2004 IEEE International Symposium on
Computer Aided Control Systems Design
Taipei, Taiwan, September 2-4, 2004

The Design of a Control System Testbed for a 2-DOF Robot

Frank S. Cheng and Lin Zhao

Central Michigan University
Mount Pleasant, Michigan, 48859, USA

Abstract – **The common challenges in the real-time control system design are the control system integration, modeling and analysis, program coding, and real-time data communication. To address the solutions to these challenges, this paper presents the design of a control system testbed for a two degrees of freedom (2-DOF) robot. The design methodology supports the designer to (1) apply the theories of robotics and feedback control systems for feasible robot's path control solutions, (2) model, simulate and analyze the path control function using Simulink software, and (3) implement the control solution as a real-time PC-based robot control system.**

I. INTRODUCTION

An industrial robot is a manipulator that has multiple powered joints. The end of the manipulator is the robot's tool center point (TCP). For most industrial robots, the actual TCP path is formed as the TCP moves from points to points within the robot's working space. The TCP movement between two points is described by a motion profile that specifies the TCP's initial and ending points, speed, and trajectory.

Developing the robot control system for the robot's TCP motion and path control involves the applications of robotics and feedback control systems theories [1], [2]. For example, in the robotics theory, two measuring systems must be established for the robot control system to control the TCP motion. The robot's world measuring system is used for measuring the robot's TCP in the robot base Cartesian coordinate system. The joint measuring system is used for measuring the joint positions of the robot. For a given TCP point measured in the world measuring system, the robot control system must know the corresponding joint values in order to position the robot's joints for the given TCP point. This requires developing the robot kinematics that establishes the mathematical relationship between the joint measurement and world measurement for all feasible TCP points. The closed-loop control system also must be designed, so that the controller is able to perform the specified control functions to the joint's variables such as position and speed based upon real-time sensor measurements of the joint. In addition, the robot control system must have a motion control strategy that specifies how to coordinate the movement of all robot joints as the TCP moves from a point to another. The use of the different motion control strategies results in the different TCP trajectories.

This paper presents the design of a control system testbed for a two degrees of freedom (2-DOF) robot. The project aims at enhancing the designers' knowledge and understanding about the design issue of a real-time robot control system. This includes the control system integration, modeling and simulation, controller coding and prototyping. The developed testbed addresses the design solutions in three parts. First, the robot's TCP motion and path control solutions are developed based upon the robotics and feedback control systems theories. Second, the performance of the control system model is analyzed by using MATLAB and Simulink software. Finally, the Simulink model-based control solution is implemented as the real-time C-code controller by using the controller prototype tool. As a result, the designed robot control system is able to control the actual robot joints and move the robot's TCP along any specified path within the robot working space. In the paper, the functions of the hardware and software components used in the 2-DOF robot control system testbed are introduced in Section II. The configurations of the 2-DOF robot mechanism and control system are described in Section III. Section IV presents the methods of implementing the functions of the motion controller and system graphical user interface (GUI). The conclusion of the study is given in Section V.

0-7803-8636-1/04/$20.00 ©2004 IEEE

II. SYSTEM HARDWARE AND SOFTWARE

The 2-DOF robot control system testbed is developed by using the Quanser's rotary motion control system [3]. The Quanser rotary system consists of a power module, a DC motor system with the built-in position and speed sensors, the Quanser's PCI data acquisition board and WinCon software. The Quanser's WinCon software is a real-time Window 2000 application supported by the commercial software packages including MATLAB, Simulink and Real Time Workshop (RWT) from the MathWorks, and Visual C++ from the Microsoft. Three functions of the WinCon software are used in the 2-DOF robot control system testbed. They are: (1) the conversion of a Simulink control model into the C source code by using RWT; (2) the compilation and linking of the generated C code for a real-time WinCon controller file; and (3) the data communication between the WinCon real-time controller and the MATLAB files.

For building the WinCon real-time controller in C-code from a Simulink model, the user must choose the RWT's configuration option for WinCon software to 'make' the system target file. The Simulink parameter called 'step-size' or 'sampling period' must also be set for the WinCon real-time controller to calculate the control solutions. The WinCon software only supports fixed 'step-size' (or 'sampling period') and typically uses the ode1 (Euler) method for solving continuous-time control solutions. With the specified simulation runtime and 'step-size', the user can determine the number of sampling data elements for an output variable in a Simulink model.

The Quanser PCI board and WinCon data communication commands support the functions of real-time data acquisition and control functions in the 2-DOF robot system. The PCI board allows the WinCon controller to interface to a variety of devices via analog and digital signals, as well as encoders. The WinCon software allows the user to use the Simulink-based interface driver block to collect real-time data from either a running Simulink model or the WinCon controller. For example, the real-time variable data can be collected and plotted in *WinCon Scope* (or *WinCon X-Y Graph*) on-line. However, due to the facts that the real-time data being saved from a *WinCon Scope* (*or WinCon X-Y Graph*) is not decimated, and the sampling frequency used by the real-time C-code is also the frequency that the *WinCon Scope* (or *X-Y Graph*) buffer uses to collect the real-time data, the user may first use the *Update/Buffer* option from the *WinCon Scope or X-Y Graph* menu bar to select the buffer duration long enough to accommodate the desired data acquisition needs, and then save the buffer data to a MATLAB data file or to the MATLAB workspace. This can be don by either using WinCon Simulink blocks '*To File*' and '*To Workspace*' in the Simulink model, or manually clicking *File/Save* item from the *WinCon Scope* and *X-Y Graph* menu bar.

There are three ways of changing the parameters of a running WinCon C-code controller. In a Simulink diagram, once the parameter of a block is changed, the new parameter will be automatically downloaded to the WinCon controller and executed as soon as the user clicks the 'Apply' button in the Simulink window. The user can also change Simulink model parameters in MATLAB workspace and manually download the modified parameters into the real-time controller by selecting the '*Edit/Update*' Diagram item from the Simulink window. In addition, the WinCon control panel also allows the user to directly change controllers parameter without the support of a running Simulink model.

The WinCon MATLAB Scripting commands allow the user to interact with the WinCon controller from the MATLAB workspace and MATLAB programs. These commands give the user the maximum flexibility in data communication and control. With these commands, the user is able to open a Simulink model, generate the real-time code from the opened Simulink model, start and stop the execution of the WinCon controller, upload the modified model parameters to the WinCon controller, and save model data into the MATLAB workspace and data file.

III. CONTROL SYSTEM FUNCTIONS

The developed 2-DOF robot system testbed is mechanically configured with two Quanser motor systems that are connected by four equal-length links as shown in Figure 1 and Figure 2. In Figure 2, P_x and P_y denote the x and y coordinates of a robot's TCP position, θ_1 and θ_2 denote the corresponding two motor-joint angles (i.e., joint 1 and joint 2) for the given TCP position. The initial robot's arm position is defined by the zero joint positions of joint 1 and joint 2. The zero positions for joint 1 and joint 2 are when the joint's output link is horizontal and vertical, respectively. The positive joint rotation is defined as the counterclockwise rotation. The developed robot inverse kinematics allows the robot control system to determine values of the two powered joints θ_1 and θ_2 for a given TCP position in P_x and P_y. With this four-joint configuration, the two motor-joints (i.e., joint 1 and joint 2) are able to move the robot's TCP within its working space in X-Y plane.

Four components are used in the 2-DOF robot control system testbed to achieve the robot's TCP path control function. They are the motion controller, motor system, Simulink models, and graphical user interface (GUI). Figure 3 shows the data communication requirement in the system. The system GUI allows the user to specify an input TCP path to the motion controller. The user can use the GUI 'path selection' function to specify a standard TCP

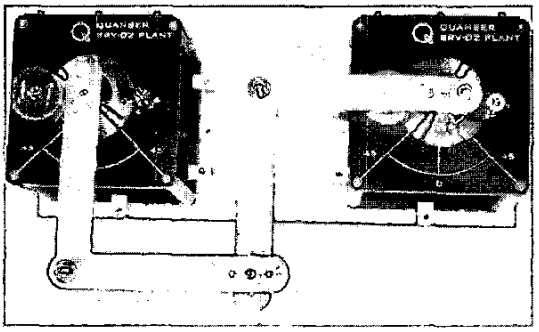

Fig. 1. The structure of the 2-DOF robot

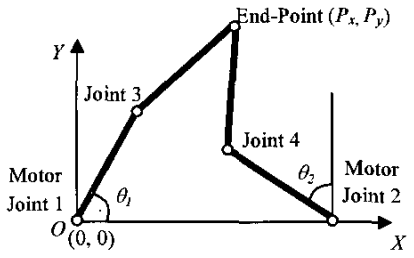

Fig. 2. Illustration of the 2-DOF robot kinematics

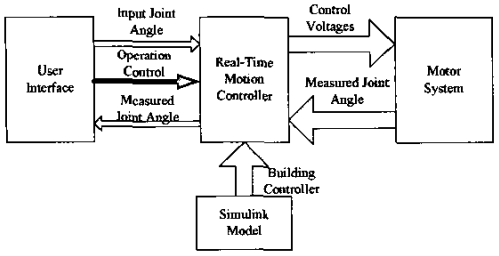

Fig. 3. System components' data communication

path such as a straight or a circle line in the robot's Cartesian system. In this case, the robot's inverse kinematics is used to calculate the corresponding two motor-joint angels for each interpolated point on the path. The user can also teach a TCP path by using the 'path teaching' function. During teaching the path, the user physically moves the robot's TCP, the position sensor on the motor-joints measures the corresponding motor-joint angles, and the WinCon software saves the measured data in real-time. Then, the robot's forward kinematics is used to transform the measured motor-joint angles to the corresponding x and y coordinates for representing the taught input TCP path. The GUI also allows the user to perform the system operations to the motion controller such as Calibration, Start, and Stop, etc. The GUI Calibration command allows the 2-DOF robot to find the robot's home position that is defined by the zero positions of the two motor-joints. After the calibration, the position sensors on the motor-joints are ready to measure the motor-joint positions. The GUI Start command starts the motion controller to move the two motor-joints for generating the TCP path. The GUI Stop command performs the system E-stop function for stopping the actual robot motion.

The motion controller is a MIMO control system in the sense that it deals with multiple joint inputs and outputs in real-time. The motion controller uses two closed-loop position control systems to control the positions of two motor-joints in real-time, respectively. The 2-DOF control system testbed integrates the Simulink software through the WinCon software. The user is able to model and simulate the closed-loop control system for the desired system response, and implement the control solution as the WinCon real-time controller.

IV. SYSTEM IMPLEMENTATION AND RESULTS

Figure 4 shows the Simulink model of the closed-loop position control system used by the motion controller to control the position of the motor-joint in the motor system. In the model, the input variable is the required motor-joint angle θ. The output variable is the actual motor-joint position measured by an incremental encoder. The controller compares the actual joint angle to the input angle and performs the designed control calculation to the angle difference for generating the control output signal V_m(V). The control output signal from the controller is then amplified through power module and sent to the motor system for driving the joint.

Figure 5 shows the Simulink model for the motor sub-system block in Figure 4. Supported by the Quanser PCI board, the WinCon Simulink interface block 'Analog Output' in Figure 5 allows the control output signal V_m(V) to be sent to the power module in the motor system. The

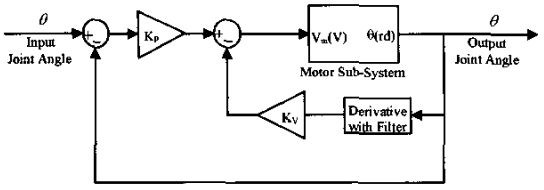

Fig. 4. The Simulink model of the position controller

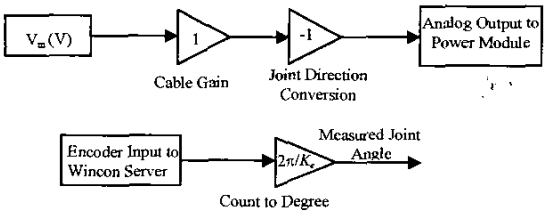

Fig. 5. Simulink model of the Quanser motor interface

interface block 'Encoder Input' allows the model to receive the encoder measurement of the motor-joint angle from the motor system in real-time. The resolution of the encoder is defined by the encoder constant K_e. In the model, the encoder measurement is converted from pulses to radians through the encoder constant K_e (K_e = 4096 pulse/radian).

In Figure 4, the motion controller uses a PD (proportional-plus-derivative) control function to perform the control calculation. The function is implemented with a proportional gain K_p in the forward loop and a derivative gain K_v with a filter in the feedback loop. The proportional gain amplifies the position error signal for a quicker joint position response, while the derivative gain regulates the joint's speed for reducing the position overshoot. Through conducting the simulation in Simulink, the K_p and K_v, values are tuned to satisfy the transient and steady response of the actual motor system. Figure 6*a* and Figure 6*b* show the step response of the position control system via the different K_p and K_v values. For example, the actual valves of K_p and K_v used by the motion controller in the 2-DOF robot system are 32 and 0.4, respectively, as shown in Figure 6*b*. As a result, the motor position control system is stable. The transient response time of the system is about 0.05 second and the steady error is zero radians.

The 2-DOF robot system GUI is implemented as a MATLAB GUI as shown in Figure 7, where the function buttons perform the data acquisition and control to the motion controller through MATLAB data structure and callback functions.

Figure 8 shows the GUI overall control function to the motion controller. During the system initialization, the GUI initializes its output data variables to the motion controller. The variables include the motor-joint angles θ_1 and θ_2, and the length of the execution time for the motion controller. Following is the robot calibration procedure in which the GUI starts the motion controller for moving the two motor-joints to their pre-defined zero positions. After the robot calibration, the user uses the 'Path Selection' pull-down

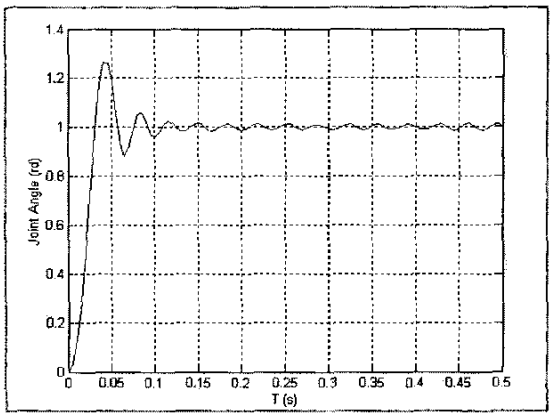

Fig. 6*a*. The controller parameters K_p = 70 and K_v = 0.3

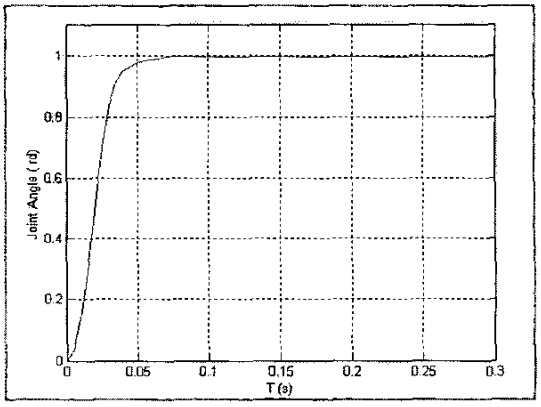

Fig. 6*b*. The controller parameters K_p = 32 and K_v = 0.4

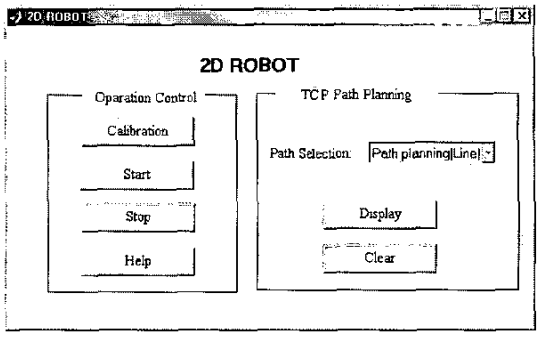

Fig. 7. The MATLAB GUI in the 2-DOF robot system

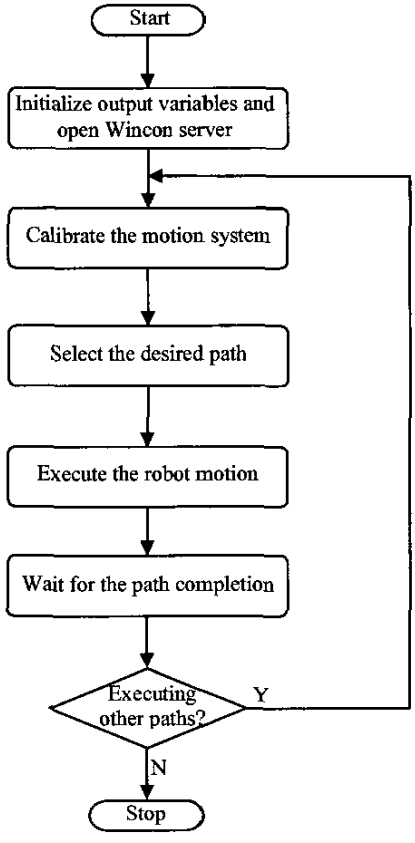

Fig. 8. The GUI overall control function

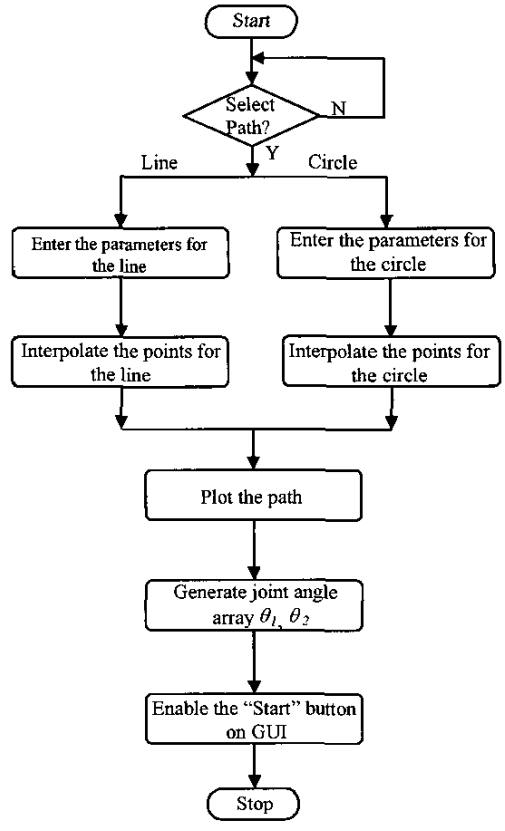

Fig. 9. The GUI Path Selection button function

menu to specify the TCP path as either a straight or a circle line. Figure 10 shows the implementation of GUI Path Selection function. Based upon the selected path type and the specified path parameters, the GUI first interpolates average 50 points for the path and generates the corresponding x and y coordinates for each interpolated point. Then, it graphically displays the input TCP path by plotting all interpolated points. The GUI also uses the inverse kinematics of the 2-DOF robot to convert the x and y coordinates of a TCP point into the corresponding motor-joint angles θ_1 and θ_2, and saves the calculated joint angles of all points into the MATLAB data file. After the path selection is done, the GUI Start button is enabled and the user is able to use the Start button to move the 2-DOF robot for generating the actual TCP path.

The implementation of the GUI Start button is shown in Figure 10. After the Start button is clicked, the GUI uses the WinCon MATLAB Scripting command *wc_start* to start the motion controller, and uses the WinCon MATLAB Scripting command *wc_update* to send the input motor-joint angles θ_1 and θ_2 to the motion controller for executing the joint motion. As the motion controller moves the actual TCP to each point, the encoder measures the actual motor-joint angle and the *WinCon Scope* buffer stores the measured data. When the robot completes the TCP path, the GUI is to do the followings: (1) using the WinCon MATLAB Scripting command *wc_saveplot* to upload all saved motor-joint angels from the *WinCon Scope* buffer and save them in the MATLAB data file, (2) using the WinCon MATLAB Scripting command *wc_load* to load the MATLAB data file to the MATLAB workspace, (3) performing the forward kinematics calculation to convert the motor-joint angles in the MATLAB workspace into the corresponding x and y coordinates of the TCP points, and (4) enabling the GUI Display button.

Clicking the GUI Display button displays the actual robot TCP path in x and y coordinates. Figure 11*a* and Figure 11*b* show the results of the actual TCP path via the

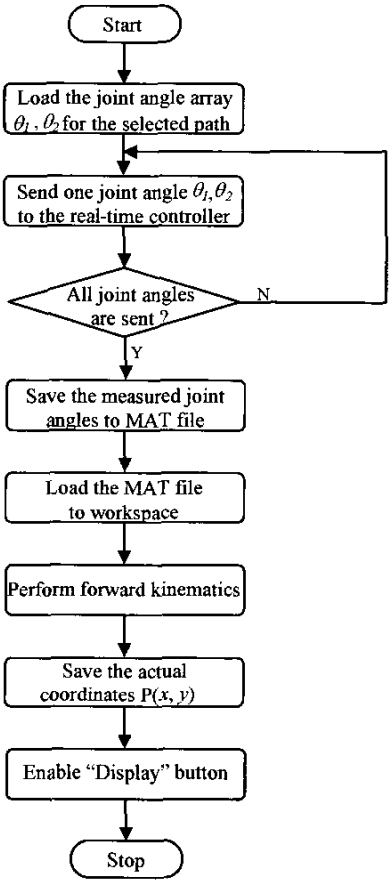

Fig. 10. The GUI Start button function

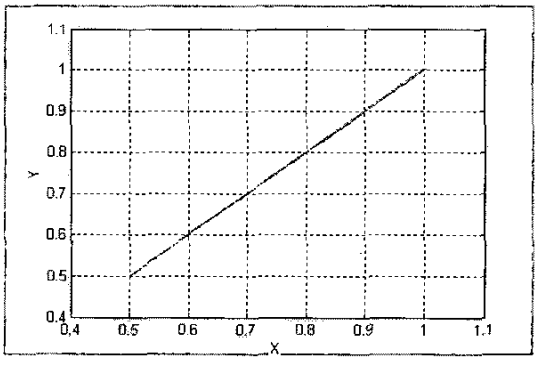

Fig. 11*a*. The actual TCP path via a straight input path

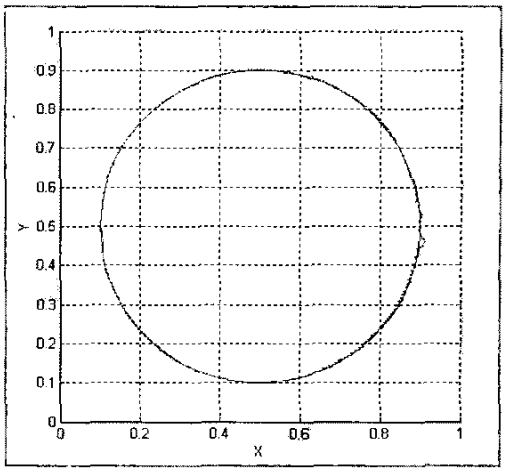

Fig. 11*b*. The actual TCP path via a circular input path

input straight and circular TCP paths. The results demonstrate that the developed 2-DOF robot control system is able to move the robot's TCP along the input TCP path with a satisfied accuracy. Interpolating more points to the section where the path inaccuracy occurs allows the robot control system generating a more accurate TCP path.

V. CONCLUSION

The design project for a 2-DOF robot control system testbed presented in this paper demonstrates an integrated method of designing the robot path control system using the DC servo control system and control design software such as MATLAB, Simulink, and Quanser WinCon. The designed 2-DOF robot control system is able to control the robot TCP to follow the user specified paths with accuracy.

REFERENCES

[1] C. Phillips, *Feedback Control Systems*, 4th ed. Prentice Hall, 2000.

[2] Niku, *Introduction to Robotics: Analysis, Systems, and Applications*, Prentice Hall, 2001.

[3] "WinCon 3.3 User's Guide," Quanser Consulting, Inc.

[4] "MATLAB 6.1 User Manual," The Mathworks, Inc.