Robotica (2005) volume 23, pp. 247–256. © 2005 Cambridge University Press
doi:10.1017/S0263574704000785 Printed in the United Kingdom

A software tool for automating the design of robot fuzzy force controllers

K. Burn*, G. Home*, M. Short† and R. Bicker‡

(Received in Final Form: June 18, 2004)

SUMMARY
This paper describes a software tool to automate a design method for robotic fuzzy force control. The original method was developed to ensure robust and stable force control in situations where environmental stiffness at the robot/task interface is unknown, obviating the use of fixed-gain controllers. It did, however, involve a manual design process requiring significant knowledge of control theory and fuzzy logic. This process has been automated in the form of a Windows-based application, requiring minimal user inputs and incorporating an automatic tuning technique for improved performance in the final controller application. Results obtained from an experimental robot are presented.

KEYWORDS: Robotics; Force control; Fuzzy control; Software automation.

1. INTRODUCTION

Traditionally, most industrial robots are designed to allow accurate and repeatable control of the position and velocity of the tooling at the device's end effector. However, if robots are to perform complex tasks in a wider range of applications in the future, it will be essential to accurately control forces and torques at the end effector/task interface. In addition, task constraints sometimes require position control in some degrees-of-freedom (DOF), and force control in others. Thus, to fulfil these extra demands, an important area of robotics research is the implementation of stable and accurate force control. However, this is often difficult to achieve in practice, particularly where robots are operating in unpredictable or disordered environments.

A large number of force control techniques of varying complexity have been proposed over the last twenty years.[1,2] The most basic direct methods simply transform joint-space torques into Cartesian-space forces and torques, either in an open-loop fashion (which do not require the explicit measurement of forces and torques) or using inner and outer closed loops for accurate control of joint torques and Cartesian forces, respectively. However, since most industrial robots have position control loops that are not easily modified, indirect methods are often preferred. These involve modifying either joint or Cartesian position demands in order to control forces by deliberately introducing position control errors and using the inherent stiffness of the manipulator in different Cartesian directions. Alternatively, it is possible to add an outer force loop in systems that have a facility for real-time path modification.[3]

Two major problems in the implementation of practical controllers are stability and robustness. Stable force control is particularly difficult to achieve in 'hard' or 'stiff' contact situations, where the control loop sampling rate may be a limiting factor. In an attempt to improve stability, various methods have been proposed, the simplest being the addition of compliant devices at the robot wrist.[4] Another solution is to employ 'active compliance' filters, where force feedback data is digitally filtered to emulate a passive spring/damper arrangement.[5] However, both methods introduce a potentially unacceptable lag. Robustness is a problem where environmental uncertainty exists, and effective force control can only be achieved by employing an accurate environment stiffness detection technique and smooth switching between controller gains.[6] This slows down task execution, and can result in unstable contact when the effective stiffness at the robot/environment interface (K_e) varies significantly.

The principles behind many force control techniques are now well established, and recent increases in processing power of low-cost computers have enabled their implementation on practical, multi-DOF robotic systems. In a number of control applications, this rapid expansion in computational power has also led to an increased interest in 'intelligent control' techniques, such as those employing fuzzy logic, artificial neural networks and genetic algorithms. The potential advantages of intelligent controllers over more conventional methods have been widely reported. They include the ability to incorporate decision-making and heuristics into their design and they can also be highly effective for identification and control of non-linear systems. They are also potentially robust, maintaining good closed-loop system performance over a wide range of operating conditions.[7]

To-date, relatively few attempts have been made at combining intelligent control methodologies with practical real-time robotic force control. In the case of fuzzy control, an explanation for this is that the derivation of fuzzy rules for non-trivial, real-world tasks is often not intuitively obvious.[8] In addition, the approach can lead to fuzzy systems with large numbers of highly coupled rules, which are difficult to interpret and time-consuming to tune manually. For this reason, fuzzy control is often employed where a 'good

* Corresponding author. Centre for Hybrid Intelligent Systems, School of Computing and Technology, University of Sunderland, Sunderland (UK). E-mail: kevin.burn@sunderland.ac.uk
† Embedded Systems Laboratory, University of Leicester (UK).
‡ School of Mechanical and Systems Engineering, University of Newcastle upon Tyne (UK).

enough' solution is required, which does not necessarily have to be an optimum one, or meet a particularly tight specification.[9]

Fuzzy control has been successfully applied in a number of applications where conventional model-based approaches are impractical, particularly in processes which are complex, non-linear or imprecisely defined.[10] The linguistic properties of fuzzy systems are particularly suited to controllers that attempt to emulate human performance, for example where an operator is replaced by an automatic control system.[11] However, in addition to problems associated with dimensionality, i.e. large numbers of rules that must be evaluated in the inference process, the performance and stability of fuzzy systems are often difficult to validate analytically.[12]

Where attempts have been made to employ fuzzy logic in explicit robot force control, the main goal has been to improve the performance when the robot is in contact with environments whose parameters are either unknown or rapidly changing. Simulation studies on adaptable fuzzy force controllers have demonstrated good tracking performance despite wide variations in environment stiffness and effective application in specific contact situations such as deburring.[13–15] Also, improved performance using a hierarchical fuzzy force control strategy has also been demonstrated for particular classes of contact situation, such as 'peg-in-hole' insertion.[16] However, most have employed one-off fuzzy solutions, designed for specific robots and applications.

This paper describes part of an ongoing research programme investigating fuzzy and neuro-fuzzy techniques applied to robotic force control. A unique structured design method has been developed specifically for robot force controllers operating in situations where stiffness of contact K_e at the robot/task interface is variable and unpredictable. The method is described, together with a user-oriented software package that has been developed to enable non-expert users to implement the technique on conventional robot manipulators. The rationale behind the approach is to help bridge the gap between academic research and the industrial user, enabling the latter to utilise intelligent control methods, without needing specialist knowledge. In order to test the software, a controller is developed and implemented on a two-axis experimental robot and its performance compared to a conventional force controller. All simulation work is performed in a Matlab/Simulink environment.

2. OVERVIEW OF ROBOTIC FORCE CONTROL

Prior to examining the fuzzy approach to controller design, it is worthwhile to outline the force control problem under consideration and describe the conventional solution initially employed.

A position-based explicit force control technique was implemented, as shown in Figure 1. In the scheme, assuming the robot is in contact with an object, the force controller output is a Cartesian position or velocity demand vector ($\underline{X}_d$ or $\underline{\Delta X}_d$) in response to an error between the force reference vector ($\underline{F}_r$) and the measured force vector ($\underline{F}_m$). Thus, in the case of a 6 degree-of-freedom (DOF) industrial robot, $\underline{F}_m$

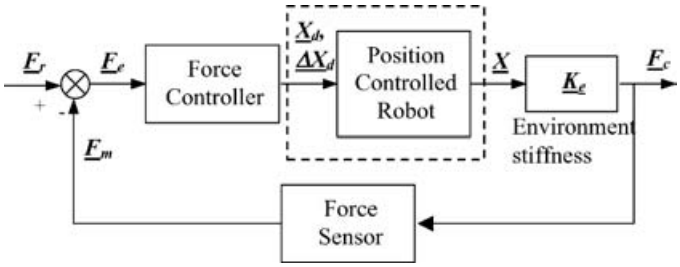

Fig. 1. Explicit force control.

would be a (6x1) vector of forces and torques measured using an appropriate sensor, $\underline{\Delta X}_d/\underline{X}_d$ would consist of translation and rotation velocity/position demands relative to a Cartesian co-ordinate frame, and so on. The actual position of the robot $\underline{X}$ is related to $\underline{F}_m$ by the stiffness vector $\underline{K}_e$, were the environment is modelled as a simple linear spring in each Cartesian direction.

The technique is suitable for implementing on many practical robot systems, most of which are equipped with Cartesian position and velocity control systems. In particular, whilst many commercially available robots do not allow users to modify their internal control loops, many provide a facility to enable an external computer to transmit position or velocity data to the controller.[17] This enables users to implement force control loops of the type shown in Figure 1, where the dashed box incorporates the individual joint control loops, together with the robot kinematics and dynamics.

A consequence of the technique is that system analysis can be simplified by replacing any vector element with an equivalent scalar quantity representing position, velocity or force data in any individual axis.[14] This is because most robot position control systems are designed to enable Cartesian set point tracking, so that end effector movements in Cartesian directions are largely decoupled. It is therefore possible to model each end effector co-ordinate independently without loss of generality, where the robot dynamics relative to that axis can be approximated by a second or third order transfer function.

The simplest idealised general form of a single axis of the experimental rig used in this study, described in more detail in section 5.1, is shown in Figure 2, where x is the displacement during contact and the conventional force controller shown is a proportional + velocity feedback (PV) controller. Thus, the contact force f_c is related to the reference force f_r as follows (assuming an ideal force sensor, such that $f_m = f_c$):

$$\frac{f_c}{f_r} = \frac{K_e K_p}{M_m s^2 + (c + K_v)s + K_e K_p} \tag{1}$$

where K_p is the proportional gain, K_v the velocity feedback gain and c and M_m the effective instantaneous robot arm damping and mass, respectively. Stiffness K_e varies between a minimum, determined by the objects in the environment with which the robot is in contact, and a maximum, limited by the stiffness of the arm and torque sensor. The latter is dominant when the robot is touching a surface of very high stiffness, i.e. in a hard contact situation.

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

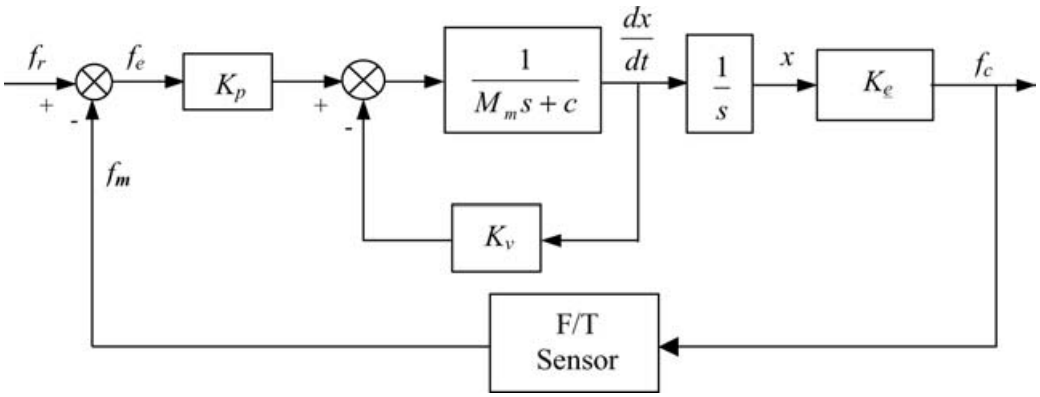

Fig. 2. Single-axis model of explicit force control.

Designing a fixed-gain conventional controller to meet a chosen specification for a specific value of K_e is, in principle, a relatively straightforward task. A problem arises when K_e is unknown or variable. This is illustrated in Figure 3, where typical step responses for K_e varying by a factor of 10 are illustrated. As expected, when the controller is tuned for stiff contact (i.e. low proportional gain), at low K_e the system is significantly overdamped with a relatively long settling time. Conversely, when it is tuned for soft contact (high proportional gain), significant overshoot and oscillatory behaviour occurs at high K_e.

In practical robotic systems the latter effect can have serious consequences, mainly in relation to system stability. In particular, the finite and relatively low sampling rates of many industrial robot control systems can result in unstable behaviour, a situation exacerbated by the presence of noise, non-linearities and other factors. For this reason, force controllers of the type described usually require some form of environment stiffness detection technique to enable the controller gains to be switched accordingly. The main problem with this process is that it is time consuming, often involving 'guarded moves' to contact in order to enable sufficient data to be collected for the algorithm to work. These methods can also be unreliable in the presence of transducer noise, and are not very effective in situations where K_e is variable or rapidly changing. In order to address these problems, a fuzzy logic approach was adopted, described in the following sections.

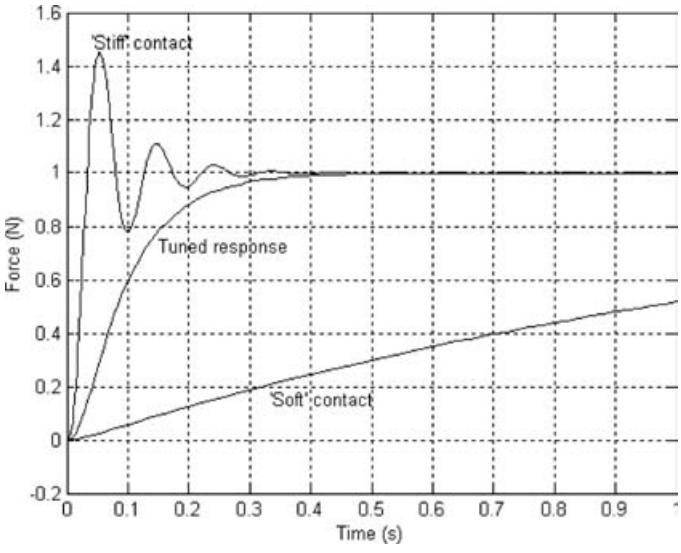

Fig. 3. Effect of varying K_e, conventional system.

3. SFAC: A FUZZY APPROACH TO ROBOTIC FORCE CONTROL

This section describes the Sunderland Fuzzy Adaptive Controller (SFAC), a new structured method for fuzzy force control.

A fuzzy inference system (FIS) can be considered as a rule-based expert system employing linguistic rules, and in control can facilitate a mathematical formulation of the uncertainty and imprecision associated with certain processes. This enables non-linear controllers to be devised which would be difficult to design using conventional methods.

The two most common forms of the fuzzy inference process are usually referred to as the Mamdani method, and the Sugeno (or Takagi-Sugeno-Kang) method.[18] To summarise, Mamdani-style systems employ output fuzzy sets and generally require more complex defuzzification techniques. In contrast, first-order Sugeno-style systems have rules of the form:

$$\textit{if } x \textit{ is } \boldsymbol{A} \textit{ and } y \textit{ is } \boldsymbol{B} \textit{ then } z = p * x + q * y + r \qquad (2)$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are fuzzy sets in the antecedent (the initial 'if' part of the fuzzy rule), x, y and z are the inputs and output, respectively, and p, q and r are constants.

As already described, attempts have been made to combine fuzzy logic and explicit robot force control. However, no general and analytically based design strategy has evolved, for example to enhance the performance of conventional controllers using fuzzy techniques. To address this, two fuzzy control techniques have been pioneered at Sunderland, including the Sunderland Fuzzy Adaptive Control (SFAC) method.

SFAC is a method for designing Sugeno-style fuzzy controllers that effectively produces a PV controller with variable gains, capable of maintaining acceptable performance irrespective of K_e.[19,20] A block diagram of the arrangement is shown in Figure 4, and the method can be summarised as follows. Firstly, a FIS is created to emulate a

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

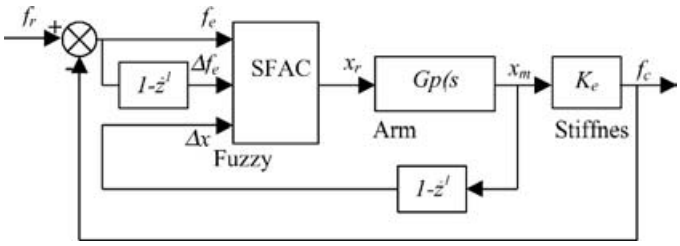

Fig. 4. Fuzzy Control Using SFAC.

conventional PV controller, tuned for a high K_e environment. The FIS is assigned three inputs (force error, force rate error and velocity: f_e, Δf_e, Δx), and one output (controller output: u), where the input ranges can be obtained mathematically, or measured from conventional system data. In order to create a linear system, initially only a single membership function (MF) for each input and output is required. A linear MF is chosen for the output, of the form given in equation (2). By assigning names *normal* to the input MFs, and u_1 to the output MF, the following rule produces the desired linear (planar) control surface. Note that a consequence of employing only one rule is that no defuzzification algorithm is required:

$$\text{if } (f_e \text{ is } normal) \text{ AND } (\Delta f_e \text{ is } normal) \text{ AND } (\Delta x \text{ is } normal) \text{ then } u \text{ is } u_1 \quad (3)$$

Output u_1 is defined by

$$u_1 = K_1 * f_e + K_2 * \Delta f_e + K_3 * \Delta x + K_4 \quad (4)$$

where $K_1$is a positive constant (equal to the forward gain K_p of a conventional PV controller) and K_3 a negative constant (equal in magnitude to the velocity feedback gain K_v). K_2 and K_4 in this case are set to zero.

The choice of MF type is influenced by the concept of data 'spread' and fully explained in reference [19]. However, to summarise, for the single rule system, each input is assigned a single Gaussian MF. This is defined for any MF i by two parameters, the centre point of the distribution and the standard deviation (k_i and σ_i, respectively) and, for an input x, the degree of membership $\mu_i(x)$ for the MF is given by:

$$\mu_i(x) = \exp\left[\frac{-(x-k_i)^2}{2\sigma_i^2}\right] \quad (5)$$

The general rule-of-thumb is that $\mu_i(x)$ is negligible when the magnitude of the input x is greater than $3\sigma_i$, where σ is measured directly for each input parameter from step response data obtained either experimentally or using simulation.

A particular advantage of the Gaussian MF is that when it is centred at zero, the sign of the input data does not have to be taken into account when calculating degree of membership $\mu_i(x)$, since it is calculated using the simple exponential formula of equation (5). This is in contrast to a textbook 'rule table' method, where changes in sign are dealt with by adding MFs to account for both positive and negative data, resulting in more complex systems.[21] It is also suitable for real-time implementation, being a simple and computationally fast calculation.

Since the single rule system emulates a conventional PV controller it suffers the same disadvantages when K_e is unknown or variable. However, having created the initial FIS, it is now possible to tune the controller using a combination of analytical and intuitive methods.

With the system tuned for high K_e, during low K_e contact the maximum value of Δf_e is reduced. This reflects the over-damped response of the system at low K_e, in this case an undesirable effect that can be improved by increasing the proportional gain component of the controller output given by equation (4) if lower Δf_e is 'detected' by the FIS. This is achieved firstly by adding a second Gaussian input MF to the Δf_e input set, with a smaller standard deviation (or 'width') than $\sigma_{f_e}|_{normal}$, and named *low*. A second rule is then added to take into account this decrease in Δf_e relative to the 'normal' (desired) profile and the knowledge that during the dynamic response to a step input, Δx is inversely proportional to K_e. In the case of a system with varying gain K_p, assuming it is initially at a lower value and increases if K_e is less than maximum (one of the fundamental aims of this system), then Δx will also begin to increase. Therefore, the following rule is added:

$$\textit{if } (\Delta f_e \textit{ is low}) \text{ AND } (\Delta x \textit{ is high}) \textit{ then } u \textit{ is } u_2 \quad (6)$$

where u_2 has the same form as u_1, equation (3), but with a modified forward gain component, K_{1a}, such that $K_{1a} > K_1$, and $\sigma_{\Delta x}|_{\text{high}} > \sigma_{\Delta x}|_{\text{normal}}$. The rule also ensures that the proportional gain component of the FIS does not increase unnecessarily when Δf_e decreases when the system approaches the setpoint.

During initial testing it was found that the system performed well over a wide range of K_e, although had a slight tendency to overshoot at some intermediate values. To eliminate this, it was found necessary to modify $\sigma_{\Delta \text{fe}}|_{\text{low}}$ to slightly reduce the width. This had the effect of reducing the firing strength of the second rule (equation (6)) in intermediate regions of K_e. An optimum value was found by trial and error.

The main attraction of this technique is its similarity to conventional controller design and its relative simplicity in only having two rules. Typical MFs for input variables Δf_e *and* Δx are shown in Figure 5.

4. THE SOFTWARE DESIGN TOOL

The SFAC design method described previously was developed as a manual technique to produce a fuzzy controller for a second order force control system. To convert it into a structured method suitable for implementation as a computer-based design tool, work was undertaken in the following five areas, to devise appropriate automation strategies:

- Definition of the FIS parameters for a second, or higher, order model of the robot arm.
- Generation of a Matlab FIS data file (henceforth referred to as a *.fis* file).

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

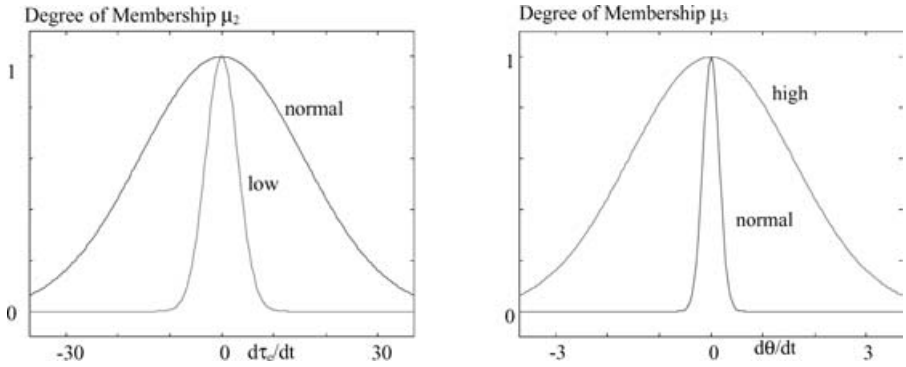

Fig. 5. Membership Functions for inputs 2 and 3 to fuzzy controller.

- Execution of a Simulink block diagram (containing the FIS and robot model) from the design tool to obtain the simulated response.
- Tuning in a Simulink environment.
- Integration of the *.fis* into the real-time control loop of a practical robot system.

Initially, it was assumed that all systems under consideration would be represented by second order models of the form given by equation (1). However, it was decided to include an additional option to facilitate the design of SFAC controllers for higher order models. This work is described in the following sub-sections.

4.1. Design calculations

The ranges of the input parameters are required in order to define the input MFs. The first step is to obtain input data ranges for Δf_e and Δx(f_e is simply ± 1, assuming force demands are normalised to a unit step input). This is achieved by using standard second order differential equations for a unit step input and assuming that the system is critically damped ($\zeta = 1$). This simplifies the analysis and is compatible with a common force control specification, where a fast response with zero overshoot is desirable. It was decided that the following system parameters would need to be readily available as design tool inputs: arm inertia, damping, desired closed loop natural frequency, and the minimum and maximum values of K_e (J, c, ω_n, $K_{e\,min}$ and $K_{e\,max}$, respectively). These can be obtained by using standard control system identification techniques, including mathematical modelling and approximation, and data obtained from experimentation.

The data ranges are thus derived as follows:

(a) f_e: $-1 < f_e < +1$ for a unit step.

(b) Δf_e (i.e. df_e/dt): For a unit step demand ($f_r = 1$), f_e can be expressed as:

$$f_e = 1 - f_c \tag{7}$$

From basic conventional control theory, f_c can be calculated as follows:[22]

$$f_c = 1 - e^{-\omega_n t}(1 + \omega_n t) \tag{8}$$

$$\therefore\ f_e = e^{-\omega_n t}(1 + \omega_n t) \tag{9}$$

The maximum value of df_e/dt occurs when $d^2 f_e/dt^2 = 0$. Since ω_n was selected as one of the user inputs, it was necessary to obtain an expression for the range of df_e/dt in terms of ω_n. Equation (9) was thus differentiated twice and the result equated to zero to obtain the maximum value for df_e/dt:

$$\frac{df_e}{dt} = -\omega_n^2 t e^{-\omega_n t} \tag{10}$$

$$\frac{d^2 f_e}{dt^2} = \omega_n^2 e^{-\omega_n t}(\omega_n t - 1) \tag{11}$$

$$\frac{d^2 f_e}{dt^2} = 0 \text{ for max/min, i.e when } t = \frac{1}{\omega_n} \tag{12}$$

thus, substituting equation (12) into equation (11):

$$MAX\left(\frac{df_e}{dt}\right) = -\omega_n^2 \left(\frac{1}{\omega_n}\right) e^{-\omega_n\left(\frac{1}{\omega_n}\right)} = -\omega_n e^{-1}$$

The range of df_e/dt can thus be expressed in terms of ω_n as follows:

$$-\omega_n e^{-1} \le \frac{df_e}{dt} \le \omega_n e^{-1} \tag{13}$$

(c) Δx (i.e. dx/dt): By definition, stiffness = force per unit displacement. Thus:

$$K_e = df_c/dx \quad \text{or} \quad df_c/dt = K_e dx/dt$$

Hence, $MAX\ (dx/dt) = (1/K_{e\,min}) * MAX(df_c/dt)$

As before, it was observed that the range of df_c/dt, and hence the range of dx/dt, could be expressed in terms of user inputs ω_n and $K_{e\,min}$. Differentiating equation (8):

$$\frac{df_c}{dt} = \omega_n^2 t e^{-\omega_n t} \tag{14}$$

Also, from equation (12), $MAX(df_e/dt)$, and hence $MAX(df_c/dt)$, occur when $t = 1/\omega_n$. Thus:

$$MAX\left(\frac{df_c}{dt}\right) = \omega_n^2 \left(\frac{1}{\omega_n}\right) e^{-\omega_n\left(\frac{1}{\omega_n}\right)} = \omega_n e^{-1}$$

$$MAX\left(\frac{dx}{dt}\right) = \frac{1}{K_{e\,min}} * MAX\left(\frac{df_c}{dt}\right) = \frac{\omega_n e^{-1}}{K_{e\,min}} \tag{15}$$

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

Therefore the range is: $\frac{-\omega_n e^{-1}}{K_{e\,min}} \leq \frac{dx}{dt} \leq \frac{\omega_n e^{-1}}{K_{e\,min}}$ (16)

These calculations are based upon a second order conventional model and, although certain approximations are made, they are considered valid in determining the data ranges of the fuzzy controller. This is because a primary aim of the FIS is to ensure that the overall system response closely approximates this conventional response throughout the range of K_e.

4.2. Automatic calculation of input MF standard deviations

Recalling that the general rule of thumb for a Gaussian distribution is that the value of the function is negligible when the magnitude of the input is greater than three standard deviations (σ_i), expressions for the standard deviations for the input MFs were deduced as follows:

(a) f_e:'*normal*': $\sigma_{fe|norm} = 1/3$ (since the range is -1 to $+1$)
(b) df_e/dt:'*normal*': $\sigma_{dfe/dt|norm} = (df_e/dt|_{max})/3$
(c) df_e/dt: '*low*': since $df_e/dt|_{max}$ is proportional to ω_n, and ω_n is proportional to $\sqrt{K_e}$ in the critically damped case, it can be deduced that

$$\sigma_{dfe/dt|low} = \sigma_{dfe/dt|norm} * \sqrt{\frac{K_{e\,min}}{K_{e\,max}}}$$

(d) dx/dt: '*high*': $\sigma_{dx/dt|high} = (dx/dt|_{max})/3$
(e) dx/dt: '*normal*': $\sigma_{dx/dt|norm} = \frac{K_{e\,min}}{K_{e\,max}} * \sigma_{dx/dt|high}$

(Note that since $dx/dt \propto 1/K_e$, 'normal' dx/dt (for $K_e = K_{e\,max}$), is less than that which occurs when K_e is lower than maximum).

To summarise, it was determined that an estimate of the input MF standard deviations can be obtained from the calculated input data ranges for dfe/dt and dx/dt and the user inputs of $K_{e\,min}$ and $K_{e\,max}$. Again each of the values can be calculated by an appropriate function in the software.

4.3. Definition of output sets

The output MFs for the two rules are defined in as:

$$u_1 = K_1 f_e + K_3 \cdot dx/dt \tag{17}$$

and

$$u_2 = K_{1a} \cdot f_e + K_3 \cdot dx/dt \tag{18}$$

where K_1 and K_{1a} are positive constants (equal to the forward gain K_p of the conventional PV controller, calculated for the case of maximum K_e), $K_1 < K_{1a}$ and K_3 is a negative constant (equal to the velocity feedback gain K_v).
From Figure 2, the equations for a second order PV control system can be derived as follows:

$$\omega_n^2 = \frac{K_e K_p}{J} \tag{19}$$

$$2\zeta\omega_n = \frac{c + K_v}{J} \tag{20}$$

Rearranging (19) and (20), we get:

$$K_p = \frac{\omega_n^2 J}{K_e} \tag{21}$$

and

$$K_v = 2\zeta\omega_n J - c \tag{22}$$

The output sets are then defined as:

$$U_1 = [K_1 \quad 0 \quad -K_3 \quad 0]$$
$$U_2 = [K_{1a} \quad 0 \quad -K_3 \quad 0]$$

(where K_{1a} is calculated for $K_{e\,min}$ case).

Thus, the values of K_1, K_{1a} and K_3 can be obtained from user inputs of ω_n, J, c and $K_{e\,min}$, and calculated by functions implemented in the software.

4.4. Software implementation

The functional and non-functional requirements are summarised in Table I.

Borland C++ Builder was chosen to implement the design tool. This is a rapid application development (RAD) tool for creating Windows applications in C++ using form-based programming. It enables rapid prototyping of graphical user interfaces (GUIs) using drag and drop techniques in a similar manner to that pioneered by Microsoft with Visual Basic. Components are dropped onto forms in the RAD design environment and manipulated using their properties, methods and events. *Borland C++ Builder* enables the programmer to utilise object oriented programming and all the benefits that go with it, such as re-use, abstraction and a modular approach to programming. An ActiveX automation link was used in the application to enable communication and data exchange between the program and Matlab/Simulink.

A Data Flow Diagram showing an overview of the logical data processing is shown in Figure 6 and example input and output screenshots are shown in Figures 7 and 8.

When the simulation is run, a Simulink model is displayed and the response can be viewed by double clicking on an 'Oscilloscope' output block. The value of the integral of time by absolute error (ITAE – described in Section 5) is also displayed. This is used in the tuning algorithm but is also an indicator of performance. A help form is automatically displayed giving a few hints on running Simulink and obtaining a plot of the response, although to run the software, detailed knowledge about running Matlab is not required – one of the principal aims of the project.

To execute the tuning option, the user must click on the Tune button. This results in a modal form ('Confirm Tuning') being displayed, which warns the user that the tuning algorithm can take about 10 minutes to execute and requests confirmation before the user can to continue. The user must then select either the Tune or Cancel button to close the form and continue with programme execution.

Currently, tuning is a simple and time-consuming operation that iteratively adjusts the MF parameters within specified bounds and, for each set of parameters, runs a

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

Table I. Summary of Requirements.

	Option 1: 2nd Order	Option 2: Higher order
User inputs from keyboard	J_m, c, $K_{e\,min}$, $K_{e\,max}$, ω_n	$f_{e\,max}$, $df_e/dt\|_{max}$, $dx/dt\|_{max}$, K_1, K_{1a}, K_3, $K_{e\,max}$, $K_{e\,min}$, ω_n
Input from existing file of type:	*filename.fuz*	*filename.fz2*
Calculated outputs to screen	Range of : f_e, df_e/dt, dx/dt σ for : $f_{e\,normal}$, df_e/dt_{normal}, df_e/dt_{low}, dx/dt_{normal}, dx/dt_{high}, K_1, K_{1a}, K_3	σ for : $f_{e\,normal}$, df_e/dt_{normal}, df_e/dt_{low}, dx/dt_{normal}, dx/dt_{high}
Save *.fuz* file option	Saves input data and calculated outputs to *filename.fuz*	Saves input data and calculated outputs to *filename.fz2*
Creation of fuzzy controller design file	Creates file called *Test.fis* formatted for Matlab use. List to screen optional	Creates file called *Test.fis* formatted for Matlab use List to screen optional.
Save *.fis* file option	Saves *Test.fis* to a user-selected *filename.fis*	Saves *Test.fis* to a user-selected *filename.fis*
Run Simulink to obtain response	User to specify value of K_e for simulation and Simulink model name from option list	User to specify value of K_e for simulation and Simulink model name from option list
Run tuning option	Optional, with results available to user	Optional, with results available to user

simulation in Simulink. The controller with the optimum ITAE (according to the desired performance specification) is then selected as the output FIS. Work is presently underway to improve the tuning algorithm, for example using genetic programming methods and other optimisation techniques.

When the tuning algorithm concludes, the Simulink model reappears. The user can then use these values to modify the *Test.fis* file and re-run the simulation with the tuned FIS.

5. EXPERIMENTAL STUDIES

5.1. Test facility

A research facility has been developed in the form of a planar, two degree-of-freedom (DOF) robot arm, with a range of customized controller options now under investigation. The arm is shown in Figure 9.

Brushless servomotors, powered by digital servoamplifiers, are used to actuate the joints, where the control loop for each axis is closed via a multitasking digital signal processor (DSP). This is embedded in a *Delta Tau* Programmable Multi-Axis Controller (PMAC) motion control card, connected to a PC bus to facilitate communications with the host processor. Each axis has an individual PID controller, with feedforward control to enable accurate velocity profile following, and the various control techniques being investigated can be implemented either via the 'open' slots on PMAC, or with a processor running in parallel with the DSP and synchronized using interrupts.

All instrumentation is connected via a local area network, currently in the form of a Controller Area Network (CAN) bus. This enables multi-drop connection of analogue, digital or customized I/O, including a six-axis force/torque sensor developed in-house for the project and a Baumer photoelectric proximity sensor.

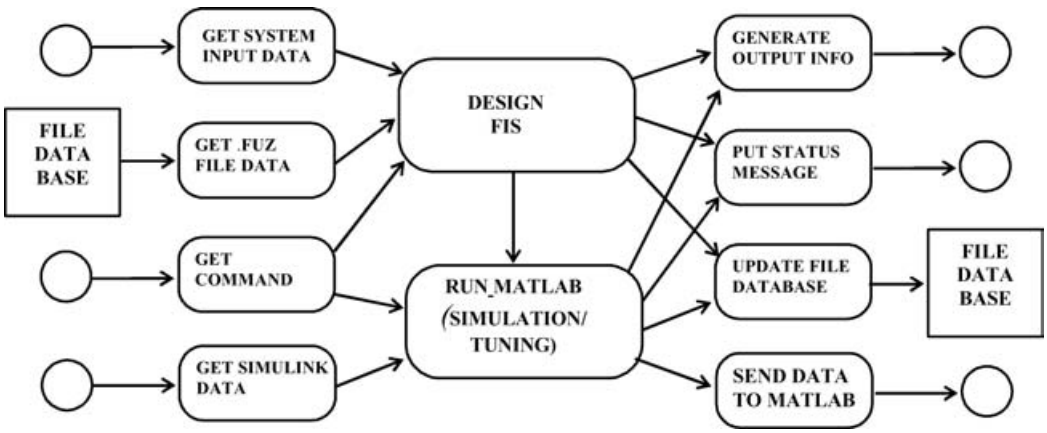

Fig. 6. Data Flow Diagram for software design tool.

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

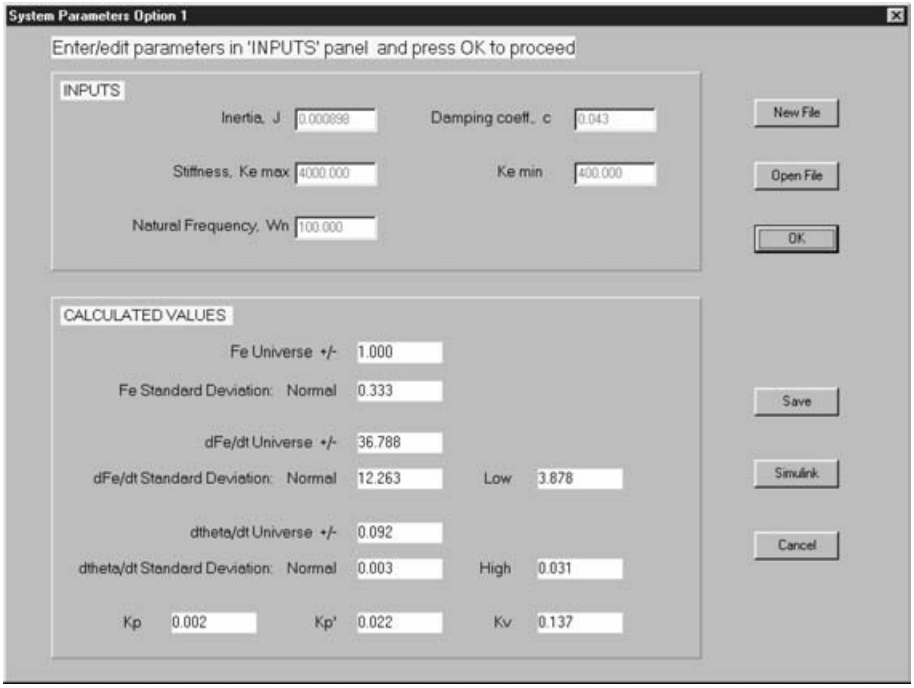

Fig. 7. Example Input screen.

The conventional controller was first designed under simulation, using an accurate model of the system developed in Matlab. Software has been created to enable the simulated controllers to be automatically integrated into the real-time control loops, which are written in *C++*. Thus, the high level, 2-DOF force controllers form an outer loop, with inner, joint level loops provided by PMAC. An advantage of the arrangement is its generic nature, since the technique could be applied to 6-DOF industrial robots equipped with a suitable path modification scheme. For all of the experiments, a sampling frequency of 500 Hz was achieved.

The contact experiments were conducted as follows: A variety of contact surfaces were used in the form of rectangular section beams clamped to the bench to form a cantilever, as illustrated schematically in Figure 10. The stiffest material was steel and the most flexible PVC, with estimated combined robot/environment transverse stiffness of 168 and 11 N/mm, respectively. The robot

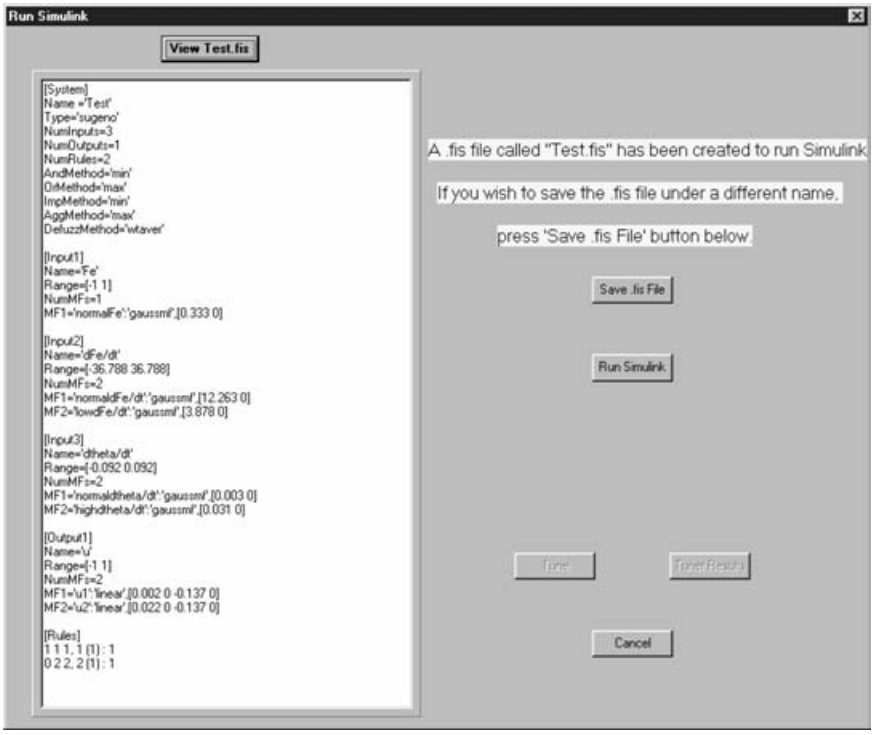

Fig. 8. Example Output screen.

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

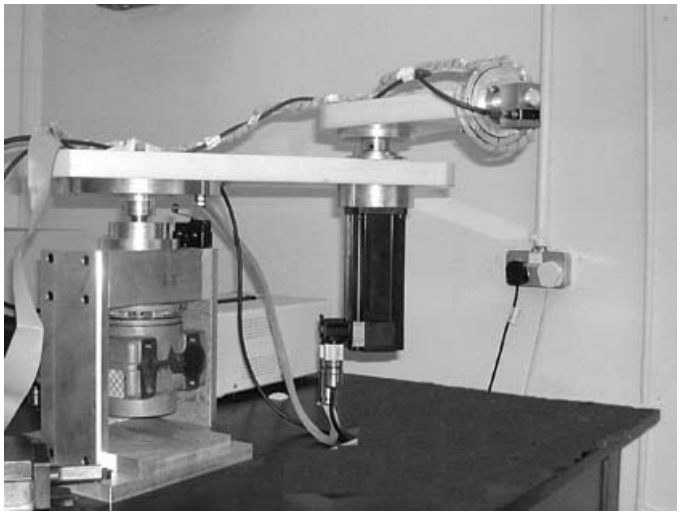

Fig. 9. Photograph of experimental robot.

was programmed to approach the environment at constant velocity, until the surface was within 5 mm, detected by the proximity sensor. The outer control scheme was then switched from velocity control to the force control method under investigation, and a force of 30 N applied to the surface. The goal was to achieve the setpoint within 1 second of the robot end effector making contact and data was logged over a period of 20 seconds.

5.2. Experimental results

Experimental results, in the form of step response tests, are shown in Figure 11, where they are contrasted with a conventional PV controller tuned for stiff contact. It can be seen that the fuzzy controller improves performance over the full range of K_e, the main effect being to speed up the response at low K_e without adversely affecting the performance at high K_e.

As previously indicated, a more quantitative measure of performance has been undertaken by measuring the integral of time by absolute error (ITAE) for the methods described above. ITAE is a popular measure of performance since it does not discriminate against the large initial error in the response following a step demand, but does penalise smaller errors at a later time. It is given by the following expression:

$$ITAE = \int_0^\infty t|e(t)|\,dt \tag{23}$$

Results for the conventional controller (tuned for high K_e), and SFAC fuzzy controller are shown in Table II, where performance index I_p calculated as follows:

$$I_p = |ITAE - ITAE_{\text{ideal}}| \tag{24}$$

Here, $ITAE_{\text{ideal}}$ is the value calculated for the conventional controller used at high K_e. Thus, $I_p = 0$ for 'ideal performance' and the higher I_p, the more performance has been degraded.

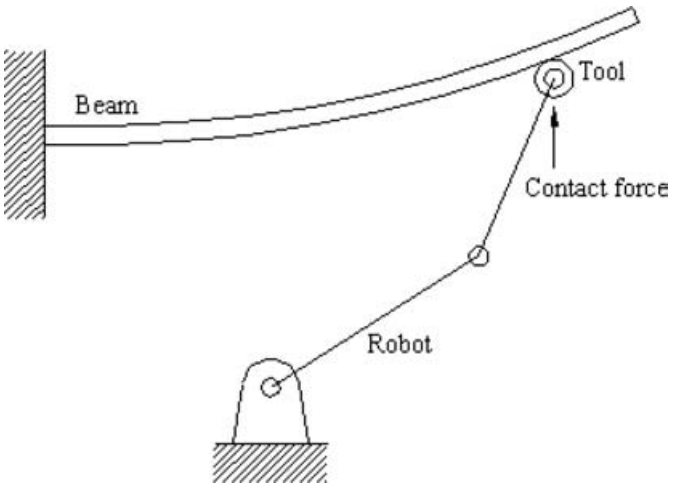

Fig. 10. Schematic of experimental beam contact task.

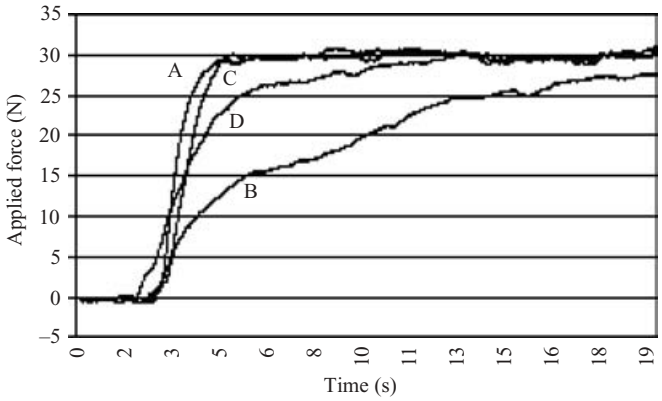

Fig. 11. Typical experimental results: A – Conventional (hard); B – Conventional (soft); C – SFAC (hard); D – SFAC (soft).

Table II. Analysis of Step Response Tests.

		K_e	
	Controller	High	Low
ITAE	Conventional	370	913
	SFAC	399	447
I_p	Conventional	0	543
	SFAC	29	77

6. CONCLUSIONS

A novel computer-based design tool has been developed that enables the user to carry out the complete design of a fuzzy controller from a minimal number of inputs. The software system developed enables the controller to be automatically validated and tested within a Matlab/Simulink environment, and the parameters of the FIS tuned to optimise its performance. The software has successfully implemented the SFAC method devised in related research, which was developed as a manual technique aimed primarily at developing fuzzy controllers for systems described by second order models, and in addition provides an automatic tuning

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press

option. The method has been enhanced during software development by incorporating the option of employing higher order system models, either through the input of extra data from a conventional model, or by using a second order approximation to the higher order system.

The practical realisation of robotic force control remains a problematic area of research. However, SFAC demonstrates the potential of using fuzzy logic to overcome fundamental difficulties associated with applications where environmental uncertainty, in a technique derived from conventional controller design theory. Thus, an important aspect of the work has involved bridging the gap between traditional automatic control theory and so-called soft computing, which aims to replace a highly analytical and mathematical approach with a behavioural and experimental one. The SFAC controller is particularly promising in this respect.

The next stage in the validation of both the SFAC method and the design software will be to implement a similar controller on a 6-DOF industrial robot, and then perform a range of comparative tests with conventional solutions. This work is currently underway using a PUMA 762 industrial robot employing a six-axis, wrist-mounted F/T sensor to measure Cartesian forces and torques. In particular, they are being applied to gear deburring and extensive work is being carried out to validate their efficiency in this, and similar, industrial applications.

In a separate project (CETUS – Classification of Environment and Terrain using Unsupervised ṢOFMs), the method is being extended for use within a novel control architecture for intelligent autonomous underwater vehicles (AUVs). This uses artificial neural networks in the form of Kohonen SOFMs to recognise and classify terrain, and to permit strategic selection of a dynamic set of navigation behaviours. The approach is based on neuro-fuzzy pre-mission learning to enable the AUV to subjectively select its navigation hierarchies dynamically, in an environment which is also dynamic. The ultimate goal is to define a mission, launch an AUV and forget, until the vehicle returns with its mission completed.

SFAC has many features which make it extensible to CETUS, and it can be primarily used as a fail safe mechanism to prevent damaging contact with obstacles, i.e. when the primary navigation is less than perfect the AUV can probe its environment up a pre-defined contact force. Thus, it will be employed primarily as an identification technique for environmental characteristics. Features it has in common with the application described in this paper include:[23]

- Dynamic environmental stiffness in a natural unstructured environment.
- Stability and robustness to help prevent vehicle loss.
- An optimum solution is not required although efficient navigation is sought.
- Fuzzy logic to enable subjective robot ‘judgements’.

As discussed earlier, few attempts have yet been made to combine intelligent control with practical real-time robotic force control and CETUS is further addressing this issue.

References

1. Ganwen Zang and Ahmad Hemami, “An Overview of Robot Force Control,” *Robotica*, **15**, Part, 473–482 (1997).
2. D. E. Whitney, “Historical Perspective and State of the Art in Robot Force Control,” *Int. J. Robotics Res.* **6**(1), 3–14 (1985).
3. R. Bicker, K. Burn, D. Glennie and S. M. Ow, “Application of force control in telerobotics,” *European Robotics and Intelligent Systems Conference* (*EURISCON ’94*), Malaga, Spain, (1994) pp. 1509–1517.
4. D. E. Whitney and J. L. Nevins, “What is the Remote Centre Compliance (RCC) and what can it do?,” *Proc Int Symp on Industrial Robots*, Washington DC (1979) pp. 135–152.
5. W. S. Kim, B. Hannaford and A. K. Bejczy, “Force Reflection and Shared Compliant Control in Operating Telemanipulators with Time Delay,” *IEEE Trans on Robotics and Automation* **8**(2), 176–185 (1992).
6. S. M. Ow, “Force Control in Telerobotics,” *PhD Thesis* (University of Newcastle upon Tyne, UK, 1997).
7. D. A. Linkens and H. O. Nyongesa, “Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications,” *IEE Proc. Control Theory Appl*, **143**(4), 367–386 (1996).
8. O. Wolkenhauer and J. M. Edmunds, “A critique of fuzzy logic in control,” *Int. J. Electrical Engineering Education*, **34**(3) 235–242 (1997).
9. T. J. Ross, *Fuzzy Logic with Engineering Applications* (McGraw-Hill, New York, 1995).
10. K. Hirota and M. Sugeno, *Advances in Fuzzy Systems, Applications and Theory*, **Volume 2**: *Industrial Applications of Fuzzy Technology in the World* (World Scientific Publishing, Singapore, 1995).
11. K. M. Passino and S. Yurkovich, *Fuzzy Control* (Addison-Wesley, Harlow, 1998).
12. S. G. Cao, N. W. Rees and G. Feng, “Lyapunov-like stability theorems for continuous-time fuzzy control systems,” *Int. J. Control*, **69**(1), 49–64 (1998).
13. M. Tarokh and S. Bailer, “Adaptive fuzzy force control of manipulators with unknown environment parameters,” *J. Robotic Sys.* **14**(5), 341–353 (1997).
14. H. Seraji, “Nonlinear and Adaptive Control of Force and Compliance in Manipulators,” *Int. J. Robotics Research* **17**(5), 467–484 (1998).
15. K. Kiguchi and T. Fukuda, “Intelligent position/force controller for industrial robot manipulators – application of fuzzy neural networks,” *IEEE Trans. Industrial Electronics* **44**(6), 753–761 (1997).
16. Lin, Shih-Tin and Ang-Kiong, “Hierarchical fuzzy force control for industrial robots,” *IEEE Trans. Industrial Electronics* **45**(4), 646–653 (1998).
17. M. Short, “A generic robot controller architecture for advanced and intelligent robots,” *PhD Thesis* (University of Sunderland, UK, 2003).
18. J.-S. R. Jang and N. Gulley, *Fuzzy Logic Toolbox for Use with MATLAB* (User Guide, The Math Works Inc., 1995).
19. K. Burn and R. Bicker, “Development of a non-linear force controller using fuzzy logic techniques,” *I. Mech. E. J. Sys. and Control Eng.* **214**(1), 197–206 (2000).
20. K. Burn, M. Short and R. Bicker, “Adaptive and Nonlinear Fuzzy Force Control Techniques Applied to Robots Operating in Uncertain Environments,” *J. of Rob. Sys.* **20**(7), 391–400 (2003).
21. L. Reznik, *Fuzzy Controllers* (Newnes-Butterworth-Heinemann, Oxford, 1997).
22. R. J. Richards, *An introduction to Dynamics and Control* (Longman, London, 1979).
23. K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru and G. Demetriou, “Control architectures for autonomous underwater vehicles,” *IEEE J. of Control Systems* **17**(6), 48–64 (1997).

https://doi.org/10.1017/S0263574704000785 Published online by Cambridge University Press