical Engineering Department, Qatar University,
Doha, Qatar

ABSTRACT

This paper presents two programs for microcomputer aided assessment of the performance of robot manipulators. The first program automatically generates robot models based on user-supplied kinematic parameters. The program also derives a kinematic model that relates the motion of manipulator end-effector to the motion of the joints using the inverse kinematic approach. The approach uses a robust inversion technique that can handle singular conditions as well as joint redundancy. A user can optionally select evaluation of kinematic capabilities of the robot manipulator, such as the ability of the end-effector to reach a specified position and orientation in space or the evaluation of the work space. The second program generates dynamic variables, such as forces and torques, based on user-supplied dynamic parameters and equations of motion of the various joints.

Both programs are written for implementation on personal computers. Several runs were carried out to demonstrate the capability and execution times of the two programs.

INTRODUCTION

The advent of computer-controlled industrial robot that can be programmed to perform a set of tasks will eliminate the need for high cost, custom designed automation equipment. It can provide for the automation of product assembly and its low cost will make possible the automation of small batch production shops. Of course, industrial robots because of its physical limitations will be only capable of performing a set of tasks that do not violate any of the robot physical constraints.

To enable selecting the right robot from several candidates or to evaluate the capability of an available robot for doing a set of tasks or to evaluate the robot performance when using various control schemes we need to employ a robot simulation package. Development of such packages has been recently receiving extensive attention, see for example Refs. 1-5. Ref. 1 gives a brief survey of various approaches used by existing robot simulation packages. Firstly-developed packages often require user knowledge of computer programming to use them. Secondly-

This paper is based on a graduation project work of the second author.

developed packages offered increased flexibility but are limited to simulation only of robots with predefined configurations. A third group of packages create semi-automatic controlling equations for customized models and require that the user be familiar with the modeling system on which these simulation packages are built. The most recent group of packages provide automatically generated solid-robot-models based on user supplied kinematic paramers, Refs. 1-3.

Extensive support of dynamic simulation is not available in most robot simulators. Few packages provide users with dynamic analysis such as forces and torques for all links to determine effects of these variables on robot performance. Users must input dynamic data such as inertia of all links and motor torque characteristics. Automatically generated simulated dynamics and real time dynamic simulation are still in the research stage. For example, Ref. 6 indicates some on-going research attempting to verify robot motions in terms of dynamic performance, including overshoot, and tool settling time.

Packages available were developed for implementation on mainframes or minicomputers. These packages depend on the availability of large memory, fast processor and large disk space. And in addition, these packages are usually expensive. The high cost of such hardware and software, of course, hinders the introduction of robotic to small batch production shops. On the other hand, the availability of low cost microcomputers spurs the investigation of the feasibility of having low-cost robot simulation package that can support the selection and use of robot manipulators in small batch production shops. The use of microcomputers for simulation has its implications on the quality of the results obtained, Ref. 7, but in many applications the advantages of the low cost of the microcomputer based system outweigh the quality improvement of results obtained by mainframe or minicomputer based simulator system.

In this paper we focus on the development and implementation of robot simulation package on personal computers. The implementation of simulation package on personal computer reduces the cost of performance evaluation significantly but requires lengthy computation time. To reduce this time, the main kinematic, dynamic and control characteristics are only considered. In addition whenever we see that a very time consuming computation will be needed for a complete evaluation, we resort to an interactive support of the user to limit his range of interest to a reasonably selected set of variables.

The robot modeling and simulation package developed for performance evaluation of robot manipulators comprises two programs. The first program evaluates the kinematic characteristics of the robot. The program comprises four modules as shown in Fig. 1. The first module automatically generates a robot model from an interactively entered specifications that describe the structure and kinematical parameters of the robot. Animated motion of a solid or wireframe

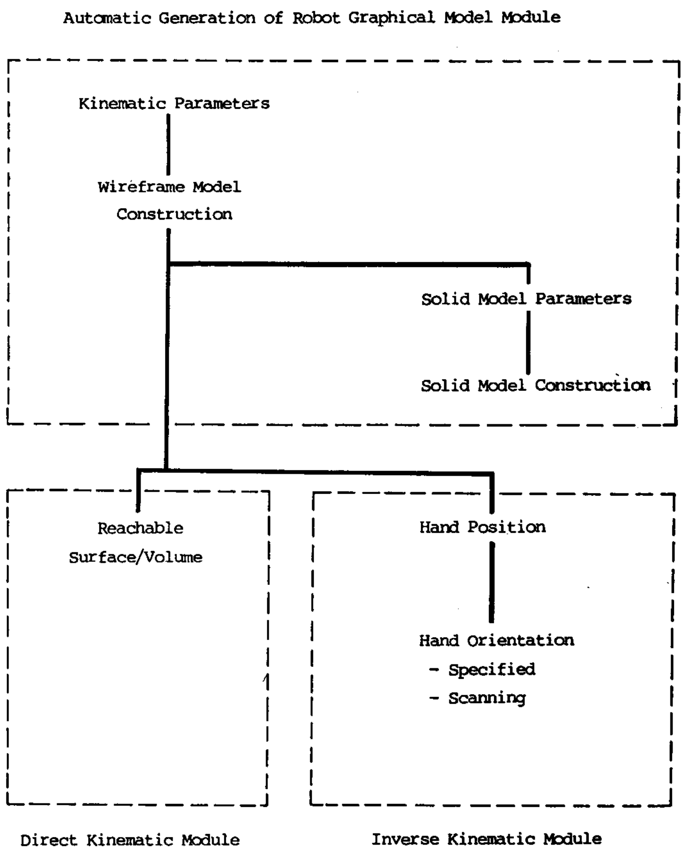

Fig. 1: Kinematical program modules

model can then be obtained. The second module solves a direct kinematic problem to evaluate the working space of the manipulator. The third and fourth modules solve inverse kinematic problem to evaluate the capability of end-effector to reach specified position and orientation. The first program is described in the first section.

The second program uses the joint kinematic variables computed by the first program to compute torques and forces needed to be applied by the joint actuators to move the end-effector along a specified trajectory. This is described in the second section.

The third section presents execution times of the various modules on Apricot and on IBM-PC compatible microcomputers. Techniques used, or that can be used, for enhancing execution time are then provided.

The simulation program described in this section is characterized by its capability of automatic modeling of robots from user entered robot kinematic parameters. The parameters entered are prompted by appropriate program questions. The user needs not to be familiar with any particular modeling method notation. All what is required is to know how to define the motion of revolute and prismatic joints according to the notation shown in Fig. 2 where the Y axis is taken along the longitudinal axis of the links. Fig. 3 shows the simplified interactive kinematic parameters needed to generate a simplified wireframe model. An example of a wireframe model is shown in Fig. 4. The parameters that must be supplied are:

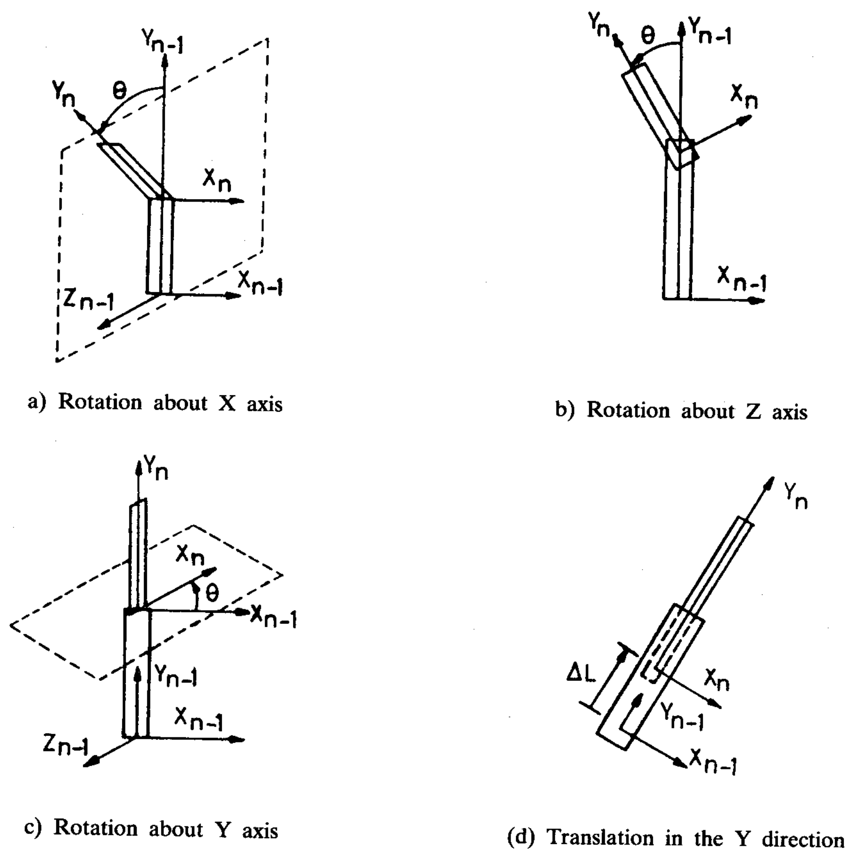

Fig. 2: Manipulator joint notation

- total number of links and joints, link lengths, type of joints (rotational/revolute, translational/prism) and rotational axes,
- maximum and minimum joint limits,
- initial position of each joint.

As shown in Fig. 4, we represented the links by the position of only four points in space to reduce computational time. And for the same purpose we also avoided drawing circules. This simple wireframe representation is appropriate for low speed computer implementation and is sufficient for preliminary and many practical kinematic evaluations of robot performance. Obviously, this representation is appropriate in selection evaluations, i.e. for screening robots capable of performing particular tasks.

```
Number of articulated segments                                  =?  4
Base height                                                     =?  50
Segment number 1 length                                         =?  40
Segment number 2 length                                         =?  80
Segment number 3 length                                         =?  80
Segment number 4 length                                         =?  40
End effector length                                             =?  20
Type of joint (1) is 1) Rotation 2) Translation                 =?  3
Choose 1 or 2, TRY AGAIN
Type of joint (1) is 1) Rotation 2) Translation                 =?  1
Along which axis joint (1) is acting:
                          1) X, 2) Y and 3) Z                   =?  2
Minimum and maximum (angle or length) of joint                  =?  -180,180
Type of joint (2) is 1) Rotation 2) Translation                 =?  1
Along which axis joint (2) is acting:
                          1) X, 2) Y and 3) Z                   =?  3
Minimum and maximum (angle or length) of joint                  =?  -100,150
Type of joint (3) is 1) Rotation 2) Translation                 =?  1
Along which axis joint (3) is acting:
                          1) X, 2) Y and 3) Z                   =?  3
Minimum and maximum (angle or length) of joint                  =?  -135,135
Type of joint (4) is 1) Rotation 2) Translation                 =?  1
Along which axis joint (4) is acting:
                          1) X, 2) Y and 3) Z                   =?  3
Minimum and maximum (angle or length) of joint                  =?  -135,135
Minimum and maximum angle rotation (about Y axis)
  of end-effector                                               =?  -180,180
Specific joint initial position:
Joint (1)                                                       =?  0
Joint (2)                                                       =?  -20
Joint (3)                                                       =?  110
Joint (4)                                                       =?  90
End effector                                                    =?  90
```

Fig. 3: Computer prompts and user entered parameters

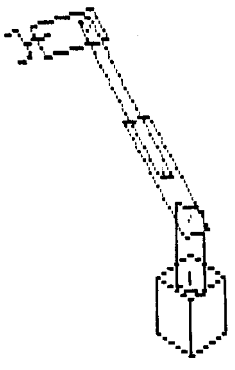
Fig. 4: Automatically generated wireframe robot model

The program provides an option for solid modeling. If this option is selected, the program prompts the user to enter extra kinematic parameters needed for solid modeling, such as the cross sectional dimensions of all links. An example of creation of a solid model is shown in Fig. 5.

Two approaches can be used for the automatic generation of robot kinematic models. The first is the Denavit-Hartenterberg's (D-H) approach described in Ref. 8 and the second is the Borrel's approach which is a classical one, described in Ref. 9. The later uses the notation shown in Fig. 2. Although the D-H notation is the most general one, it has not been used here as it requires a trained user to use the program. Instead, the Borrel's notation and approach of Ref. 9 is used because it is

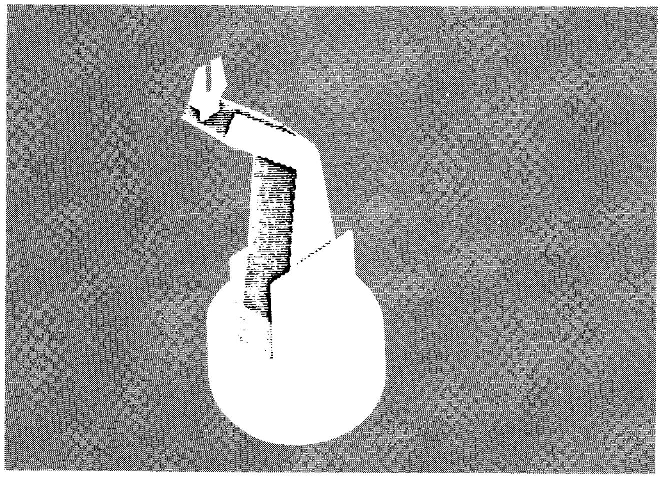
Fig. 5: Automatically generated solid robot model

simpler and requires no user training. The approach can automatically generate kinematic models for a large range of industrial robots. Further capabilities for handling particular link shapes or other complexities will be added in the future with graphic icons support. The procedure of automatic generation of mathematical and graphical model is described below.

Procedure 1: Automatic Generation of Robot Model:

1) Accept user definition of number of joints, lengths and links, joint types, maximum and minimum joint limits, and axes of joint motions;

2) accept user specifications of initial positions of joints;

3) accept user selection of creation of wireframe or solid model and specifications of cross section dimensions if the solid modeling is selected;

4) construct rotational and translational transformation matrices using Borrel's approach, Ref. 9;

4) compute the transformation matrices of joints;

5) compute consecutive transformation matrices to obtain the position of the points that describe the links;

6) compute isometric projection of the position of the points that describe the links and connect them to obtain a model for the manipulator;

7) if the solid modeling is selected remove hidden surfaces and paint the visible ones with appropriate colour and brightness.

The second module of the program addresses the direct kinematic problem for evaluation of reachable (or working) surfaces and volumes. The user selects some of the joints to be locked at specified values, and gives the range and incremental steps of motion for the remaining joints. The motion of the robot is then animated and the traces of end-effector motion are plotted.

An example of a resultant motion due to incremental steps in two joints is shown in Fig. 6 which represent a working surface area. A work or a reachable volume can be obtained due to motion of three joints that have no parallel axes. The procedure of the module is summarized below.

Procedure 2: Direct Kinematic Problem Module:

1) Invoke steps 1-6 of procedure 1 to generate a kinematic robot model;

2) accept user selection of the joints that should be varied and their incremental steps and maximum and minimum limits;

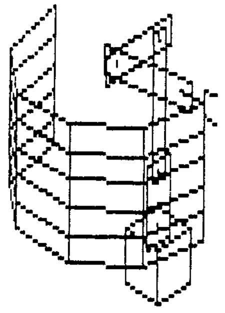
Fig. 6: End-effector scanning of a work space area

3) repeat steps 4-6 for all combinations of moving joint positions;
4) compute the consecutive transformation matrices of the set of joint positions to determine the position of links and consequently the position of the end-effector;
5) invoke step 6 of procedure 1 to draw a wireframe robot model;
6) save the position coordinate of the end-effector;
7) draw and/or print the end-effector positions.

In evaluating robot performance, the user may be interested to check whether the manipulator can move the end-effector from an initial to a specified final position. The user may also need to evaluate the capability of the end-effector in achieving a final specified orientation. In essence, the user specifies the final end-effector position and wants to determine the corresponding joint positions. This is known as part of the inverse kinematic problem in robotics. Refs. 8-9 show how to solve this problem which, in general, requires using a matrix inverse algorithm or a robust generalized inverse routine to handle singular positions, e.g. Ref. 10. The procedures of the end-effector position and orientation modules are described below:

Procedure 3: Inverse Kinematic Problem: I — End-effector Position Module:

1) Accept initial and final end-effector position coordinates specified by the user;
2) create end-effector translational motion steps along a line path connecting the initial and final positions of the end-effector;
3) solve for the joint incremental motion needed to produce the end-effector translations of step (2) by inverting the Jacobian matrix and obtaining a practically feasible joint solutions;

4) compute consecutive matrix transformations to obtain changes of link and end-effector positions due to increments of joint positions;

5) invoke step 6 of procedure 1 to draw a wireframe model of the robot;

6) repeat steps 3-5 until either the required end-effector position is obtained or a problem of achieving a specific position is encountered and prompt the user by the final status.

Procedure 4: Inverse Kinematic Problem: II — End-effector Orientation Module:

1) Apply procedure 3 to check if the specified end-effector position is reachable or not and if yes, proceed through the following steps;

2) accept the user selection of one of the two options: — check the capability of reaching a specified orientation, — check all possible orientations at the present end-effector position;

3) for a specified orientation, solve the inverse kinematic problem to determine joint increments for obtaining the required orientation;

4) compute consecutive transformation matrices to obtain the position of the links and end-effector;

5) invoke step 6 of procedure 1 to draw a wireframe model of the robot;

6) if the option of scanning all possible orientations is selected then repeat steps 3-5 for various hand orientations and prompt the user of the results and of difficulties of achieving some of the orientations when encountered;

7) provide a wireframe model of the robot and a diagram showing end-effector orientation at a specific position, as shown in Fig. 7, or all possible end-effector orientations at specific positions, as shown in Fig. 8.

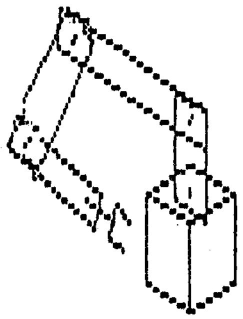

Fig. 7: Manipulator end-effector achieving a prespecified position and orientation

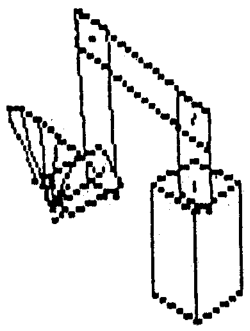

Fig. 8: End-effector scanned orientations at a specified position

DYNAMIC PERFORMANCE EVALUATION

Two approachers are used in dynamic modeling and simulation of robots. The first uses a user written routine to compute joint torques. The second approach automatically generates a dynamic model, Ref. 8, using the kinematical and dynamical parameters provided by the user. In this paper, we used the first approach because it is simpler and requires less computation which is appropriate for microcomputer applications. Of course, the second approach is better as it does not require user familiarity with the dynamics of the robot and its axes notations. This approach is currently under investigation for future implementation.

To obtain joint torques to move the end-effector along a prespecified path, we run the inverse-kinematic position module at first, to obtain a sequence of joint displacements. The sequence is saved in a file which is then fed to the dynamic module. The procedure of computation is summarized below.

Procedure 5: Joint Forces and Torques:

1) Accept manipulator kinematic and dynamic parameters entered by the user;
2) read joint-position sequences for moving the end-effector along a specified path;
3) accept user entered time period required between motion sequential steps;
4) compute joint speeds and accelerations based on the incremental joint positions and user required time step;
5) check joint speeds and accelerations against their maximum and minimum limits, and if any joint violates one of its limits then reduce the time period and repeat steps 4 and 5;

6) compute joint forces and torques using the user supplied routine;

7) check joint forces and torques against their limits and if any joint violates one of its limit, then reduce the time period and repeat steps 4-7;

8) plot speed, acceleration and torque/force for selected joints.

MICROCOMPUTER IMPLEMENTATION EXPERIENCE

We coded the two programs in BASIC language. The special graphic commands of the GW-BASIC version 2 were used. The two programs were run at first under GW-BASIC Interpreter on the IBM-PC and NCR-PC4I microcomputers which employ Intel 8086 processors and on Apricot XI-20 microcomputer which employs an Intel 8086 processor. The screen resolution for the Apricot, IBM-PC, and NCR-PC4I monitors used were 800x400, 640x200, and 640x400 respectively.

The kinematic performance of four-joint robot manipulators were evaluated, Figs. 4, 6-8. The computation time for each complete cycle of procedure 1 or 2 elapsed 15, 15, 20 seconds for the IBM-PC, NCR-PC4I, and Apricot microcomputers respectively. The cycle of procedure 3 elapsed 144, 144, and 170 seconds and the cycle of procedure 4 elapsed 160, 160, 185 seconds for the IBM-PC, NCR-PC4I and Apricot microcomputers respectively.

To speed up computation we obtained a compiled (an executable program) version of the nongraphics portion of the first program. The graphical simulation portion was excluded because the available Compiler, at time of evaluation, did not support graphic commands. Therefore, we let the compiled portion finish all computation required and save the joint positions obtained in a RAM disk file. This file was then read by the graphical simulation portion of the program for display. This approach has led to execution time reduction by a factor of 10 for computation. When an 8087 coprocessor support was usd, we obtained an additional time reduction factor of 1.5. Thus the complete solution of one direct kinematic problem with wireframe representation required 1-1.3 seconds on IBM-PC compatible and an Apricot XI-20 microcomputers.

The solid modeling represenatation of 4 joint robot manipulator on the Apricot XI-20 microcomputer required 140 seconds when run under GW-BASIC interpreter. This time was reduced to about 115 seconds when we used the support of the complier and the mathematical coprocessor.

The first program with the GW-BASIC interpreter occupied 64 K Byte of the computer memory. Due to this size limitation the interpreter did not allow direct further expansion to allow for extra facilities or for increasing the number of points considered on the wireframe model. Of course, these problem can be overcome by dividing the program into several ones and pass data through them via external files. Also language Interpreters and Compilers that don't have the 64 K Byte limitation have recently started to appear. Thus the size of the program will not be a serious limiting factor for implementing robot simulation and evaluation packages on microcomputers. Moreover, the problem will tend to vanish if we use improved Interpreters and Compilers or microcomputers of the Advanced Technology (AT) generation (which uses 16 bit processors, e.g. Intel 80286) or higher performance technology of the 80386 or 80486 generations.

On the speed of computation side, we can see that the current technology is appropriate for solving direct kinematic problem in off or on-line mode. The off-line computation of the inverse kinematic problem which has best computation time in the order of 10 seconds is reasonably acceptable. This time is considered high in on-line applications specially if there is a man controlling the arm (e.g. in resolved-rate and resolved-acceleration operator control). When a man is involved in the control loop, the computation period and joint response time should be less than 2 seconds.

Advanced technology does not offer too much improvement in computation speed. Only a factor between 10 to 20 enhancement is expected. On the other hand, better computation speed improvement for on-line applications, can be obtained by using floating point accelerators with software that supports mathematical functions and in particular trignometric ones. These type of floating point accelerators can speed up computation by a factor of 20 as claimed by Systolic Systems Company of San Jose, California, for example. However, these accelerators are usually costly. The cost can be larger than the cost of the microcomputer system itself.

The accelerators reduce computation time but don't affect the time of drawings. In the four joint manipulator example considered here, the speed of computation alone for the inverse kinematic problem was about 1.5 seconds. This time increases when the number of joints increases. To decrease this time we either use floating point accelerators or personal computers with faster central processors and math-coprocessors.

Procedure 5 was used to compute the inverse dynamic problem of a robot manipulator. The end-effector path was a straight line as shown in Fig. 9. Fig. 10 shows a schematic diagram of the 4 joint robot used in the study. Each iteration required less than a second for computation of the joint speed, acceleration and torques. Examples of the program output results are shown in Figs. 11-13 for joint number two. Finer time steps can be easily obtained for smoother output results.

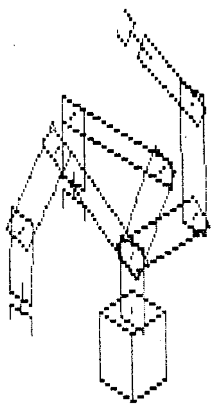

Fig. 9: Manipulator following a straight line path

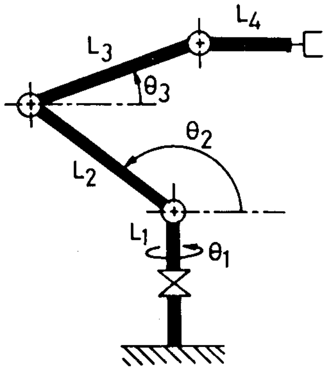

Fig. 10: Schematic diagram of the robot used in the dynamic performance evaluation

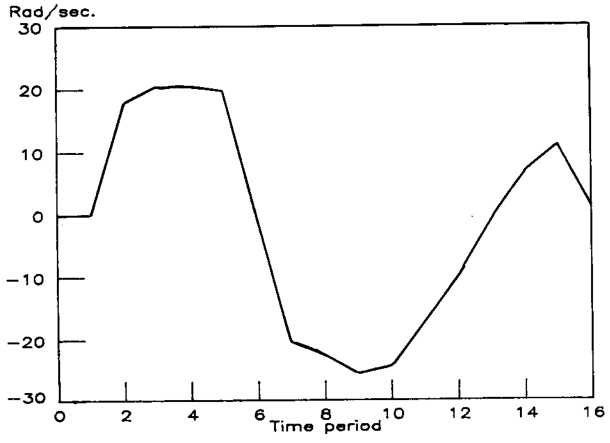

Fig. 11: Speed of joint no. 2

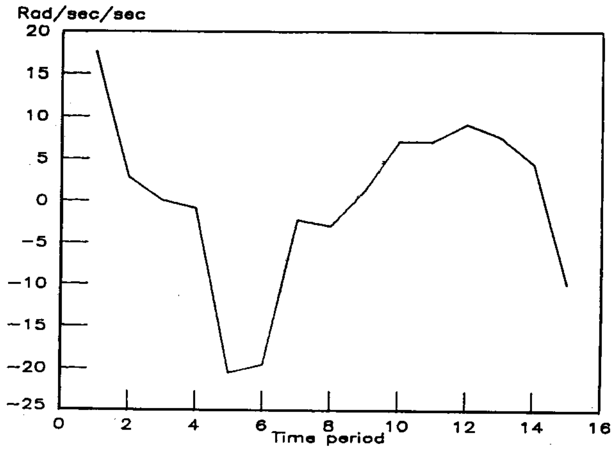

Fig. 12 Acceleration of Joint no. 2

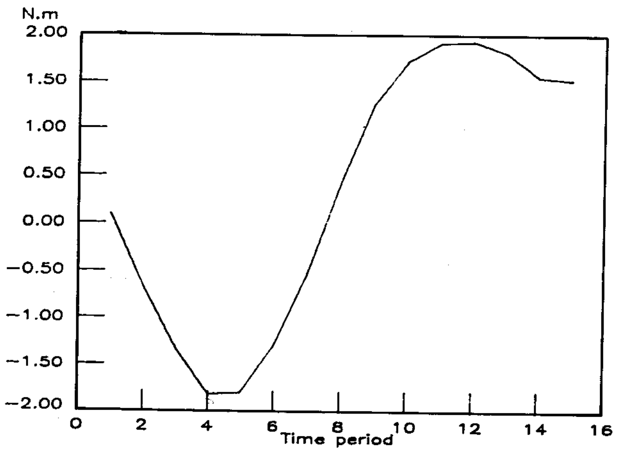

Fig. 13 Actuator torque of Joint no. 2

SUMMARY AND CONCLUSIONS

Two computer programs for evaluating the performance of robot manipulators on personal computers are presented. The first program has several modules. One module creates wireframe and solid model of robot manipulators based on user-defined kinematic parameters. The other modules solve the direct kinematic problem to evaluate reachable surfaces and volumes, and solve the inverse kinematic problem to evaluate the capability of the end-effector to achieve a specified position and orientation. The second program computes joint torques or forces required to move the end-effector along a specific trajectory.

The two programs can be used to evaluate the kinematic and dynamic characteristics of many manipulators to select the right one suitable for performing a range of tasks. Program sizes can be easily accommodated by recently provided Interpreters and Compilers. Several computer runs for robot simulations revealed execution times (using compiled versions supported by a mathematical cop-rocessor) in the order of 1 second and 10 seconds for one cycle of the solution of direct and inverse kinematic problems respectively. Also solid modeling was found to take several multiples of the time needed for creating simplified wireframe models. Thus it would be reasonable to adopt simplified wireframe models in robot simulation on personal computers unless there is a particular need for creating solid models. The computation time of the direct and inverse kinematic problem show that the current technology of microcomputer is appropriate for off-line computations. In on-line applications that use the inverse kinematics approach the need may rise for using a floating-point accelerator or a central processor with high clock speed. The need depends on the number of joints of the manipulator.

On-line computations that use inverse kinematics in a closed loop system will be subjected to much stringent computation speed conditions that cannot be determined without knowing the structure of the controller, the sensor and the robot to be controlled.

ACKNOWLEDGEMENT

The authors acknowledge the contribution of Mr. Mohamed S. Takriti in developing the computer module for the generation of the solid modeling of the robot.

REFERENCES

1. **Szeto, K.W., 1985,** "Simulation of Parameterized Robots with Solid Modeling", Proceedings of the IASTED International Symposium on Robotics and Automation, Lugano, June 24-26.

2. **Szeto, K.W., 1985,** "An Approach to Modeling Robots in Workcells for Graphical Simulation", Masters Thesis, UCLA Manuf. Engr. Program.

3. **Szeto, K.W.** *et al.*, "Parameterization of Robots for Simulation", UCLA Manuf. Engr. Program Report 8501.

4. **Derby, S.J. 1982,** "General Robot Arm Simulation Program (GRASP): Part 1, A Program to Evaluate the Performance of Industrial Robots in their Working Environment", Proc. 2nd Intl. Comp. Engr. Conf., ASME.

5. **Derby, S.J., 1982,** "General Robot Arm Simulation Program (GRASP): Part 2, Methods of Joint Solutions and the Reachable Volume", Proc. 2nd Intl. Comp. Engr. Conf., ASME.

6. **Stauffer, R., 1984,** "Robot System Simulation", Robotics Today.

7. **Ahmed, S.V., 1985,** "Microcomputer for Simulations and Graphics: A Comparison with Mainframes and Minicomputers", Proceedings of IECON'85, the 1985 conference on Industrial Electronics, Control, and Instrumentation, Nov. 18-22, pp. 588-592.

8. **Paul, R.P. 1981,** Robot Manipulators, MIT Press.

9. **Coiffet, P., 1981,** Robot Technology: Modelling and Control, Prentice Hall.

10. **Klema, V.C.** and **Laub, A.L., 1980,** "Singular Value Decomposition: Its Computation and Some Applications", IEEE Trans. on Automatic Control, Vol. AC-25, No. 2, pp. 164-176.