

Modeling, Simulation, Animation, and Control for a Single Robotic Manipulator

Richard P. Metzger Jr ^{*} Armando A. Rodriguez [†]

Department of Electrical Engineering
Center for System Science and Engineering
Arizona State University
Tempe, AZ 85287-7606

October 1, 1996

Keywords: Robot, PUMA, interface, simulation module, graphics module, education module.

Abstract

This paper describes a Windows/C₊₊-based pc environment for simulating and animating a PUMA 560 robot under automatic control. The program consists of four modules: (i) a program user-interface module, (ii) a simulation module, (iii) a graphics/animation module, and (iv) an education module. The program user interface module allows the user to interact with the program. Specifically, this module permits the user to select model parameters, control law structure, control law parameters, reference commands, disturbances, initial conditions, integration routine, and integration routine parameters. The simulation module is responsible for generating a numerical solution for the closed loop dynamics. The graphics/animation module updates plots and animation on the screen using data generated by the simulation module. The education module provides the user with interactive lessons designed to teach fundamental systems and controls concepts. Designed to communicate with MATLAB, it is shown how this

environment may be used to analyze many “what if” scenarios - observing system dynamics both graphically and through animation, making the environment a very useful tool for teaching control system design concepts.

1 Introduction

The recent revolution in personal computing has seen computing power soar while prices have dropped. As a result, powerful pc's are now commonly available. Given this, it is now possible to exploit the new technology to significantly enhance the way in which systems and controls education is delivered. Because of today's computing power [8], for example, complex simulations - not long ago considered formidable - can now be easily performed. Recent trends in computer speed have also made fast animation of dynamical systems a possibility [4]. In short, today's pc technology now permits the development of new interactive graphical visualization environments which could revolutionize the teaching of systems and controls. This opportunity, has permitted the authors to develop such environments [6], [7], [16], [22], [23], [24], [25]. This paper describes the development of an interactive robot manipulator environment. It is shown how many fundamental system and control concepts may be readily observed and understood with such an environment.

The remainder of this paper is organized as follows. Section 2 discusses the mathematical model describing the PUMA 560 system. In Section 3, features within the robot arm environment are described. Sec-

^{*}Currently completing a Master of Science in EE.

[†]This research has been supported, in part, by the National Science Foundation (NSF) through the Coalition to Increase Minority Degrees (CIMD), the ASU Center for Innovation in Engineering Education (CIEE), the Boeing A.D. Welliver Faculty Fellowship Program, and the Intel Corporation. For additional information please contact aar@asu.edu.

tion 4 then demonstrates the utility of the environment as an educational tool. Finally, Section 5 summarizes the paper and presents directions for future research.

2 System Description: Models

The robot arm environment permits the study of a Unimate PUMA 560 robot. The mathematical models used to describe this system are now discussed.

PUMA 560 Robot: Nonlinear Model. In this work, a 12th order nonlinear dynamical model, relating six joint angles to six joint torques, is used to describe the motion of the PUMA 560. Lagrangian techniques were used to derive the model [28]. While this model is appropriate for simulation, it is too complex for analysis, design, and the introduction of basic systems and controls concepts.

Single Free Joint: Nonlinear Model. Given the above, we decided to focus on a simplified PUMA 560 configuration in which all joints, but one, are assumed to be locked. This simplifies the system dynamics considerably. The resulting dynamical model may be described by the following 2nd order nonlinear ordinary differential equation:

$$J\ddot{\theta} + B\dot{\theta} - mgl\sin\theta = \tau + \tau_o \quad (1)$$

where θ denotes the angle which the free joint makes with respect to the vertical and is measured in radians, τ_o denotes an applied equilibrium joint torque and is measured in Newton-meters, τ denotes an applied joint torque deviation with respect to τ_o and is measured in Newton-meters, J denotes the effective moment of inertia and is measured in kilogram-meters², B denotes the effective joint friction and is measured in kilogram-meters² per second, m denotes the effective mass of the rotating elements and is measured in kilograms, l denotes the distance between the effective center of mass and the axis of rotation and is measured in meters, and g denotes the acceleration due to gravity measured in meters per second².

While this model does not capture motor dynamics or link flexing, it is appropriate to illustrate fundamental control system design concepts. Of interest here is maintaining the free joint angular orientation given a desired angle or angle command. Given this, it follows that the joint angle is a critical variable.

The end-effector position is another important variable.

Single Free Joint: Linear Model. When Equation 1 is linearized about the equilibrium point ($\theta = 0, \tau = -\tau_o$), one obtains the following 2nd order linear ordinary differential equation:

$$J\ddot{\theta} + B\dot{\theta} - mgl\theta = \tau \quad (2)$$

Other, more complex, robot arm configurations and models will be considered in the future (e.g. two link robot arm). It should be noted that the simulator described contains a full six degree of freedom nonlinear model, a nonlinear sliding mode controller, and a trajectory generator [5], [26], [28]. These however are beyond the scope of this paper.

3 Description of Environment

The robot arm environment is written in Windows/C++ [1], [3], [10], [12], [20], [29]. It consists of four modules:

- a program user interface (PUI) module,
- a simulation module,
- a graphics/animation module, and
- an education module.

The environment was divided into modules to facilitate updates and enhancements created by different programmers.

Program User Interface Module. The PUI provides an interface between a user and the robot arm environment program. Written in Windows/C++, the environment provides pull-down menus which allow the user to modify critical system parameters in real-time. These parameters include, for example, mass properties, moments of inertias, joint friction, etc. The user may also select amongst different control laws (e.g. PID, state feedback, sliding mode, and LQR/LQG). Initial conditions, reference commands, disturbances, and integration routines (e.g. Euler, Runge-Kutta, etc.) may be selected by the user from a menu. Other menu options for data storage and plotting exist. Data storage routines automatically format saved simulation data for use within other environment modules as well as external programs

(e.g. MathWork's MATLAB and Microsoft's Excel). User-selected variables saved from an earlier simulation may be plotted against current simulation data. Multimedia lessons, which use audio and video will be accessible through the PUI.

Simulation Module. The main purpose of the simulation module is to accurately solve the appropriate set of ordinary differential equations. The simulation module contains routines required by the trajectory generator, different robot configurations, control laws, integration methods, and data storage routines. The robot arm environment's models, trajectory generator, numerical integration, and control law routines are included in this module.

Graphics/Animation Module. The main purpose of the graphics module is to use data provided by the simulation module to update graphics (i.e. plots) and animations on the screen. Data and plots are displayed within *child windows* [11], [19]. Animation is created using one of two methods: *polyhedral approximations* of robot links or high quality bitmaps of the robot.

- *Polyhedral Approximation.* The polyhedral approximations for robot links are created using modified techniques and routines from a C++ animation toolkit [3]. Polyhedrals are used to represent each link of the robot in an animation of simulation data from the 12th order nonlinear model or as required by the education module's lessons.
- *Bitmaps.* High quality bitmaps are created with Corel's CorelDRAW and CorelPHOTO-PAINT¹. Bitmaps are used for animations of simulation data from simplified robot configurations. Therefore, bitmaps are primarily used for the education module's lessons.

Education Module. The education module contains routines which implement interactive multimedia lessons. The lessons use text, audio, video, and animation to convey concepts. Users answer questions via interactive menus. Correct answers are supported with a multimedia explanation - includ-

¹CorelDraw and CorelPHOTO-PAINT are trademarks of Corel Corp. CorelDraw is a general purpose drawing/graphics manipulation utility. CorelPHOTO-PAINT is a general purpose image processing utility.

ing "supportive audio." Incorrect answers are followed up by hints, partial explanations, and additional chances.

4 Educational Utility

The utility of the environment as an educational tool is now demonstrated. Several examples of the single robot link system will be presented. Additional examples will be presented at the 1997 ICSEE.

The model parameters J , B , and l were found by solving a linear least squares problem using simulation data obtained from the 12th order nonlinear PUMA model. The model parameters used for the single link robot system were: $m = 27.29$ kg, $J = 1.9852kg - m^2$, $B = 2.0267 \frac{kg-m}{sec}$, $l = 0.0293$ m, and $g = 9.8 \frac{m}{sec^2}$. The controller used was as follows:

$$\tau = K_1 e \quad (3)$$

$$e = r - K_2 \theta \quad (4)$$

$$K_1(s) = k_1 \left[\frac{s+a}{s} \right] \quad (5)$$

$$K_2(s) = k_2(s+b) \left[\frac{100}{s+100} \right] \quad (6)$$

where $k_1 = 9.9260$, $k_2 = 1.1082$,

$$a = \frac{B}{2J} + \sqrt{\frac{B^2}{4J^2} + \frac{mgl}{J}} = 2.5617, \quad (7)$$

and $b = 0.9024$ were used as design parameters. The pole at $s = -100$ in K_2 was introduced only to make K_2 proper.

With a second order plant and "essentially" a first order compensation scheme, three dominant closed loop poles result. The design parameters were selected such that the dominant closed loop poles were at

$$s = -2 \pm j1, -2.5617. \quad (8)$$

Here, the last root is a pole of the plant linearized about the equilibrium point ($\theta = 0, \tau = -\tau_o$). Selecting the zero of K_1 to be $s = -a = -2.5617$ fixes one of the closed loop poles at $s = -2.5617$.

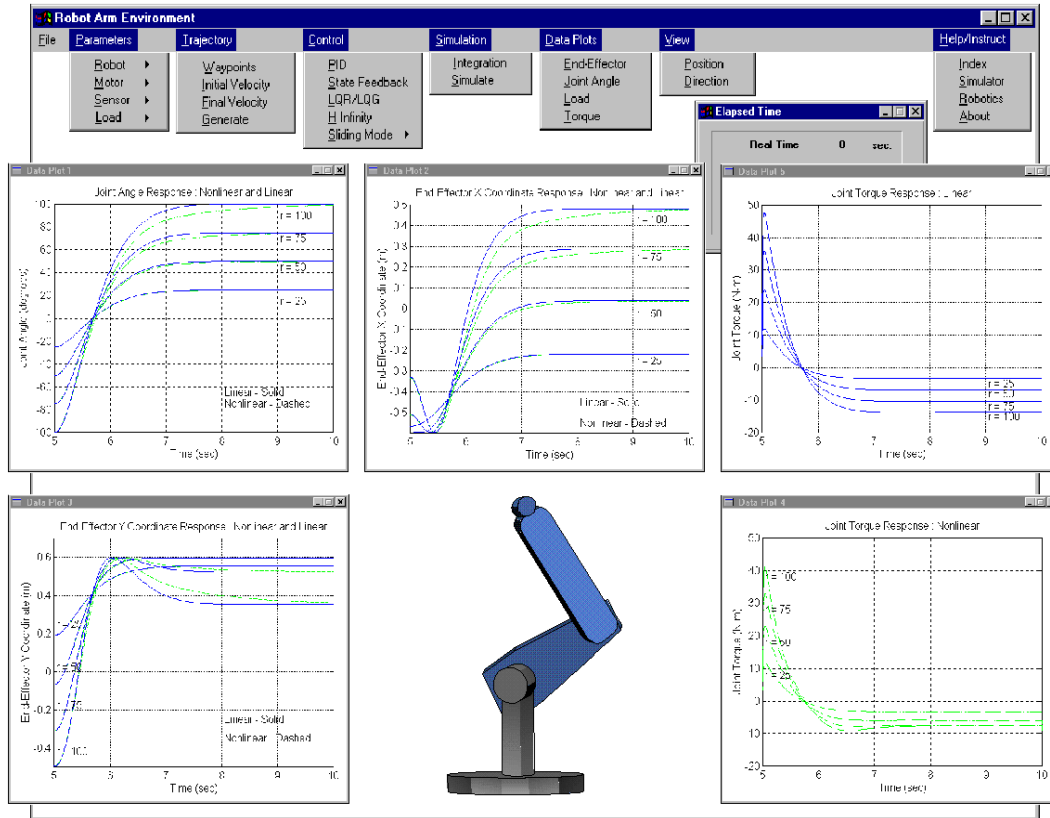


Figure 1: Single Link Robot Command Following ($r = 25, 50, 75, 100$ degrees)

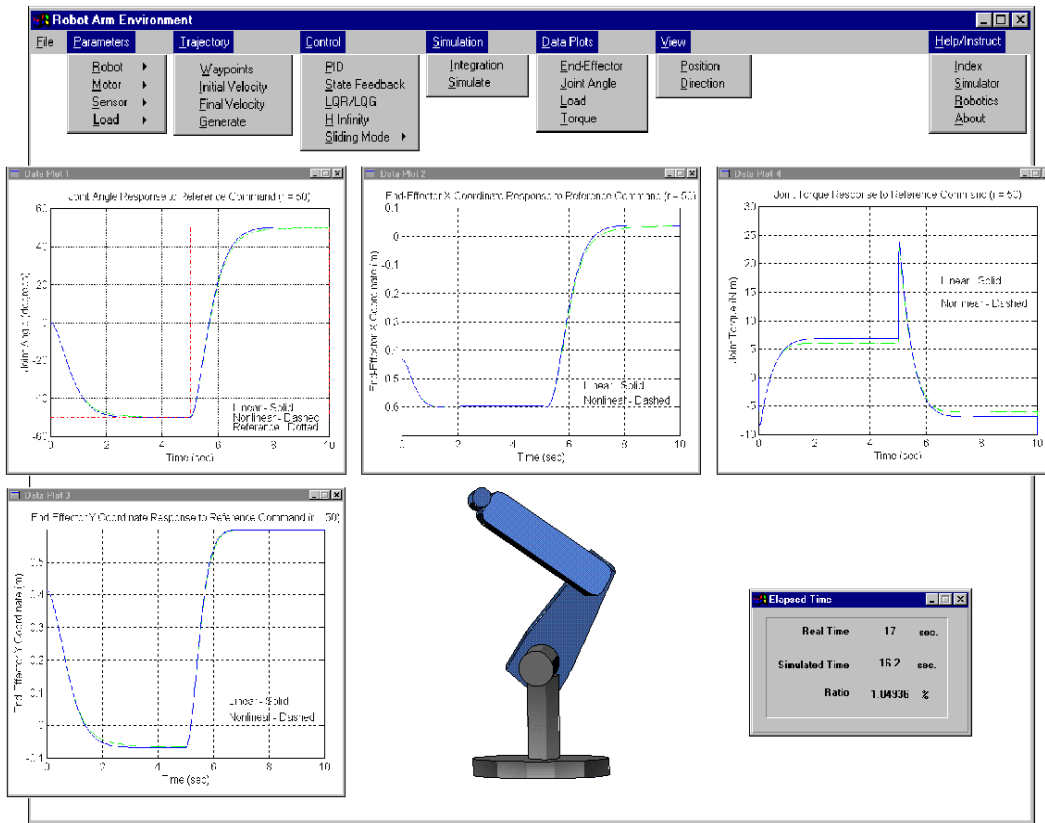


Figure 2: Single Link Robot Command Following ($r = 50$ degrees)

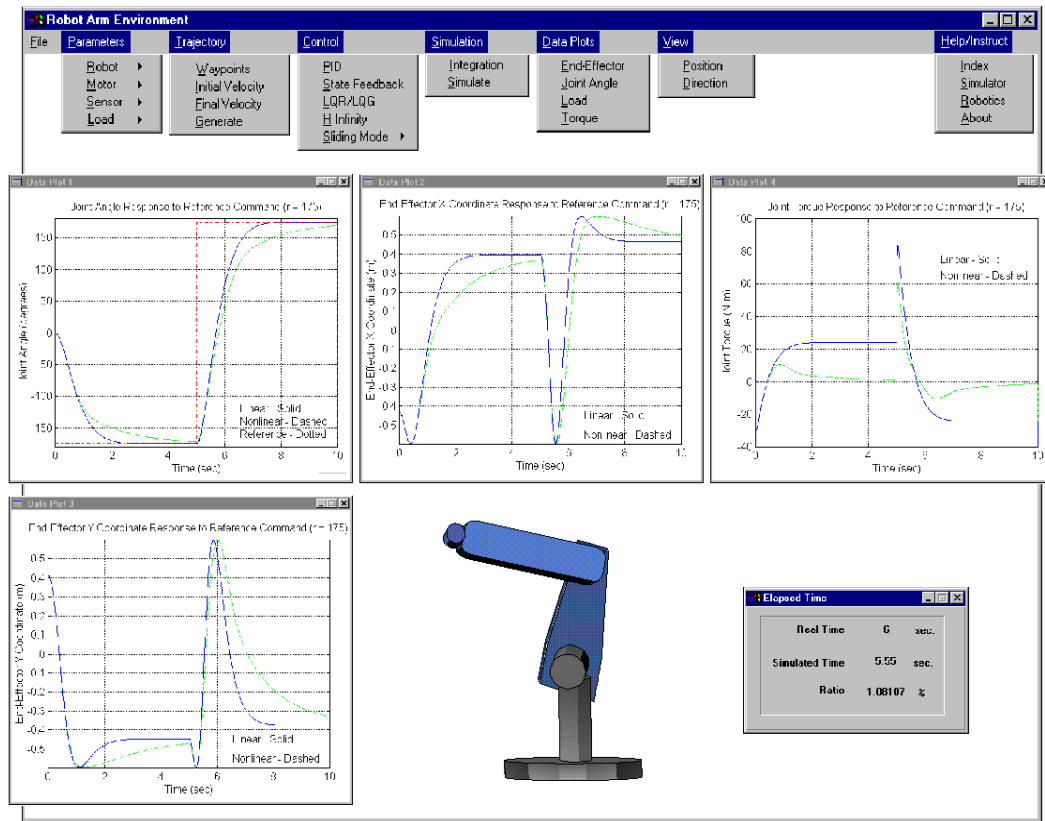


Figure 3: Single Link Robot Command Following ($r = 175$ degrees)

Given this, one then obtains the following design equation for the other compensator parameters:

$$s^2 + \left[\frac{k_1 k_2}{J} + \frac{B}{2J} - \sqrt{\frac{B^2}{4J^2} + \frac{mgl}{J}} \right] s + \frac{k_1 k_2 b}{J} = s^2 + 4s + 5 \quad (9)$$

Figure 1 shows a screen dump of the environment for different reference commands: $r = 25, 50, 75, 100$ degrees. Both linear and nonlinear plots are given for θ , x , y , and τ . The plots show that the linear and nonlinear models agree quite well for the smaller reference commands. As the reference commands get larger, the maximum error between the nonlinear and linear models for the joint angles approaches 10 degrees. The end-effector coordinates x and y each approach a maximum error of approximately 8 centimeters between the models. The maximum error between the nonlinear and linear joint torque is approximately 5 Newton-meters. This error is caused by the small angle approximation ($\sin\theta \approx \theta$) used in deriving the linear model. Because of this, the linear model doesn't show the reduced torque requirement to keep the arm at $\theta \approx 180$ degrees (i.e. almost downward).

Figure 2 shows a screen dump of the environment for a reference command of $r = 50$ degrees. Both linear and nonlinear plots are given for θ , x , y , and τ . The linear data closely approximates the nonlinear data in the θ , x , y , and τ plots for the $r = 50$ degrees reference command.

Figure 3 shows a screen dump of the environment for a reference command of $r = 175$ degrees. Once again, both linear and nonlinear plots are given for θ , x , y , and τ plots. The error between the models improves for each θ , x , and y plots as time increases and the steady state values are approached. However, the error between the models in the τ plot does not improve. This error is the result of the small angle assumption made in deriving the linear model.

5 Summary and Directions

The single robotic manipulator environment described in this paper provides a flexible educational tool for teaching control system design concepts. Currently, more complex models and control laws (e.g. motors, saturations, anti-windup, etc.) are being implemented. JAVA (<http://java.sun.com>) and VRML implementations are also being developed [14], [15]. Future work will focus on the continued development of lessons for the single link robot and development of the double link robot with appropriate interactive lessons. The goal is to obtain an interactive multimedia environment which can be used to build open-ended problem solving skills while encouraging exploration and self discovery.

References

- [1] L. Adams, *Supercharged C++ Graphics*, McGraw Hill, 1992, pp. 207-223.
- [2] B.A.A. Antao, A.J. Brodersen, J.R. Bourne, and J.R. Cantwell, "Building Intelligent Tutorial Systems for Teaching Simulation in Engineering Education," *IEEE Transactions on Education*, Vol. 35, No. 1, February 1992, pp. 50-56.
- [3] R. E. Bradford, *Real-Time Animation Toolkit in C++*, J. Wiley & Sons Inc., New York, NY, 1995.
- [4] R. Braham, "Math and visualization: New Tools, New Frontiers," *IEEE Spectrum*, November 1992, pp. 17-36.
- [5] J. Côté, C.M. Gosselin, and D. Laurendeau, "Generalized Inverse Kinematics Functions for Puma Manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, June 1995, pp. 404-408.
- [6] K.J. Elliott and A.A. Rodriguez, "A Graphical Dual Robot Simulation for the PC," *Proceedings of the 1995 International Conference on Simulation in Engineering Education*, Las Vegas, NV, January 15-18, 1995, pp. 77-81.
- [7] M.F. DeHerrera and A.A. Rodriguez, "Trying to 'Shoot' an Evasive Monkey: A Tool for Designing and Evaluating Adaptive Learning Algorithms," *Proceedings of the 1996 International Conference on Simulation in Engineering Education*, San Diego, CA, January 14-17, 1996, pp. 31-36.
- [8] L. Geppert, "The New Contenders," *IEEE SPECTRUM: Fast and Powerful*, December 1993.
- [9] D. Hearn and M.P. Baker, *Computer Graphics*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
- [10] L. Heing, *Advanced Graphics Programming Using C/C++*, John Wiley & Sons Inc., 1993, Chapter 1.
- [11] L. Heiny, *Windows Graphics Programming with Borland C++*, John Wiley & Sons Inc., New York, NY, 1992.
- [12] S. Holzner, *Borland C++ Windows Programming*, Brady Publishing, Indianapolis, IN, 3rd Edition, 1994.
- [13] E. Kreyszig, *Advanced Engineering Mathematics*, Fifth Edition, J Wiley & Sons Inc., New York, NY, pp.842-847, 1983.
- [14] L. Lemay and C.L. Perkins, *Teach Yourself JAVA in 21 Days*, Sams Net Publishing, IN, 1996.
- [15] S.N. Matsuba and B. Roehl, *Special Edition Using VRML*, Que Corp, IN, 1996.
- [16] R.P. Metzger, K.J. Elliott, and A.A. Rodriguez, "Modelling, Analysis, and Graphical Visualization of a Dual Robot Arm System: A PC Based Environment," *Proceedings of the 1996 International Conference on Simulation in Engineering Education*, San Diego, CA, January 14-17, 1996, pp. 175-180.
- [17] R.P. Metzger and A.A. Rodriguez, "Modeling, Simulation, Animation, and Control for the PUMA 560 Robot: An Interactive Education Environment", in preparation.
- [18] R.P. Metzger, "Modeling, Simulation, Animation, and Real-Time Control of Dynamical System: Interactive Environments," Masters Thesis, Department of Electrical Engineering, Arizona State University, in preparation.
- [19] S. Oualline, *Windows Programming with Borland C++*, M&T Books, New York, NY, 1993.
- [20] S. Potts and T. S. Monk, *Borland C++ 4 By Example*, Que Corporation, Indianapolis, IN, 1994.
- [21] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C*, Second Edition. Cambridge University Press, New York, NY, 1994.
- [22] A.A. Rodriguez and R. Aguilar, "Graphical Visualization of Missile-Target Air-to-Air Engagements: An Educational Tool for Designing and Evaluating Missile Guidance and Control Systems," *Journal of Computer Applications in Engineering Education*, Vol. 3, No. 1, 1995, pp. 5-20.
- [23] A.A. Rodriguez and M.F. DeHerrera, "Modeling, Simulation, and Graphical Visualization of a Twin Lift Helicopter System Under Automatic Control: An Educational Tool," to appear in the Proceedings of the 1996 Conference on *Frontiers In Education*, Salt Lake City, Utah, Nov 6-9, 1996.
- [24] M. Sonne, A.A. Rodriguez, "A PC-based Graphics System for the Evaluation of Missile Guidance and Control Laws", Proceedings of the American Control Conference. Baltimore, MD. June 29-July 1, 1994, pp. 2726-2730.
- [25] M. Sonne, A.A. Rodriguez, "PC's in the Design and Evaluation of Guidance and Control Systems for Missiles," Proceedings of the ICSEE. Tempe, AZ. Jan 21-23, 1994, pp. 329-334.
- [26] M.W. Spong, and M. Vidyasagar, *Robot Dynamics and Control*, J. Wiley & Sons Inc., New York NY, 1989.
- [27] T.J. Tarn, A.K. Bejczy, and X. Yun, "Design of Dynamic Control of two cooperating robot arms: Closed chain formulation," *Proceedings IEEE International Conference Robotics and Automation*, Raleigh, NC, 1987, pp. 1242-1247.
- [28] T.J. Tarn, A.K. Bejczy, X. Yun, and X. Ding, "Dynamic Equations for the Six-Link Puma 560 Robot Arm," Laboratory Report SSM-RL-86-05, Department of Systems Science and Mathematics, Washington University, St. Louis, MO, Oct 1986.
- [29] P. Yao, *Borland C++ 4.0 Programming for Windows*, Random House Inc., New York, NY, 1994.