UNIVERSITY *of* NORTHUMBRIA *at* NEWCASTLE

A Preliminary Investigation on the Feasibility of A Fuzzy Logic Controller for An Electric Welding Machine

Sayeed N Ghani
University of Northumbria at Newcastle
Department of Electrical, Electronic Engineering and Physics

April 18, 1996

A Preliminary Investigation on the Feasibility of A Fuzzy Logic Controller for An Electric Welding Machine

1 Introduction

This feasibility study has been carried out for Rolls Royce International Research and Development (RRIRD), Fossway, Newcastle upon Tyne.

RRIRD utilizes an electric arc welding machine to repair turbine blades by building layers of metal on to the corroded blades damaged whilst in use. They have found this to be more cost effective than to manufacture or purchase altogether new blades. At the moment their welding machine has no feedback control over temperature. As a consequence the temperature varies rather widely as the arc travels over the workpiece. The reasons for such a behaviour are the non uniformity of the dimension of the blade along the coordinate axes and end discontinuities. This results in the thermal resistance and capacitance varying with the position of the arc. The resistance of the arc itself fluctuates with time causing wide variation in heat power injected.

The following types of controllers could provide closed-loop temperature control.

(a) Self-tuning control [1].

(b) Model reference adaptive control (MRAC) [2].

(c) Fuzzy logic control (FLC) [3, 4].

Self tuning control (a) requires explicit parameter estimation. This approach was rejected in favour of MRAC (b) and FLC (c) on grounds of unnecessary complexity resulting from the usage of a parameter estimator. Further, because of the large amount of numerical calculations involved expensive high speed processors may be required if such an approach was taken. The MRAC approach using signal synthesis adaptation circumvents the necessity of explicit estimation of the dynamic parameters of the plant, and can be easily implemented in analog electronics. A digital processor may also be used instead. MRACs are essentially nonlinear time varying systems, and rigorous procedures exist for the design of unconditionally stable systems using hyperstability concepts. For self-tuning control design procedures also exist which would yield unconditionally stable system. The other drawback from which self-tuning control suffers is that the controller is linear, and a properly designed nonlinear controller can have much superior performance [5]. FLCs are nonlinear controllers; when designed properly they will out perform any linear controllers.The greatest advantage of FLCs is that their synthesis does not require explicit knowledge of the dynamic model of the plant. Such controllers use their knowledge base, developed from heuristics and human intelligence, to force the plant to follow or track the reference input.

At this moment no attention has been given to the MRAC approach. It was decided to investigate the feasibility of FLC because of its unique features described above, and this document reports this. There is a problem however. No rigorous design procedure exist based upon stability issues which could synthesize rules guaranteeing unconditional stability and optimum performance. However, methods for tuning of FLC parameters exist, in which the stability issue [3] is

addressed via a supervisory control, leading to sub-optimal fuzzy adaptive control. Because FLCs are artificial intelligence (AI) systems, rules are usually devised through human ingenuity to surmount this stability problem. Selection of suitable numerical values for the parameters is, however, pure guess work compared to design methods based on the stability point of view. In this investigation the worst case combination of plant parameters from stability point of view was unknown. It was, therefore, decided to address the stability issue, posterior to the controller design, by simulating the behaviour of the total system under feedback control. First approach would be to establish how the system would track a sinusoidal variation in the demanded temperature reference when the plant parameters were altered randomly by a factor of two above and below the nominal values. Second approach would be to apply Lyapunov stability test on the system. If the system satisfies this test then one could safely conclude that the proposed system will always be stable. Since Lyapunov stability tests are rather conservative, failure to satisfy the criterion does not mean that the system will necessarily be unstable. Here simulation becomes an invaluable aid. Further, in order to apply Lyapunov test, a model of the plant is required, and for this purpose usage of nominal values may be inconclusive. As mentioned earlier, in this section, the worst case of parameter combination is just not known. When all these considerations are taken into account simulation using random parameters seems to be the best approach towards stability assessment.

2 Initial information provided by RRIRD

Like most design methodology the procedure adopted here utilizes simulation of the total closed-loop system in an interative fashion. For this purpose hardly any quantitative information about the nominal values of the dynamic parameters of the plant was available other than a time of 2 s required to heat the welding spot to the working temperature. The other specification was that the controller should be capable of accepting a maximum temperature demand of 1500 deg C. It was also stated that the current source is capable of delivering around 120 A with time lag of around 1 ms, and the shortest permissible sampling period for the digital processor was 10 ms. No information was, however, available on the resistance of the electric arc, thermal resistance or capacitance of the workpiece from which the dynamic model of the plant could be established for simulation purposes.

3 Strategy for design

It was decided to proceed as follows.

(a) Establish a quantitative dynamic model from common sense for simulation purposes.

(b) Decide on the structure, topology or architecture for control.

(c) Generate rules for the FLC

(d) Write a C programme for simulating the total system on a PC.

(e) Establish suitable values for the controller parameters through simulation.

(f) Randomly generate the dynamic parameters of the plant within prescribed upper and lower bounds. Thus check the

stability, and study the performance of the total system by simulating realistic operating conditions.

(g) Apply Lyapunov's criterion to check global asymptotic stability of the system.

4 Dynamic model of the thermal subsystem

The thermal aspects of the plant are modelled by the following electrical analog in Figure 1.

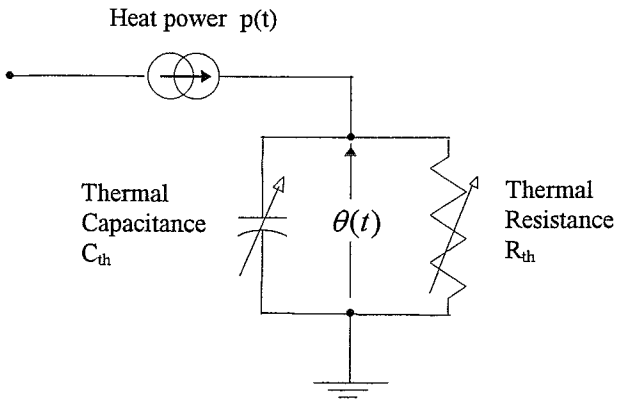


Figure 1. Electric circuit analog of the welding spot.

The various quantities shown in this figure are as follows.

Instantaneous current delivered by the current generator $i(t)$ A
Resistance of the electric arc R_{arc} Ω
Thermal resistance R_{th} deg C W^{-1}
Thermal capacitance C_{th} W s (deg C)$^{-1}$
Temperature above ambient θ (t) deg C
Thermal power input p(t) $= i^2 R_{th}$ W

The dynamic model of the thermal subsystem is derived as follows.

$$C_{th}\frac{d\theta}{dt} + R_{th}\theta(t) = p(t)$$

$$\dot{\theta}(t) + \frac{\theta(t)}{R_{th}C_{th}} = \frac{p(t)}{C_{th}}$$

Taking Laplace transforms, and neglecting the initial condition

$$\bar{s}\bar{\Theta} + \frac{\bar{\Theta}}{R_{th}C_{th}} = \frac{\bar{P}}{C_{th}}$$

$$(\bar{s} + \frac{1}{R_{th}C_{th}})\bar{\Theta} = \frac{\bar{P}}{C_{th}}$$

Hence, the transfer function of the thermal subsystem is

$$\frac{\bar{\Theta}}{\bar{P}} = \frac{1}{C_{th}(\bar{s} + \frac{1}{R_{th}C_{th}})}$$

For a step input of heat power of magnitude P

$$\bar{P}(\bar{s}) = \frac{P}{\bar{s}}$$

The output temperature in the complex frequency $\bar{s}$ domain is

$$\bar{\Theta}(\bar{s}) = \frac{P}{C_{th}(\bar{s} + \frac{1}{R_{th}C_{th}})\bar{s}}$$

Expanding by partial fractions

$$\bar{\Theta}(\bar{s}) = PR_{th}(\frac{1}{\bar{s}} - \frac{1}{(\bar{s} + \frac{1}{R_{th}C_{th}})})$$

Response in time domain can be obtained by taking the inverse Laplace transform.

$$\theta(t) = PR_{th}[1 - \exp(\frac{1}{R_{th}C_{th}})]$$

The steady state temperature is then

$$\theta_f = PR_{th} = i^2 R_{arc} R_{th}$$

The block diagram for the open-loop system is shown in Figure 2.

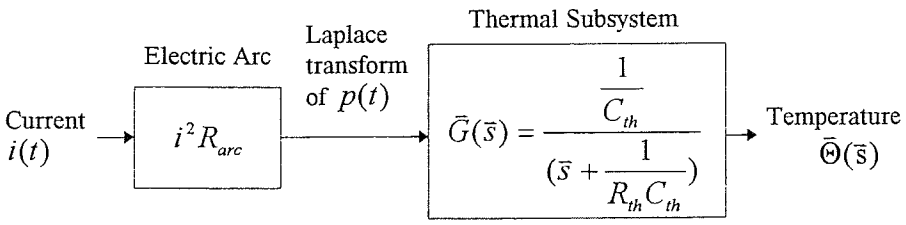


Figure 2. Block diagram for the open-loop system.

Because R_{arc} is an unknown quantity, for modelling purposes it is convenient to lump this with the thermal subsystem. The open-loop system so modified is shown in Figure 3.

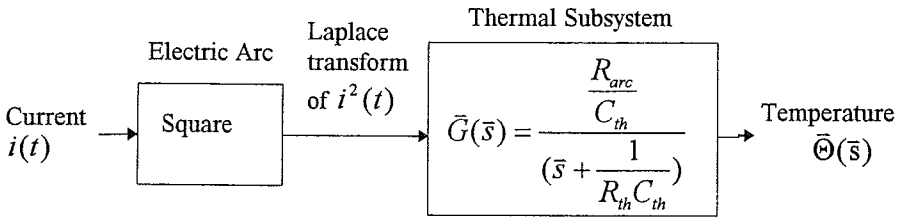


Figure 3. Modified block diagram for the open-loop system.

The transfer function for the thermal subsystem, modified to include the resistance of the electric arc R_{arc} , from now onwards will be referred as

$$\bar{G}(\bar{s}) = \frac{K}{\bar{s} + a}$$

where the numerator constant $K = \frac{R_{arc}}{C_{th}}$; and the pole $a = \frac{1}{R_{th}C_{th}}$. The time constant τ for this first order subsystem is $\tau = R_{rh}C_{th}$.

5 Nominal parameters of the thermal subsystem

The nominal parameters of the model were deduced as follows. The rise time was specified to be 2 s. Therefore, $4\tau = 2$; $\tau = 0.5$ s and the pole $a = \frac{1}{\tau} = 2$ s^{-1}. At steady state $\bar{s} = 0$, and the steady state temperature $\theta_{ss} = \frac{K}{a} i^2$. For a current $i = 17$ A $\theta_{ss} = 1500$ deg centigrade is attained.

Hence, $K = \frac{\theta_{ss}}{i^2} a = (\frac{1500}{17^2})(2) = 10.38$. The nominal values for numerator 'K' and pole 'a' are taken as 10 deg C A^{-2} s^{-1} and 2 s^{-1}.

6 Feedback control

It is intended to provide automatic correction to any drift of the output temperature by means of closed-loop feedback control using a FLC in the forward path. The FLC must be fed with the error $[e(t) = \theta_r(t) - \theta_o(t)]$ between the desired temperature reference $\theta_r(t)$ and the actual output temperature $\theta_o(t)$ to provide control input $u(t)$ to the plant. Additionally, the FLC is fed with the time derivative

$\dot{e}(t)$ of this error signal to provide adequate damping when the error is near zero. This will provide facility for designing the system such that it will respond with little overshoot. This closed-loop system is shown in Figure 4.

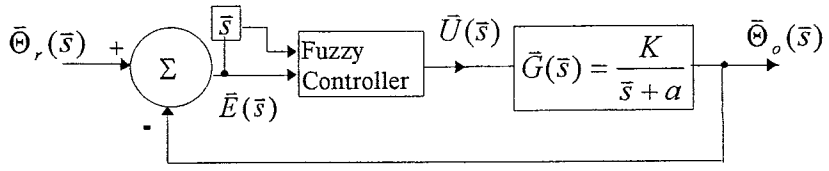


Figure 4. Block diagram for the closed-loop system $u(t) = i^2(t)$.

Since the FLC will be incorporated as a computer programme in a real-time digital computer fitted with A/D converter for input, and D/A converter for output the block diagram of Figure 4 needs to be modified to account for finite sampling time. The D/A converter is essentially a zero order hold (ZOH), and this has to be accounted for as well. This leads to Figure 5 explicitly showing sampling and ZOH.

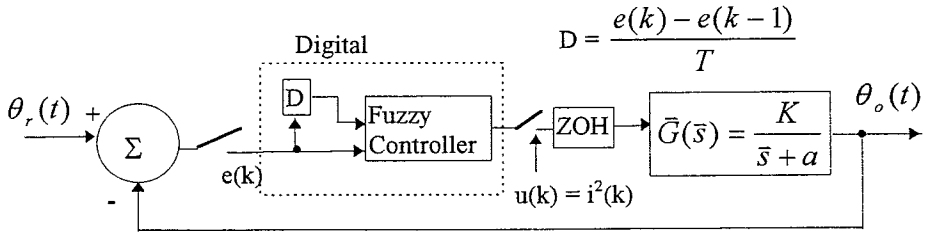


Figure 5. Feedback control incorporating a digital computer.

In the computer implementation of Figure 5 the output of the FLC is shown to be $u(k) = i^2(k)$. This is only for modelling and simulation purposes. In actual practice the FLC outputs a current demand $u(k) = i(k)$. The current generator upon receipt of this demand delivers the current to the welding arc in a time fraction of that of the sampling period T. For simulation purposes, therefore, the dynamics of the current generator can be neglected, and can be ignored for all practical purposes. It must be emphasized that such dynamics can be easily incorporated should the validity of the above assumption is under question. The unknown arcing resistance R_{arc} has be included in the numerator 'constant' K.

7 Digitizing the thermal subsystem

In order to incorporate the thermal subsystem in the simulation of the digitally controlled overall system, the time continuous model needs to be converted to its discrete time counterpart. This can be achieved either through (a) z-transforms or via (b) state-space modelling. Method (a) was chosen on grounds simplicity.

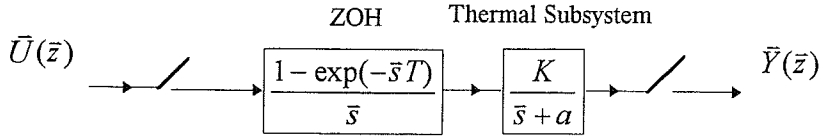


Figure 6. Thermal subsystem fed with time discrete signal.

$$\frac{\bar{Y}(\bar{z})}{\bar{U}(\bar{z})} = Z\left[\frac{1-\exp(-\bar{s}T)}{\bar{s}}\bar{G}(\bar{s})\right]$$

$$= \{1-\exp(-\bar{s}T)\}\, Z\left[\frac{\bar{G}(\bar{s})}{\bar{s}}\right]$$

Since $\exp(-\bar{s}T)$ represents unit time delay $\exp(-\bar{s}T) = \bar{z}^{-1}$.

Hence, $$\frac{\bar{Y}(\bar{z})}{\bar{U}(\bar{z})} = (1-\bar{z}^{-1})\, Z\left[\frac{\bar{G}(\bar{s})}{\bar{s}}\right]$$

$$= (1-\bar{z}^{-1})\, Z\left[\frac{K}{\bar{s}(\bar{s}+a)}\right]$$

Expanding by partial fractions

$$\frac{\bar{Y}(\bar{z})}{\bar{U}(\bar{z})} = (1-\bar{z}^{-1})\, Z\left[\frac{K/a}{\bar{s}} - \frac{K/a}{\bar{s}+a}\right]$$

$$= \frac{K}{a}(1-\bar{z}^{-1})\, Z\left[\frac{1}{\bar{s}} - \frac{1}{\bar{s}+a}\right]$$

$$= \frac{K}{a}\left(\frac{\bar{z}-1}{\bar{z}}\right)\left[\frac{\bar{z}}{\bar{z}-1} - \frac{\bar{z}}{\bar{z}-\exp(-aT)}\right]$$

$$= \frac{K}{a}\left[1 - \frac{\bar{z}-1}{\bar{z}-\exp(-aT)}\right]$$

$$= \frac{K}{a}\frac{1-\exp(-aT)}{\bar{z}-\exp(-aT)}$$

From above the discrete time description of the thermal subsystem can be derived as follows.

$$[\bar{z}-\exp(-aT)]\bar{Y}(\bar{z}) = \frac{K}{a}[1-\exp(-aT)]\bar{U}(\bar{z})$$

$$\bar{z}\bar{Y}(\bar{z}) - \exp(-aT)\bar{Y}(\bar{z}) = \frac{K[1-\exp(-aT)]}{a}\bar{U}(\bar{z})$$

The model of the thermal subsystem in discrete time domain is then

$$y(k+1) = \{\exp(-aT)\}y(k) + \frac{K[1-\exp(-aT)]}{a}u(k)$$

The time constant of the thermal subsystem is $\tau = 0.5$ s. The sampling period is chosen to be $T = \frac{\tau}{50} = \frac{0.5}{50} = 0.01$ s.

8 Membership functions

Fuzzy control rules make use of fuzzy membership functions which are discussed in this section. For input there are two sets of membership functions which fuzzyfies the crisp values of the error itself, and the time rate of change of error. For the output there is only one set of singleton with distinctly different mean values used for defuzzification to crisp values.

8.1 Membership functions for error

The membership functions are as shown in Figures 7, 8 and 9.

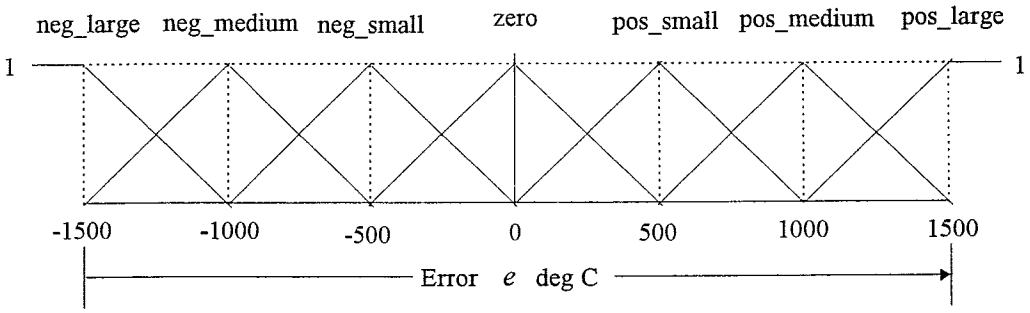


Figure 7. Membership functions for fuzzyfying crisp error e

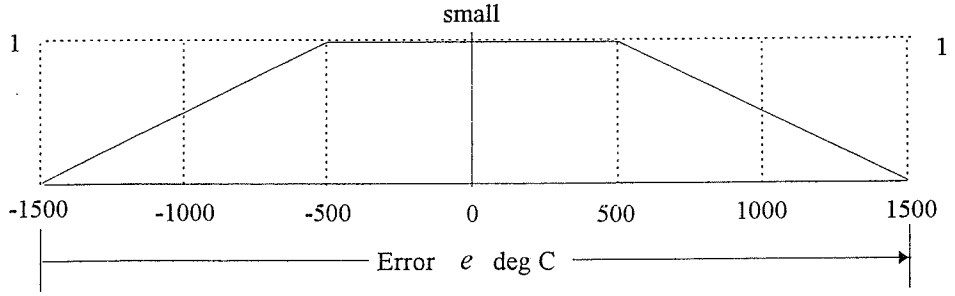


Figure 8. Membership functions for fuzzyfying crisp error e

8.2 Membership functions for rate of change of error

The membership functions are shown in Figure 9.

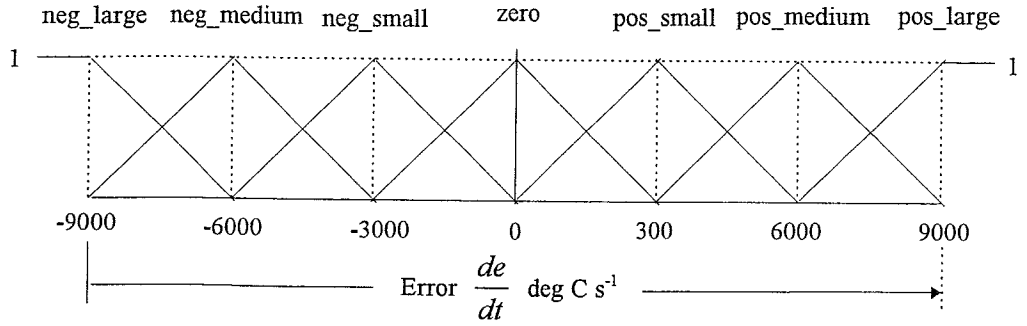


Figure 9. Membership functions for fuzzyfying crisp error rate of change $\frac{de}{dt}$

8.3 Membership functions for output fuzzy sets

These are singletons as shown in Figure 10 below.

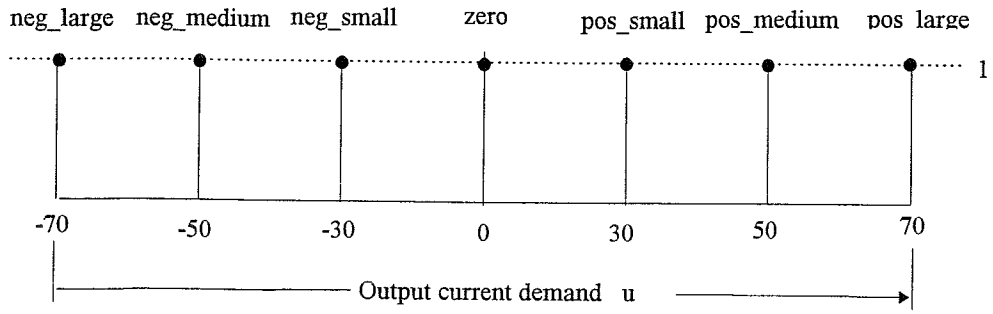


Figure 10. Membership functions for obtaining crisp output u from the controller.

9 Fuzzy control rules

Two variety of rules are incorporated - proportional (P) rules and differential (D) rules. Together they provide something like P + D type control action albiet nonlinear in nature [6].

9.1 Proportional rules

The following rules generate output only due to the error itself; therefore can be considered similar in effect to the proportional controller.

Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 4: If *error* is *zero* then *u* is *zero*

Rule 5: If *error* is *neg_small* then *u* is *zero*

Rule 6: If *error* is *neg_medium* then *u* is *zero*

Rule 7: If *error* is *neg_large* then *u* is *zero*

9.2 Differential rules

The following rules develop output due to rate of change of error when the error itself is small. Thus, they provide damping to reduce or all together eliminate overshoot.

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u* is *zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

10 Defuzzified output

Let x_1 and x_2 be the crisp values for the two inputs to the FLC. They are fuzzified using the membership functions discussed in Section 8 to indicate the degree A_1 and A_2 to which x_1 and x_2 respectively belong to the class labelled by the linguistic description. The i_th fuzzy control rule can be stated as follows.

Rule i: If x_1 is $A_{i,1}$ and x_2 is $A_{i,2}$ then u is B_i

where u is the output; $A_{i,1}$, $A_{i,2}$ are the input membership functions, and B_i is the output membership function appearing in the i_th rule.

The degree α_i to which the antecedent of the i_th rule is satisfied is given by

$$\alpha_i = \min\{A_{i,1}(x_1), A_{2,i}(x_2)\}$$

The value α_i is the degree of fulfillment of the i_th rule.

The crisp value of the controller output is obtain from $u = \dfrac{\sum_{i=1}^{n} w_i \alpha_i B_i^d}{\sum_{i=1}^{n} w_i \alpha_i}$ where B_i^d is the defuzzified value of the output membership function B_i.

For the singletons shown in Figure 10 B_i^d s correspond to the abscissa. The weights w_i are assigned to each rule to indicate its relative importance. In this investigation weights were chosen from inspired guess work, and from a number of simulation runs. They are

$$w_1 = 1, w_2 = 1, w_3 = 10, w_4 = 1, w_5 = 1, w_6 = 1, w_7 = 1, w_8 = 0.1, w_9 = 0.1, w_{10} = 0.1,$$
$$w_{11} = 0.1, w_{12} = 0.1, w_{13} = 0.1 \text{ and } w_{14} = 0.1$$

Further, only positive output from the FLC was allowed, and any negative output was simply suppressed. This is because in the context of electro-thermal power negative current is meaningless - negative current does not produce cooling as implied by the FLC.

11 Results of simulation

The performance of the FLC was extensively studied by simulation. All simulations were carried out for a sampling period $T = 0.01$ s for the following operating conditions.

(a) The workpiece was at room temperature, and the plant parameters were set at their nominal values of $K = 10$ and $a = 2$. A step change in the demanded reference temperature $\theta_r(t) = 1500$ deg C was applied to the controller. The response of the closed-loop system is shown in Graph 1. From this graph the initial current demanded by the controller was approximately $\sqrt{4900} = 70$ A. Graph 2 is identical to Graph 1 except for much reduced number of samples only to highlight the initial part of the response.

(b) The ability of the system to track a reference signal was studied next. The plant parameters were set at the above nominal values, and a sinusoidal variation of temperature $\theta_r(t) = 1000 + 500\sin(2\pi ft)$ deg C with a frequency of 0.2 Hz was demanded. Graph 3 clearly shows that the closed-loop system was able to respond to this demand with small error.

(c) The final study involved use of random values for the plant parameters between specified upper and lower bounds on them. The maximum values for K and a were set at $K_{\max} = 20$ and $a_{\max} = 4$ respectively. The corresponding minimum values were chosen to be $K_{\min} = 5$ and $a_{\min} = 1$. The values of the randomly generated plant parameters are given in Table 1. For each of these randomly generated set, the closed system was simulated for the same demanded sinusoidal variation of reference temperature as in (b) above. Graph 4 shows the behaviour of the error between the reference temperature and the plant output. It can be seen that although stability is always maintained by the closed-loop system, the error increases with the increase in the numerical value of the pole 'a'. The sensitivity of this error is, however, much lower for the other plant parameter K. Further observations on this aspect are contained in Section 14: Recommendations and Section 15: Conclusions.

12 Computer programmes

Two computer programmes were developed in the popular C-language. Although both programmes have a common core, they differ in the demanded temperature variation, and in the choice of plant parameters. Additionally, they differ in the way and the number of output files created.

The programme 'weld1.c' prompts the user to input the nominal parameters of the thermal subsystem. It then simulates the performance of the closed-loop system when a step change in temperature of 1500 deg C is demanded from the system. The programme opens a file 'weld1.dat' in the current directory and writes the simulation data in this file. The programme is given in Appendix - A.

The programme 'weld2.c' prompts the user to input the upper and the lower bounds for the two plant parameters. It also asks for the number of data files to be created to contain simulation data for each of the randomly generated parameter sets. The programme simulates the tracking behaviour of the closed-loop system to a sinusoidal variation of 0.2 Hz in the demanded temperature as discussed in Section 11 (b). This programme is contained in Appendix - B.

13 Stability assessment

Stability of the overall controlled system is assessed using Lyapunov's procedure. For this the model for the error dynamics need to be first established.

13.1 Error dynamics

The dynamic model for the thermal subsystem in descrete time domain, as shown in Section 7, is

$$y(k+1) = \{\exp(-aT)\}y(k) + \frac{K[1-\exp(-aT)]}{a}u(k)$$

For nominal values of $K = 10$ and $a = 2$

$$y(k+1) = 0.98020y(k) + 0.099007u(k)$$

Hence, error

$$x_1(k) = r(k) - y(k)$$

$$x_1(k+1) = r(k+1) - y(k+1)$$

$$= r(k+1) - 0.98020y(k) - 0.099007u(k)$$

But $y(k) = r(k) - x_1(k)$

Therefore, $x_1(k+1) = r(k+1) - 0.98020r(k) + 0.98020x_1(k) - 0.099007u(k)$

For constant $r(k) = R$

$$x_1(k+1) = 0.98020x_1(k) - 0.099007u(k) + 0.0198R$$

Table 1: Randomly generated parameters of the plant $\bar{G}(\bar{s}) = \dfrac{K}{\bar{s}+a}$

Nominal values $K = 10$ and $a = 2$

$K_{max} = 20$ and $a_{max} = 4$ $\quad\quad$ $K_{min} = 5$ and $a_{min} = 1$

File names	Plant Parameters	
	Numerator constant K	Denominator Pole a
fu_21_00	5.1584	1.0119
fu_21_01	10.0273	1.0998
fu_21_03	10.3359	1.6516
fu_21_04	13.0546	1.5873
fu_21_06	15.5051	3.8498
fu_21_07	9.1218	2.3329
fu_21_09	6.6347	3.0947
fu_21_10	13.4652	1.1245
fu_21_12	7.4775	3.4465
fu_21_13	15.2831	3.2931
fu_21_15	17.4145	3.8786
fu_21_16	8.2914	2.2804
fu_21_18	19.2891	3.5186
fu_21_19	18.8487	3.4327
fu_21_21	11.7655	2.8143
fu_21_22	14.9251	2.7992
fu_21_24	13.2377	3.1601
fu_21_25	6.7093	2.2186
fu_21_27	6.8215	3.0138
fu_21_28	12.1235	2.4733

Rate of change of error

$$x_2(k+1) = \frac{x_1(k+1) - x_1(k)}{T}$$

$$= \frac{1}{0.01}(-0.01980x_1(k) - 0.099007u(k) + 0.01980R)$$

$$= -1.980x_1(k) - 9.9007u(k) + 1.98R$$

In vector-matrix form the error dynamics is then

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.98020 & 0 \\ -1.98 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{pmatrix} -0.099007 \\ -9.9007 \end{pmatrix} u(k) + \begin{pmatrix} 29.70 \\ 2970 \end{pmatrix}$$

We can write the above as $\underline{x}(k+1) = \underset{\sim}{A}\,\underline{x}(k) + \underline{B}\,u(k) + \underline{E}$

13.2 Lyapunov criteria for global asymptotic stability

We next define a Lyapunov function $V(k) = \underline{x}^T(k)\,\underset{\sim}{P}\,\underline{x}(k)$ where $\underset{\sim}{P}$ is a symmetrical positive definite matrix. Then $V(k+1) = \underline{x}^T(k+1)\,\underset{\sim}{P}\,\underline{x}(k+1)$ and Lyapunov's criterion for global asymptotic stability is $\Delta V = V(k+1) - V(k) < 0$.

Hence, $$\Delta V = \underline{x}^T(k)[\underset{\sim}{A}^T\,\underset{\sim}{P}\,\underset{\sim}{A} - \underset{\sim}{P}]\underline{x}(k) + 2u(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underset{\sim}{A}\,\underline{x}(k)$$
$$+u^2(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underline{B} + \underline{E}^T\,\underset{\sim}{P}[2\{\underset{\sim}{A}\,\underline{x}(k) + \underline{B}\,u(k)\} + \underline{E}]$$

The criterion for global asymptotic stability then becomes

$$\underline{x}^T(k)[\underset{\sim}{P} - \underset{\sim}{A}^T\,\underset{\sim}{P}\,\underset{\sim}{A}]\underline{x}(k) > 2u(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underset{\sim}{A}\,\underline{x}(k) + u^2(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underline{B}$$
$$+\underline{E}^T\,\underset{\sim}{P}[2\{\underset{\sim}{A}\,\underline{x}(k) + \underline{B}\,u(k)\} + \underline{E}]$$

or $$\frac{\underline{x}^T(k)[\underset{\sim}{P} - \underset{\sim}{A}^T\,\underset{\sim}{P}\,\underset{\sim}{A}]\underline{x}(k)}{\underline{x}^T\,\underline{x}} > \frac{2u(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underset{\sim}{A}\,\underline{x}(k)}{\underline{x}^T\,\underline{x}} + \frac{u^2(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underline{B}}{\underline{x}^T\,\underline{x}}$$
$$+\frac{\underline{E}^T\,\underset{\sim}{P}[2\{\underset{\sim}{A}\,\underline{x}(k) + \underline{B}\,u(k)\} + \underline{E}]}{\underline{x}^T\,\underline{x}}$$

A more stringent (conservative) criterion than the above is

$$\text{Min eigen}[\underset{\sim}{P} - \underset{\sim}{A}^T\,\underset{\sim}{P}\,\underset{\sim}{A}] > \frac{2u(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underset{\sim}{A}\,\underline{x}(k)}{|x|^2} + \frac{u^2(k)\,\underline{B}^T\,\underset{\sim}{P}\,\underline{B}}{|x|^2}$$
$$+\frac{\underline{E}^T\,\underset{\sim}{P}[2\{\underset{\sim}{A}\,\underline{x}(k) + \underline{B}\,u(k)\} + \underline{E}]}{|x|^2}$$

If either of the last two criteria, stated above, holds for every region of the state space, then the total controlled system will be globally asymptotically stable.

We chose $\underset{\sim}{P} = \begin{pmatrix} 138 & 0 \\ 0 & 1 \end{pmatrix}$ so that the eigen-values of the matrix $(\underset{\sim}{P} - \underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A})$ are real, distinct and around unity. The above $\underset{\sim}{P}$ will yield eigenvalues 1 and 1.49.

13.3 Calculation for numerical values of $\underset{\sim}{P}$

Let the Lyapunov weighting matrix be $\underset{\sim}{P} = \begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix}$.

x should be so chosen that the matrix $-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} + \underset{\sim}{P}$ has one unity eigenvalue, and the other eigenvalue near unity, say 1.5.

$$\underset{\sim}{P}\,\underset{\sim}{A} = \begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0.9802 & 0 \\ -1.98 & 0 \end{pmatrix} = \begin{pmatrix} 0.9802x & 0 \\ -1.98 & 0 \end{pmatrix}$$

$$-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} = -\begin{pmatrix} 0.9802x & -1.98 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0.9802x & 0 \\ -1.98 & 0 \end{pmatrix} = \begin{pmatrix} -(0.9802)^2 x - (1.98)^2 & 0 \\ 0 & 0 \end{pmatrix}$$

Hence, $-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} + \underset{\sim}{P} = \begin{pmatrix} -0.9608x - 3.9204 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} x & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.03921x - 3.9204 & 0 \\ 0 & 1 \end{pmatrix}$

Next, we derive the eigenvalues of $-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} + \underset{\sim}{P}$. The characteristic equation is

$$\left| \bar{s}\,\underset{\sim}{I} - (-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} + \underset{\sim}{P}) \right| = 0$$

or $\begin{vmatrix} \bar{s} - 0.03921x + 3.9204 & 0 \\ 0 & \bar{s} - 1 \end{vmatrix} = 0$ or $(\bar{s} - 1)(\bar{s} - 0.03921x + 3.9204) = 0$

The above yields either $(\bar{s} - 1) = 0$ or $(\bar{s} - 0.03921x + 3.9204) = 0$. Since one of the eigenvalue is specified to be 1.5

$$0.03921x - 3.9204 = 0 \quad \text{Hence,} \quad x \approx 138.$$

In fact, for $\underset{\sim}{P} = \begin{pmatrix} 138 & 0 \\ 0 & 1 \end{pmatrix}$, the eigenvalues of $-\underset{\sim}{A}^T \underset{\sim}{P}\,\underset{\sim}{A} + \underset{\sim}{P}$ are $\lambda_1 = 1$ and $\lambda_2 = 1.49$.

13.4 State-space partitions and associated bounds on control

The state-space is first divided into cells, Figure 11, and each contain the respective upper and lower bounds on the control output. A specimen calculation for Cell 3-4 is shown below, and all other calculations can be found in Appendix - C.

Maximum value of u : This is obtained at the bottom right-hand corner of the cell.

The relevent rules for this cell are as follows.

Rule 4: If *error* is *zero* then *u* is *zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_4)(\alpha_4) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(1)(30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{3}{1.1} = 2.73$$

Minimum value of u : This is obtained at the top left-hand corner of the cell.

The relevent rules for this cell are:

Rule 5: If *error* is *neg_small* then *u* is *zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u* is *zero*

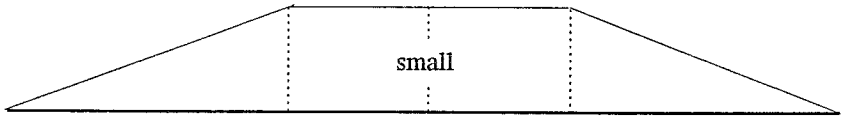
$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_5)(\alpha_5) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(1)(0)}{(1)(1) + (0.1)(1)}$$

$$= \frac{0}{1.1} = 0$$

Less stringent of the two Lyapunov's criteria mentioned in Section 13.2 has been used to check stability. Unfortunately, the criterion could not be satisfied. However, the results of simulation indicate that the FLC so designed is complely stable with no overshoot. This apparent anomaly is due to the fact that the control effort $u(k)$ applied at the beginning of a sampling period results in such an overwhemlingly large increase in the magnitude of the rate of change of error $x_2(k)$ that V(k+1) > V(k), and Lyapunov's criterion is violated. It must be borne in mind that Lyapunov's criterion is only a sufficiency condition for stability, and not a necessary condition. Therefore, although the present design fails to meet this criterion, nevertheless the error $x_1(k)$ decreases monotonically to a small value. If the design method used ensured that the Lyapunov's criterion is always satisfied in every cell, the result would be a design which is much softer in its application of control effort $u(k)$. The resulting rate of change of error $x_2(k)$ will have a maximum magnitude such that V(k+1) is always less than V(k), thus satisfying the Lyapunov's criterion at each and every point of the state-space. This means that although the controller will be comparatively sluggish, but a lower power actuator would suffice to generate the control effort. Thus, the overall design will be that much economical.

!3 Recommendation

(a) The FLC used in this study should be implemented in the computer controlling the welding machine. This should not pose any difficulty. All that is required is to load the code for the controller from any one of the two programmes contained in Appendices A and B. A first order digital filter with a break frequency of 10 Hz would, however, be required to filter out any inevitable high frequency noise present in the measurement of the output temperature. The dynamic performance of the controller should be measured, from an experimental rig, using any data logging equipment or a PC fitted with an A/D card. Experimentally measured performance, thus obtained, should then be compared with those obtained by simulation. If there is a substantial departure then attention should be given towards obtaining more realistic values for the plant parameter set K and a.



Error e

Error_rate $\frac{de}{dt}$ \ e	neg_large -1500	neg_medium -1000	neg_small -500	zero 0	pos_small 500	pos_medium 1000	pos_la 15(
neg_large -9000	0	0	0	29.2 0	45.2 29.0	70.0 44.3	
neg_medium -6000	0	0	0	29.4 0	46.2 29.2	70.0 45.2	
neg_smal -3000	0	0	0	29.7 0	47.6 29.4	70.0 46.2	
zero 0	1.43 0	2.73 0	2.73 0	30.0 0	49.1 29.7	70.0 47.6	
pos_small 3000	2.38 0	4.55 1.43	4.55 2.73	30.2 2.73	50.0 30.0	70.0 49.1	
pos_medium 6000	3.30 0	6.36 2.38	6.36 4.55	30.4 4.55	51.0 30.2	70.0 50.0	
pos_large 9000							

Figure 11. Bounds on control force in individual cells within state-space.

These parameters can be defined as functions of time to represent actual working conditions as the arcing spot travels up and down the workpiece. Further simulations should then carried out with these functions of time. Only when the differences in performance between those obtained experimentally, and those obtained by simulation have fallen below an acceptable threshold, it could be

obtained by simulation have fallen below an acceptable threshold, it could be concluded that the behaviour of welding plant is fully understood, and the model is usable for further research.

(b) The FLC should be loaded on to a PC containing A/D and D/A cards for closed-loop control of a practical system. This can be accomplished by removing the digital model of the thermal system from the C - code presented in Appendix - A. Further, codes have to be inserted for data acquisition and output, and this can be easily completed within a couple days.

(c) The weights on the rules and the shapes of the input fuzzy sets, Figures 7, 8 and 9, have been chosen on a pure guess work basis. The same statement is true for the various abscissa values. This also applies to the base values for the output singletons of Figure 10. It can, therefore, be seen that there are a number of design elements associated with a FLC - (i) input fuzzy sets with membership shape, (ii) output fuzzy sets, (iii) control rules of the form IF ... then ..., (iv) weights on the control rules. This leads to a large number of design variables which can be scientifically chosen so as to optimise the performance of the closed-loop system. The author of this report has developed a new optimisation algorithm EVOP [7] ideally suited for such an application. The following shape, Figure 12, for an input fuzzy set is recommended, and if required the number of linear segments can be increased at will.

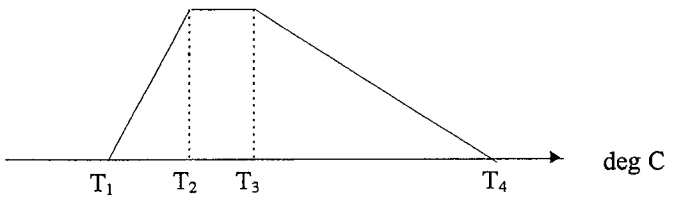


Figure 12. Recommended shape of input fuzzy sets.

While optimising the closed loop performance stability issues must not be overlooked. The author's programme 'evop' can directly deal with explicit constraints on the design variables, and also constraints which are implicit in nature. As shown in Section 13.2 stability requirements can be recast as constraints. Since the algorithm ensures that no constraint can ever be breached, the optimum design vector will automatically satisfy the stability requirement.

(d) Finally, real-time optimisation of the FLC parameters should be investigated. It should be possible to train an artificial neural network, over a period of time, with the values of the optimum parameter sets so that gradually less emphasis is placed on the usage of numerical optimisation algorithm in real-time. In other words the system will learn to progressively improve its performance with passage of time. The author has developed his optimisation algorithm [7] with a view to cope with such real-time applications. The beauty of the algorithm is that such optimisation in real-time will always be conducted in complete safety. The plant operation is guaranteed to be within the safe operating region, and no constraint can ever be violated. Such an adaptive FLC should be a major contribution in the arena of control engineering.

15 Conclusions

This investigation, using simulation, has shown that closed-loop temperature control of a welding machine using a FLC is entirely feasible. A dynamic model of the thermal subsystem comprising of the welding arc and the workpiece, in discrete time domain, has been developed for simulation studies. In practice, as the electric arc travels over the workpiece, the dynamic parameters of the thermal subsystem varies over a wide range. It has been shown that synthesis of a FLC does not require explicit knowledge of the dynamic parameters of this thermal subsystem. Control effort is generated simply by monitoring the dynamic behaviour of the output variable. To study stability the plant parameters have been chosen randomly within an upper and a lower limit twice and half that of the nominal values, respectively. The results of simulations indicate that the design is robust enough to cope with such wide variations of the parameters. Main elements of FLC design have been identified, and a route for optimising the design off-line has been proposed. Optimisation in real-time has also been recommended, and further research towards the development of a Neuro-FLC has been advocated. Recommendations (a) and (b) are for immediate attention, whereas (c) and (d) are topics for further research in optimisation of FLC s.

Stability studies using Lyapunov's method has been conducted. Although the results of simulations indicate that the FLC so designed is completely stable within ± 200 percent variations over the nominal parameter values, it fails to satisfy the Lyapunov's stability criterion. Nominal values of the parameters were used, and no further attempt was made to repeat the test using worst case parmeter combination. It was considered, because of practical reasons mentioned earlier, that stability studies using computer simulation would be more fruitful and closer to reality at this stage. For safety critical applications it is imperative to satisfisfy the stringent Lyapunov's global stability issue. The author has proposed a way forward, and would require further time and resources to address this issue. Because a FLC was not available at RRIRD to conduct stability analysis forthwith, this had to be designed at the outset, and this is where the present effort lies. The stability problem has not been overlooked. It has been checked via a large number of simulations. The advantage of following the above procedure is that worst parameter combinations, from stability point of view, are available from simulations using randomly generated plant parameters.

16 References:

(1) P E Wellstead and M B Zarrop: 'Self-Tuning Systems - Control and Signal Processing', Wiley, 1991, ISBN 0 471 93054-7.

(2) Yoan Landau: `Adaptive Control - A Model Reference Approach', Marcel Dekker.

(3) Li-Xin Wang: `Adaptive Fuzzy Systems and Control - Design and Stability Analysis', PTR Prentice Hall, 1994, ISBN 0-13-099631-9.

(4) J Yen, R Langari and L A Zadeh: `Industrial Applications of Fuzzy Logic and Intelligent Systems', IEEE Press, 1995, ISBN 0-7803-1048-9.

(5) G F Franklin, J D Powell and M L Workman: `Digital Control of Dynamic Systems', Addison-Wesley, 1990, ISBN 0-201-51884-8.

(6) S Chiu and S Chand: 'Fuzzy Controller Design and Stability Analysis for an Aircraft Model', Journal unknown.

(7) S N Ghani: `Performance of Global Optimisation Algorithm EVOP for Non-linear Non-differentiable Constrained Objective Functions', 1995 IEEE International Conference on Evolutionary Computation (ICEC'95), Perth, Western Australia, 29th Nov - 1st Dec 1995, Volume 1, pp 320 - 325.

APPENDIX - A

Computer programme 'weld1.c'

```
#include <stdio.h>
#include <math.h>
/* #define PI 3.14159265358979323846  */
void main(void);

void main(void)
{
 FILE *outfile_ptr;
 double u, a, T, phi, x, K, r, error, error_rate;
 double pos_large_error, pos_medium_error;
 double pos_small_error, zero_error, neg_small_error;
 double neg_medium_error, neg_large_error, weight;
 double pos_large_error_rate, pos_medium_error_rate;
 double pos_small_error_rate, zero_error_rate, neg_small_error_rate;
 double neg_medium_error_rate, neg_large_error_rate, denom;
 double spler, spmer, spser, szer, snser, snmer, snler;
 double small_error;
 int k;
   printf("                    INPUT DATA FOR THE FUZZY CONTROLLER
SIMULATOR\n\n\n");
 printf("\> Enter nominal value for the plant numerator gain constant K = ");
 scanf("%lf", &K);
 printf("\> Enter nominal value for the plant denominator constant a = ");
 scanf("%lf", &a);
 if ( (outfile_ptr = fopen("weld1.dat", "w")) == NULL)
        {
        printf("Could not open output file \n");
        exit(1);
```



```
        }
        fprintf(outfile_ptr, "Nominal value for the numerator gain constant K =
%10.4f\n",
                                                K);
        fprintf(outfile_ptr, "Nominal value for the denominator constant a =
%10.4f\n",
                                                a);
        T = 0.01;
        phi = exp(-a * T);
        x = 0.0;
        error_rate = 0.0;
        for (k = 0; k <= 200; k+= 1)
                {
                r = 1500.0;
                error = (r - x);
                                                        /*      Membership
functions for error*/
                if (error >= 1500.0)
                        {
                                pos_large_error = 1.0;
                                pos_medium_error = 0.0;
                                pos_small_error = 0.0;
                                zero_error = 0.0;
                                neg_small_error = 0.0;
                                neg_medium_error = 0.0;
                                neg_large_error = 0.0;
                                small_error = 0.0;
                        }
                else if (error >= 1000.0)
                        {
                                pos_large_error = -2.0 + (1.0/500.0) * error;
                                pos_medium_error = 3.0 - (1.0/500.0) * error;
                                pos_small_error = 0.0;
                                zero_error = 0.0;
```



```
                        neg_small_error = 0.0;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 1.5 - (1.0/1000.0) * error;
                }
        else if (error >= 500.0)
                {
                        pos_large_error = 0.0;
                        pos_medium_error = -1.0 + (1.0/500.0) * error;
                        pos_small_error = 2.0 - (1.0/500.0) *error;
                        zero_error = 0.0;
                        neg_small_error = 0.0;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 1.5 - (1.0/1000.0) * error;
                }
        else if (error >= 0.0)
                {
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = (1.0/500.0) *error;
                        zero_error = 1.0 - (1.0/500.0) * error;
                        neg_small_error = 0.0;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 1.0;
                }
        else if (error >= -500.0)
{
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 1.0 + (1.0/500.0) * error;
```



```
                        neg_small_error = - (1.0/500.0) * error;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 1.0;
                }
        else if (error >= -1000.0)
{
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 2.0 + (1.0/500.0) * error;
                        neg_medium_error = -1.0 - (1.0/500.0) * error;
                        neg_large_error = 0.0;
                        small_error = 1.5 + (1.0/1000.0) * error;
                }
        else if (error >= -1500.0)
                {
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 0.0;
                        neg_medium_error = 3.0 + (1.0/500.0) * error;
                        neg_large_error = -2.0 - (1.0/500.0) * error;
                        small_error = 1.5 + (1.0/1000.0) * error;
                }
        else
                {
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
```



```
                    neg_small_error = 0.0;
                    neg_medium_error = 0.0;
                    neg_large_error = 1.0;
                    small_error = 0.0;
                }


                                        /* Membership functions for error_rate*/
            if (error_rate >= 9000.0)
                {
                    pos_large_error_rate = 1.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = 0.0;
                    zero_error_rate = 0.0;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= 6000.0)
                {
                    pos_large_error_rate = -2.0 + (1.0/3000.0) * error_rate;
                    pos_medium_error_rate = 3.0 - (1.0/3000.0) *
error_rate;
                    pos_small_error_rate = 0.0;
                    zero_error_rate = 0.0;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= 3000.0)
                {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = -1.0 + (1.0/3000.0) *
error_rate;
```



```
                    pos_small_error_rate = 2.0 - (1.0/3000.0) *error_rate;
                    zero_error_rate = 0.0;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
            }
        else if (error_rate >= 0.0)
            {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = (1.0/3000.0) *error_rate;
                    zero_error_rate = 1.0 - (1.0/3000.0) * error_rate;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
            }
        else if (error_rate >= -3000.0)
            {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = 0.0;
                    zero_error_rate = 1.0 + (1.0/3000.0) * error_rate;
                    neg_small_error_rate = - (1.0/3000.0) * error_rate;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
            }
        else if (error_rate >= -6000.0)
{
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = 0.0;
                    zero_error_rate = 0.0;
                    neg_small_error_rate = 2.0 + (1.0/3000.0) * error_rate;
```



```
                        neg_medium_error_rate = -1.0 - (1.0/3000.0) *
error_rate;
                        neg_large_error_rate = 0.0;
                }
            else if (error_rate >= -9000.0)
                {
                        pos_large_error_rate = 0.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 3.0 + (1.0/3000.0) *
error_rate;
                        neg_large_error_rate = -2.0 - (1.0/3000.0) * error_rate;
                }
            else
                {
                        pos_large_error_rate = 0.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 0.0;
                        neg_large_error_rate = 1.0;
                }
    if (small_error >= pos_large_error_rate) spler = pos_large_error_rate;
            else spler = small_error;
            if (small_error >= pos_medium_error_rate) spmer =
pos_medium_error_rate;
            else spmer = small_error;
            if (small_error >= pos_small_error_rate) spser =
pos_small_error_rate;
            else spser = small_error;
            if (small_error >= zero_error_rate) spmer = zero_error_rate;
            else szer = small_error;
```



```
                if (small_error >= neg_small_error_rate) snser =
neg_small_error_rate;
                else snser = small_error;
                if (small_error >= neg_medium_error_rate) snmer =
neg_medium_error_rate;
                else snmer = small_error;
                if (small_error >= neg_large_error_rate) snler = neg_large_error_rate;
                else snler = small_error;
                weight = 0.1;
                u = pos_large_error * 70.0 + pos_medium_error * 50.0;
                u = u + 5.0 * pos_small_error * 30.0 + zero_error * 0.0;
                u = u + neg_small_error * 0.0 + neg_medium_error * 0.0;
                u = u + neg_large_error * 0.0;
                u = u + weight * (spler * 70.0 + spmer * 50.0 + spser * 30.0);
                u = u + weight * (szer * 0.0 + snser * -30.0 + snmer * -50.0);
                u = u + weight * (snler * -70.0);
                denom = pos_large_error + pos_medium_error + 5.0 *
pos_small_error;
                denom = denom + zero_error + neg_small_error +
neg_medium_error;
                denom = denom + neg_large_error;
                denom = denom + weight * (spler + spmer + spser + szer + snser +
                        snmer + snler);
                u = u / denom;
                if (u) u = u * u;
                else u = 0.0;
                fprintf(outfile_ptr, "%10d %10.4lf %10.4lf %10.4lf %10.4lf %10.4lf\n",
                k, u, r, x, error, error_rate);
                x = phi * x + K * (1.0 - phi)* u/a;
                error_rate = (r - x - error)/T;
                }
        fclose(outfile_ptr);
}
```


APPENDIX - B

Computer programme 'weld2.c'

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <dos.h>
#define PI 3.14159265358979323846

void main(void)
{
 FILE *outfile_ptr;
 double u, a, T, phi, x, K, r, error, error_rate;
 double pos_large_error, pos_medium_error;
 double pos_small_error, zero_error, neg_small_error;
 double neg_medium_error, neg_large_error, weight;
 double pos_large_error_rate, pos_medium_error_rate;
 double pos_small_error_rate, zero_error_rate, neg_small_error_rate;
 double neg_medium_error_rate, neg_large_error_rate, denom;
 double spler, spmer, spser, szer, snser, snmer, snler;
 double small_error, K_min, K_max, a_min, a_max;
 int k, j, no_of_data_files;
 time_t t;

 srand(1);
   printf("                    INPUT DATA FOR THE FUZZY CONTROLLER
SIMULATOR\n\n\n");
 printf("\> Enter maximum value for the plant numerator gain constant K = ");
 scanf("%lf", &K_max);
```



```
printf("\> Enter minimum value for the plant numerator gain constant K = ");
scanf("%lf", &K_min);
printf("\> Enter maximum value for the plant denominator constant a = ");
scanf("%lf", &a_max);
printf("\> Enter minimum value for the plant denominator constant a = ");
scanf("%lf", &a_min);
printf("\> Enter required number of data files 'no_of_data_files' = ");
scanf("%d", &no_of_data_files);
for (j=0; j < no_of_data_files; j++)
{
char string[30];
char *f = "fu_";
char *s = string;
char tim[11];
char *p;
char *pp;
int i;


pp = tim;
time(&t);
p = ctime(&t);
p += 14;
while (*pp++ = *p++);
pp = tim;
pp += 5;
for (i=0; i < 6; i++) *pp++ = '\0';
pp = string;
while (*pp++ = *f++);
pp = tim;
p = string;
p += 3;
while (*p++ = *pp++);
```



```
strcat(string, ".dat");
string[5] = '_' ;
if ( (outfile_ptr = fopen(s, "w")) == NULL)
        {
        printf("Could not open output file \n");
        exit(1);
        scanf("%d", &k);
        }
      K = K_min + (K_max - K_min) * (double) rand()/RAND_MAX;
      a = a_min + (a_max - a_min) * (double) rand()/RAND_MAX;
      fprintf(outfile_ptr, "Value for the numerator gain constant K = %10.4f\n",
                                                  K);
      fprintf(outfile_ptr, "Value for the denominator constant a = %10.4f\n",
                                                  a);
      T = 0.01;
      phi = exp(-a * T);
      x = 0.0;
      error_rate = 0.0;
      for (k = 0; k <= 1000; k+= 1)
            {
            r = 1000.0 + 500.0 * sin(2.0 * PI * 0.2 *(double)k * T);
            error = (r - x);
                                        /*   Membership functions for error */
            if (error >= 1500.0)
                  {
                        pos_large_error = 1.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 0.0;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 0.0;
```



```
        }
else if (error >= 1000.0)
        {
                pos_large_error = -2.0 + (1.0/500.0) * error;
                pos_medium_error = 3.0 - (1.0/500.0) * error;
                pos_small_error = 0.0;
                zero_error = 0.0;
                neg_small_error = 0.0;
                neg_medium_error = 0.0;
                neg_large_error = 0.0;
                small_error = 1.5 - (1.0/1000.0) * error;
        }
else if (error >= 500.0)
        {
                pos_large_error = 0.0;
                pos_medium_error = -1.0 + (1.0/500.0) * error;
                pos_small_error = 2.0 - (1.0/500.0) *error;
                zero_error = 0.0;
                neg_small_error = 0.0;
                neg_medium_error = 0.0;
                neg_large_error = 0.0;
                small_error = 1.5 - (1.0/1000.0) * error;
        }
else if (error >= 0.0)
        {
                pos_large_error = 0.0;
                pos_medium_error = 0.0;
                pos_small_error = (1.0/500.0) *error;
                zero_error = 1.0 - (1.0/500.0) * error;
                neg_small_error = 0.0;
                neg_medium_error = 0.0;
                neg_large_error = 0.0;
                small_error = 1.0;
```



```
                }
        else if (error >= -500.0)
{
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 1.0 + (1.0/500.0) * error;
                        neg_small_error = - (1.0/500.0) * error;
                        neg_medium_error = 0.0;
                        neg_large_error = 0.0;
                        small_error = 1.0;
                }
        else if (error >= -1000.0)
{
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 2.0 + (1.0/500.0) * error;
                        neg_medium_error = -1.0 - (1.0/500.0) * error;
                        neg_large_error = 0.0;
                        small_error = 1.5 + (1.0/1000.0) * error;
                }
        else if (error >= -1500.0)
                {
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 0.0;
                        neg_medium_error = 3.0 + (1.0/500.0) * error;
                        neg_large_error = -2.0 - (1.0/500.0) * error;
                        small_error = 1.5 + (1.0/1000.0) * error;
```



```
                }
        else
                {
                        pos_large_error = 0.0;
                        pos_medium_error = 0.0;
                        pos_small_error = 0.0;
                        zero_error = 0.0;
                        neg_small_error = 0.0;
                        neg_medium_error = 0.0;
                        neg_large_error = 1.0;
                        small_error = 0.0;
                }


                                /*      Membership functions for error_rate*/
    if (error_rate >= 9000.0)
                {
                        pos_large_error_rate = 1.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 0.0;
                        neg_large_error_rate = 0.0;
                }
        else if (error_rate >= 6000.0)
                {
                        pos_large_error_rate = -2.0 + (1.0/3000.0) * error_rate;
                        pos_medium_error_rate   =   3.0   -   (1.0/3000.0)   *
error_rate;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 0.0;
```



```
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= 3000.0)
                {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = -1.0 + (1.0/3000.0) *
error_rate;
                    pos_small_error_rate = 2.0 - (1.0/3000.0) *error_rate;
                    zero_error_rate = 0.0;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= 0.0)
                {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = (1.0/3000.0) *error_rate;
                    zero_error_rate = 1.0 - (1.0/3000.0) * error_rate;
                    neg_small_error_rate = 0.0;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= -3000.0)
                {
                    pos_large_error_rate = 0.0;
                    pos_medium_error_rate = 0.0;
                    pos_small_error_rate = 0.0;
                    zero_error_rate = 1.0 + (1.0/3000.0) * error_rate;
                    neg_small_error_rate = - (1.0/3000.0) * error_rate;
                    neg_medium_error_rate = 0.0;
                    neg_large_error_rate = 0.0;
                }
            else if (error_rate >= -6000.0)
```



```
        {
                        pos_large_error_rate = 0.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 2.0 + (1.0/3000.0) * error_rate;
                        neg_medium_error_rate = -1.0 - (1.0/3000.0) *
error_rate;
                        neg_large_error_rate = 0.0;
                }
            else if (error_rate >= -9000.0)
                {
                        pos_large_error_rate = 0.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 3.0 + (1.0/3000.0) *
error_rate;
                        neg_large_error_rate = -2.0 - (1.0/3000.0) * error_rate;
                }
            else
                {
                        pos_large_error_rate = 0.0;
                        pos_medium_error_rate = 0.0;
                        pos_small_error_rate = 0.0;
                        zero_error_rate = 0.0;
                        neg_small_error_rate = 0.0;
                        neg_medium_error_rate = 0.0;
                        neg_large_error_rate = 1.0;
                }
    if (small_error >= pos_large_error_rate) spler = pos_large_error_rate;
            else spler = small_error;
```



```
                if (small_error >= pos_medium_error_rate) spmer =
pos_medium_error_rate;
                else spmer = small_error;
                if (small_error >= pos_small_error_rate) spser =
pos_small_error_rate;
                else spser = small_error;
                if (small_error >= zero_error_rate) spmer = zero_error_rate;
                else szer = small_error;
                if (small_error >= neg_small_error_rate) snser =
neg_small_error_rate;
                else snser = small_error;
                if (small_error >= neg_medium_error_rate) snmer =
neg_medium_error_rate;
                else snmer = small_error;
                if (small_error >= neg_large_error_rate) snler = neg_large_error_rate;
                else snler = small_error;
                weight = 0.1;
                u = pos_large_error * 70.0 + pos_medium_error * 50.0;
                u = u + 10.0 * pos_small_error * 30.0 + zero_error * 0.0;
                u = u + neg_small_error * 0.0 + neg_medium_error * 0.0;
                u = u + neg_large_error * 0.0;
                u = u + weight * (spler * 70.0 + spmer * 50.0 + spser * 30.0);
                u = u + weight * (szer * 0.0 + snser * -30.0 + snmer * -50.0);
                u = u + weight * (snler * -70.0);
                denom = pos_large_error + pos_medium_error + 10.0 *
pos_small_error;
                denom = denom + zero_error + neg_small_error +
neg_medium_error;
                denom = denom + neg_large_error;
                denom = denom + weight * (spler + spmer + spser + szer + snser +
                                snmer + snler);
                u = u / denom;
                if (u) u = u * u;
                else u = 0.0;
                fprintf(outfile_ptr, "%10d %10.4lf %10.4lf %10.4lf %10.4lf %10.4lf\n",
```



```
                k, u, r, x, error, error_rate);
            x = phi * x + K * (1.0 - phi)* u/a;
            error_rate = (r - x - error)/T;
            }
        fclose(outfile_ptr);
        delay(1000);
        }
}
```


APPENDIX - C

State-space Partitions and Associated Bounds on Control

The state-space is first divided into cells, and the respective upper and lower bounds on the control outputs are calculated as below.

Cell 1-1: Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_6)(\alpha_6) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(0.5)(-50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{-2.5}{1.05} = -2.81 \Rightarrow 0$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_7)(\alpha_7) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(0) + (0.1)(0)(-70)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1} = 0$$

Cell 1-2: Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_6)(\alpha_6) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(0.5)(-30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{-1.5}{1.05} = -1.429 \Rightarrow 0$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_7)(\alpha_7) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(0)(-50)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1.} = 0$$

Cell 1-3:

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_6)(\alpha_6) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(0.5)(0)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{0}{1.05} = 0$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_7)(\alpha_7) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(0)(-30)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1.} = 0$$

Cell 1-4:

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_6)(\alpha_6) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(0.5)(30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{1.5}{1.05} = 1.429$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_7)(\alpha_7) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(0)(0)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1} = 0$$

Cell 1-5:

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_9)(\alpha_9)(B_9^d)}{(w_6)(\alpha_6) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(0.5)(50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{2.5}{1.05} = 2.381$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_7)(\alpha_7) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(0)(30)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1.} = 0$$

Cell 1-6:

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_8)(\alpha_8)(B_8^d)}{(w_6)(\alpha_6) + (w_8)(\alpha_8)} = \frac{(1)(1)(0) + (0.1)(0.5)(70)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{3.5}{1.05} = 3.3$$

Rule 7: If *error* is *neg_large* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\min} = \frac{(w_7)(\alpha_7)(B_7^d) + (w_9)(\alpha_9)(B_9^d)}{(w_7)(\alpha_7) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(0)(50)}{(1)(1) + (0.1)(0)}$$

$$= \frac{0}{1.} = 0$$

Cell 2-1:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_5)(\alpha_5) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(1)(-50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-5}{1.1} = -4.55 \Rightarrow 0$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_6)(\alpha_6) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(0) + (0.1)(0.5)(-70)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{-3.5}{1.05} = -3.3 \Rightarrow 0$$

Cell 2-2:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_5)(\alpha_5) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(0) + (0.1)(1)(-30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-3.0}{1.1} = -2.73 \Rightarrow 0$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_6)(\alpha_6) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(0.5)(-50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{-2.5}{1.05} = -2.38 \Rightarrow 0$$

Cell 2-3:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_5)(\alpha_5) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(1)(0)}{(1)(1) + (0.1)((1)}$$

$$= \frac{0}{1.1} = 0$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_6)(\alpha_6) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(0.5)(-30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{-1.5}{1.05} = -1.43 \Rightarrow 0$$

Cell 2-4:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_5)(\alpha_5) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(1)(30)}{(1)(1) + (0.1)(0.1)}$$

$$= \frac{3}{1.1} = 2.73$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_6)(\alpha_6) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(0.5)(0)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{0}{1.05} = 0$$

Cell 2-5:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_9)(\alpha_9)(B_9^d)}{(w_5)(\alpha_5) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(1)(50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{5}{1.1} = 4.55$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_6)(\alpha_6) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(0.5)(30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{1.5}{1.05} = 1.43$$

Cell 2-6:

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_8)(\alpha_8)(B_{10}^d)}{(w_5)(\alpha_5) + (w_8)(\alpha_8)} = \frac{(1)(1)(0) + (0.1)(1)(70)}{(1)(1) + (0.1)(1)}$$

$$= \frac{7}{1.1} = 6.36$$

Rule 6: If *error* is *neg_medium* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\min} = \frac{(w_6)(\alpha_6)(B_6^d) + (w_9)(\alpha_9)(B_9^d)}{(w_6)(\alpha_6) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(0.5)(50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{2.5}{1.05} = 2.38$$

Cell 3-1:

Rule 4: If *error* is *zero* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_4)(\alpha_4) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(1)(-50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-5}{1.1} = -4.55 \Rightarrow 0$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_5)(\alpha_5) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(0) + (0.1)(1)(-70)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-7}{1.1} = -6.36 \Rightarrow 0$$

Cell 3-2:

Rule 4: If *error* is *zero* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_4)(\alpha_4) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(1)(-30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-3}{1.1} = -2.73 \Rightarrow 0$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_5)(\alpha_5) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(1)(-50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-5}{1.1} = -4.55 \Rightarrow 0$$

Cell 3-3:

Rule 4: If *error* is *zero* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_4)(\alpha_4) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(1)(0)}{(1)(1) + (0.1)(1)}$$

$$= \frac{0}{1.1} = 0$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_5)(\alpha_5) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(1)(-30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-3}{1.1} = -2.73 \Rightarrow 0$$

Cell 3-4:

Rule 4: If *error* is *zero* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_4)(\alpha_4) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(1)(30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{3}{1.1} = 2.73$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_5)(\alpha_5) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(1)(0)}{(1)(1) + (0.1)(1)}$$

$$= \frac{0}{1.1} = 0$$

Cell 3-5:

Rule 4: If *error* is *zero* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_9)(\alpha_9)(B_9^d)}{(w_4)(\alpha_4) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(1)(50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{5}{1.1} = 4.55$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* *pos_small*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_5)(\alpha_5) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(1)(30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{3}{1.1} = 2.73$$

Cell 3-6:

Rule 4: If *error* is *zero* then *u is zero*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_4)(\alpha_{45})(B_4^d) + (w_8)(\alpha_8)(B_8^d)}{(w_4)(\alpha_4) + (w_8)(\alpha_8)} = \frac{(1)(1)(0) + (0.1)(1)(70)}{(1)(1) + (0.1)(1)}$$

$$= \frac{7}{1.1} = 6.36$$

Rule 5: If *error* is *neg_small* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\min} = \frac{(w_5)(\alpha_5)(B_5^d) + (w_9)(\alpha_9)(B_9^d)}{(w_5)(\alpha_5) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(1)(50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{5}{1.1} = 4.55$$

Cell 4-1:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_3)(\alpha_3) + (w_{13})(\alpha_{13})} = \frac{(10)(1)(30) + (0.1)(1)(-50)}{(10)(1) + (0.1)(1)}$$

$$= \frac{295}{10.1} = 29.2$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_4)(\alpha_4) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(0) + (0.1)(1)(-70)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-7}{1.1} = -6.36 \Rightarrow 0$$

Cell 4-2:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_3)(\alpha_3) + (w_{12})(\alpha_{12})} = \frac{(10)(1)(30) + (0.1)(1)(-30)}{(10)(1) + (0.1)(1)}$$

$$= \frac{297}{10.1} = 29.4$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_4)(\alpha_4) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(0) + (0.1)(1)(-50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-5}{1.1} = -4.55 \Rightarrow 0$$

Cell 4-3:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_3)(\alpha_3) + (w_{11})(\alpha_{11})} = \frac{(10)(1)(30) + (0.1)(1)(0)}{(10)(1) + (0.1)(1)}$$

$$= \frac{300}{10.1} = 29.7$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_4)(\alpha_4) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(0) + (0.1)(1)(-30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{-3}{1.1} = -2.73 \Rightarrow 0$$

Cell 4-4:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_3)(\alpha_3) + (w_{10})(\alpha_{10})} = \frac{(10)(1)(30) + (0.1)(1)(30)}{(10)(1) + (0.1)(1)}$$

$$= \frac{303}{10.1} = 30$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_4)(\alpha_4) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(0) + (0.1)(1)(0)}{(1)(1) + (0.1)(1)}$$

$$= \frac{0}{1.1} = 0$$

Cell 4-5:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_9)(\alpha_9)(B_9^d)}{(w_3)(\alpha_3) + (w_9)(\alpha_9)} = \frac{(10)(1)(30) + (0.1)(1)(50)}{(10)(1) + (0.1)(1)}$$

$$= \frac{305}{10.1} = 30.2$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_4)(\alpha_4) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(0) + (0.1)(1)(30)}{(1)(1) + (0.1)(1)}$$

$$= \frac{3}{1.1} = 2.73$$

Cell 4-6:

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_8)(\alpha_8)(B_8^d)}{(w_3)(\alpha_3) + (w_8)(\alpha_8)} = \frac{(10)(1)(30) + (0.1)(1)(70)}{(1)(1) + (0.1)(1)}$$

$$= \frac{307}{10.1} = 30.4$$

Rule 4: If *error* is *zero* then *u is zero*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\min} = \frac{(w_4)(\alpha_4)(B_4^d) + (w_9)(\alpha_9)(B_9^d)}{(w_4)(\alpha_4) + (w_9)(\alpha_9)} = \frac{(1)(1)(0) + (0.1)(1)(50)}{(1)(1) + (0.1)(1)}$$

$$= \frac{5}{1.1} = 4.55$$

Cell 5-1:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\max} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_2)(\alpha_2) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(50) + (0.1)(0.5)(-50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{47.5}{1.05} = 45.23$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_3)(\alpha_3) + (w_{14})(\alpha_{14})} = \frac{(10)(1)(30) + (0.1)(1)(-70)}{(10)(1) + (0.1)(1)}$$

$$= \frac{293}{10.1} = 29.0$$

Cell 5-2:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_2)(\alpha_2) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(50) + (0.1)(0.5)(-30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{48.5}{1.05} = 46.2$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_3)(\alpha_3) + (w_{13})(\alpha_{13})} = \frac{(10)(1)(30) + (0.1)(1)(-50)}{(10)(1) + (0.1)(1)}$$

$$= \frac{295}{10.1} = 29.2$$

Cell 5-3:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_2)(\alpha_2) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(50) + (0.1)(0.5)(0)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{50}{1.05} = 47.6$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_3)(\alpha_3) + (w_{12})(\alpha_{12})} = \frac{(10)(1)(30) + (0.1)(1)(-30)}{(10)(1) + (0.1)(1)}$$

$$= \frac{297}{10.1} = 29.4$$

Cell 5-4:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_2)(\alpha_2) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(50) + (0.1)(0.5)(30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{51.5}{1.05} = 49.1$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_3)(\alpha_3) + (w_{11})(\alpha_{11})} = \frac{(10)(1)(30) + (0.1)(1)(0)}{(10)(1) + (0.1)(1)}$$

$$= \frac{300}{10.1} = 29.7$$

Cell 5-5:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_9)(\alpha_9)(B_9^d)}{(w_2)(\alpha_2) + (w_9)(\alpha_9)} = \frac{(1)(1)(50) + (0.1)(0.5)(50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{52.5}{1.05} = 50$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_3)(\alpha_3) + (w_{10})(\alpha_{10})} = \frac{(10)(1)(30) + (0.1)(1)(30)}{(10)(1) + (0.1)(1)}$$

$$= \frac{303}{10.1} = 30$$

Cell 5-6:

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_8)(\alpha_8)(B_8^d)}{(w_2)(\alpha_2) + (w_8)(\alpha_8)} = \frac{(1)(1)(50) + (0.1)(0.5)(70)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{53.5}{1.05} = 51$$

Rule 3: If *error* is *pos_small* then *u* is *pos_small*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\min} = \frac{(w_3)(\alpha_3)(B_3^d) + (w_9)(\alpha_9)(B_9^d)}{(w_3)(\alpha_3) + (w_9)(\alpha_9)} = \frac{(10)(1)(30) + (0.1)(1)(50)}{(10)(1) + (0.1)(1)}$$

$$= \frac{305}{10.1} = 30.2$$

Cell 6-1:

Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_8)(\alpha_8)(B_8^d)}{(w_1)(\alpha_1) + (w_8)(\alpha_8)} = \frac{(1)(1)(70) + (0.1)(0)(70)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 14: If *error* is *small* and *error_rate* is *neg_large* then *u* is *neg_large*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{14})(\alpha_{14})(B_{14}^d)}{(w_2)(\alpha_2) + (w_{14})(\alpha_{14})} = \frac{(1)(1)(50) + (0.1)(0.5)(-70)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{46.5}{1.05} = 44.5$$

Cell 6-2:

Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_1)(\alpha_1) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(70) + (0.1)(0)(-30)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 13: If *error* is *small* and *error_rate* is *neg_medium* then *u* is *neg_medium*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{13})(\alpha_{13})(B_{13}^d)}{(w_2)(\alpha_2) + (w_{13})(\alpha_{13})} = \frac{(1)(1)(50) + (0.1)(0.5)(-50)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{47.5}{1.05} = 45.2$$

Cell 6-3:

Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_1)(\alpha_1) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(70) + (0.1)(0)(0)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 12: If *error* is *small* and *error_rate* is *neg_small* then *u* is *neg_small*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{12})(\alpha_{12})(B_{12}^d)}{(w_2)(\alpha_2) + (w_{12})(\alpha_{12})} = \frac{(1)(1)(50) + (0.1)(0.5)(-30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{48.5}{1.05} = 46.2$$

Cell 6-4:

Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_1)(\alpha_1) + (w_{10})(\alpha_{10})} = \frac{(1)(1)(70) + (0.1)(0)(30)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 11: If *error* is *small* and *error_rate* is *zero* then *u is zero*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{11})(\alpha_{11})(B_{11}^d)}{(w_2)(\alpha_2) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(50) + (0.1)(0.5)(0)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{50}{1.05} = 47.6$$

Cell 6-5: Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_9)(\alpha_9)(B_9^d)}{(w_1)(\alpha_1) + (w_9)(\alpha_9)} = \frac{(1)(1)(70) + (0.1)(0)(50)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 10: If *error* is *small* and *error_rate* is *pos_small* then *u* is *pos_small*

$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_{10})(\alpha_{10})(B_{10}^d)}{(w_2)(\alpha_2) + (w_{11})(\alpha_{11})} = \frac{(1)(1)(50) + (0.1)(0.5)(30)}{(1)(1) + (0.1)(0.5)}$$

$$= \frac{51.5}{1.05} = 49.1$$

Cell 6-6: Rule1: If *error* is *pos_large* then *u* is *pos_large*

Rule 8: If *error* is *small* and *error_rate* is *pos_large* then *u* is *pos_large*

$$u_{\max} = \frac{(w_1)(\alpha_1)(B_1^d) + (w_8)(\alpha_8)(B_8^d)}{(w_1)(\alpha_1) + (w_8)(\alpha_8)} = \frac{(1)(1)(70) + (0.1)(0)(70)}{(1)(1) + (0.1)(0)}$$

$$= \frac{70}{1} = 70$$

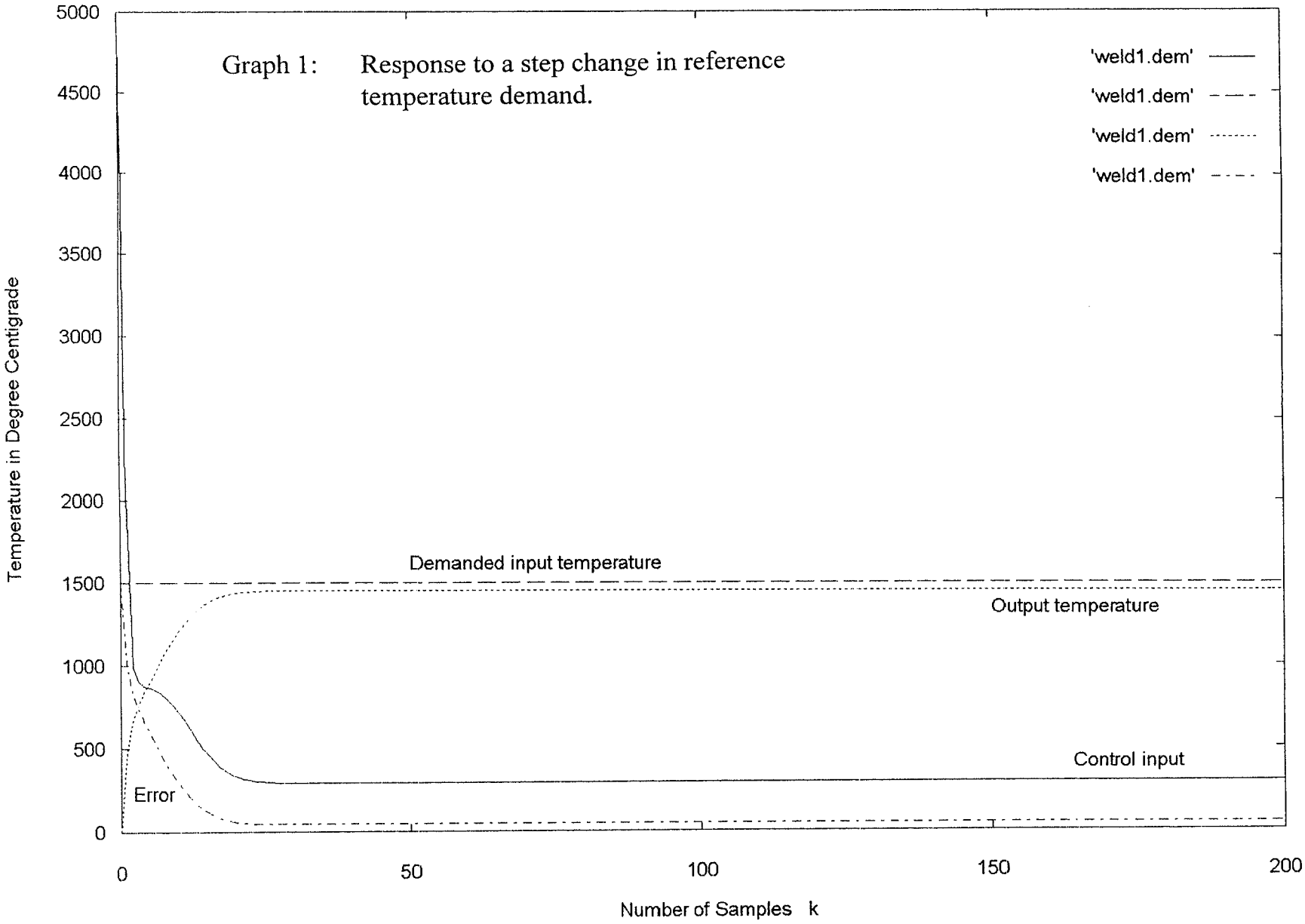
Rule 2: If *error* is *pos_medium* then *u* is *pos_medium*

Rule 9: If *error* is *small* and *error_rate* is *pos_medium* then *u* is *pos_medium*

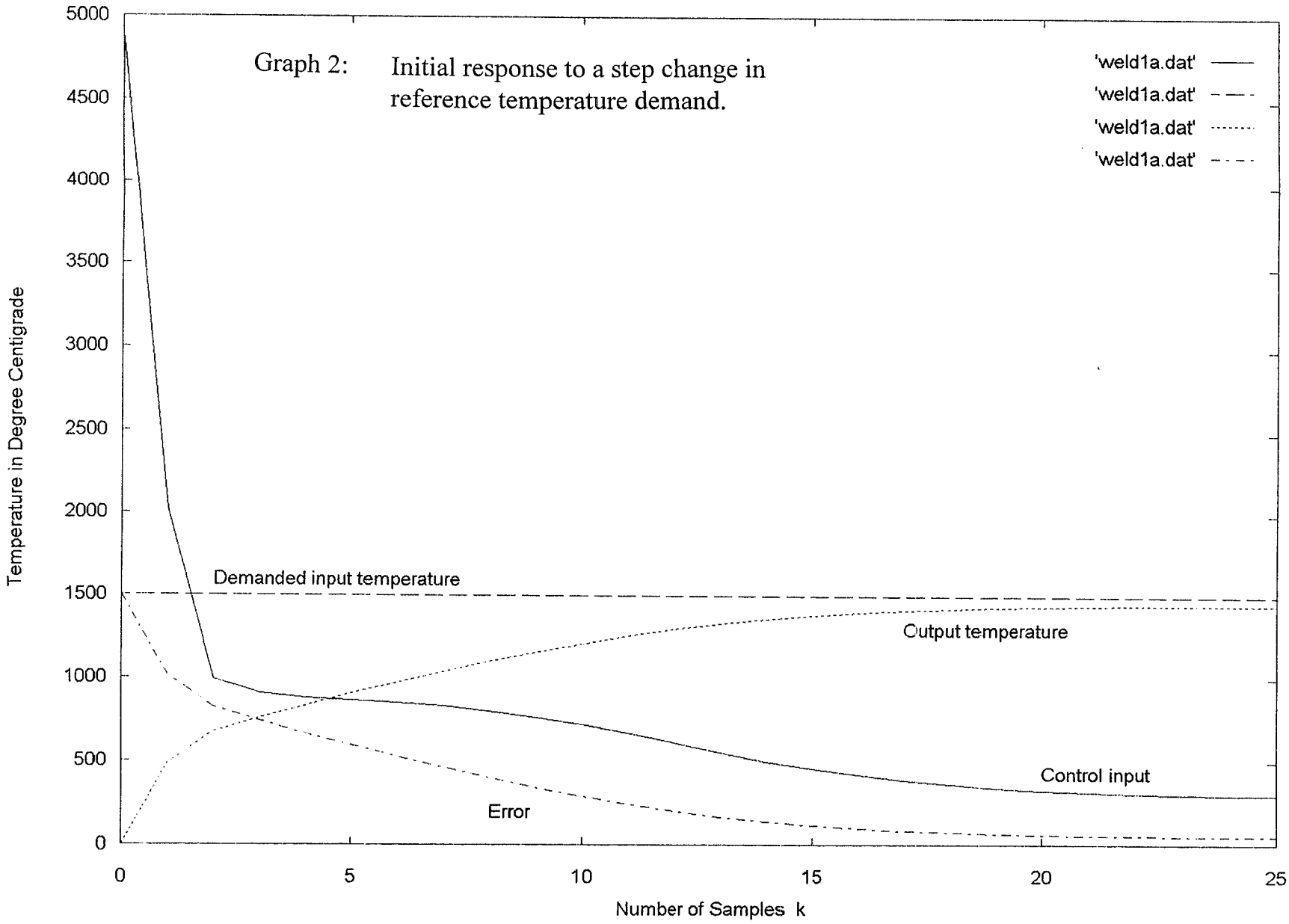
$$u_{\min} = \frac{(w_2)(\alpha_2)(B_2^d) + (w_9)(\alpha_9)(B_9^d)}{(w_2)(\alpha_2) + (w_9)(\alpha_9)} = \frac{(1)(1)(50) + (0.1)(0.5)(50)}{(1)(1) + (0.1)(0.5)}$$

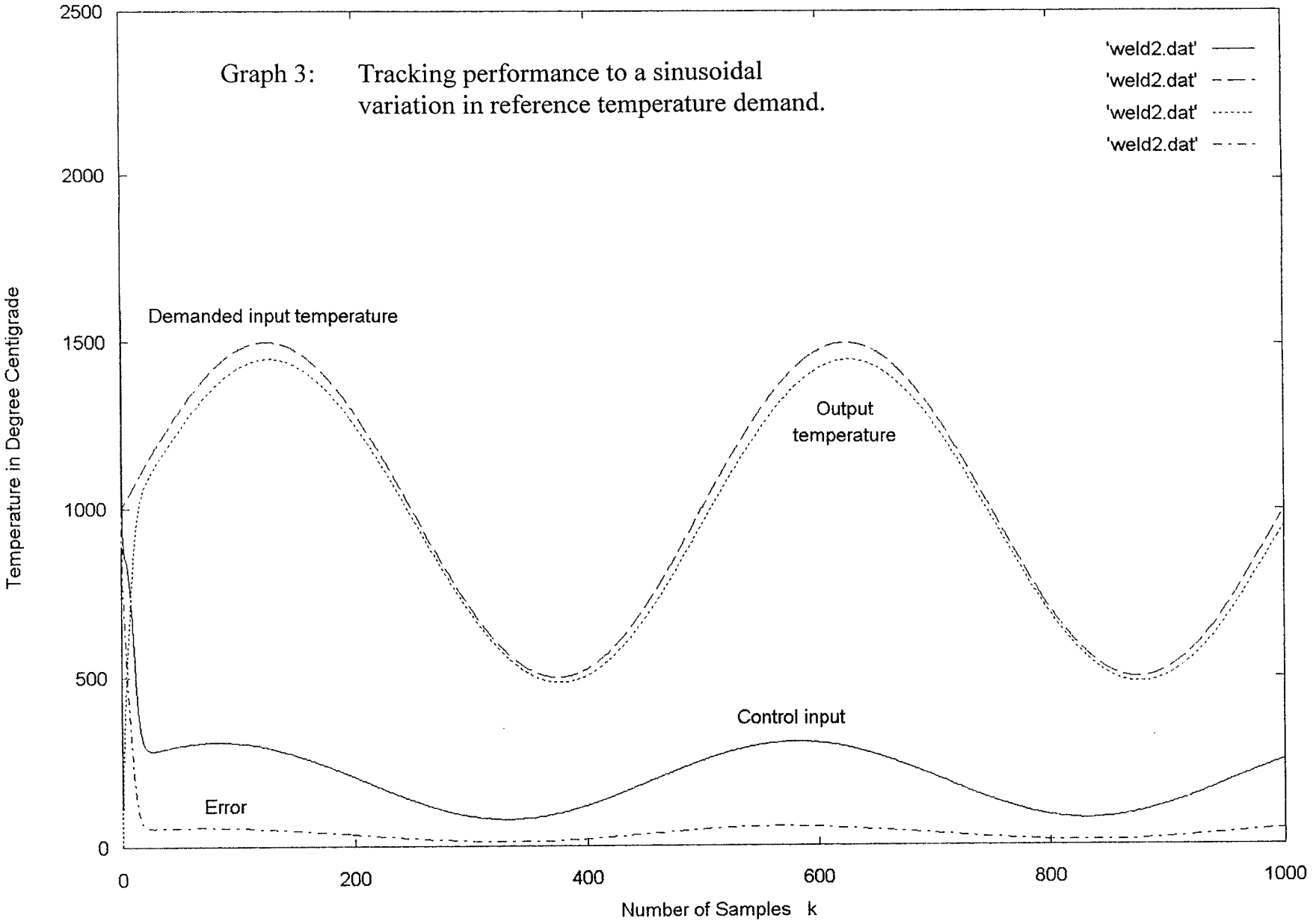
$$= \frac{52.5}{1.05} = 50$$

Graph 1: Response to a step change in reference temperature demand.

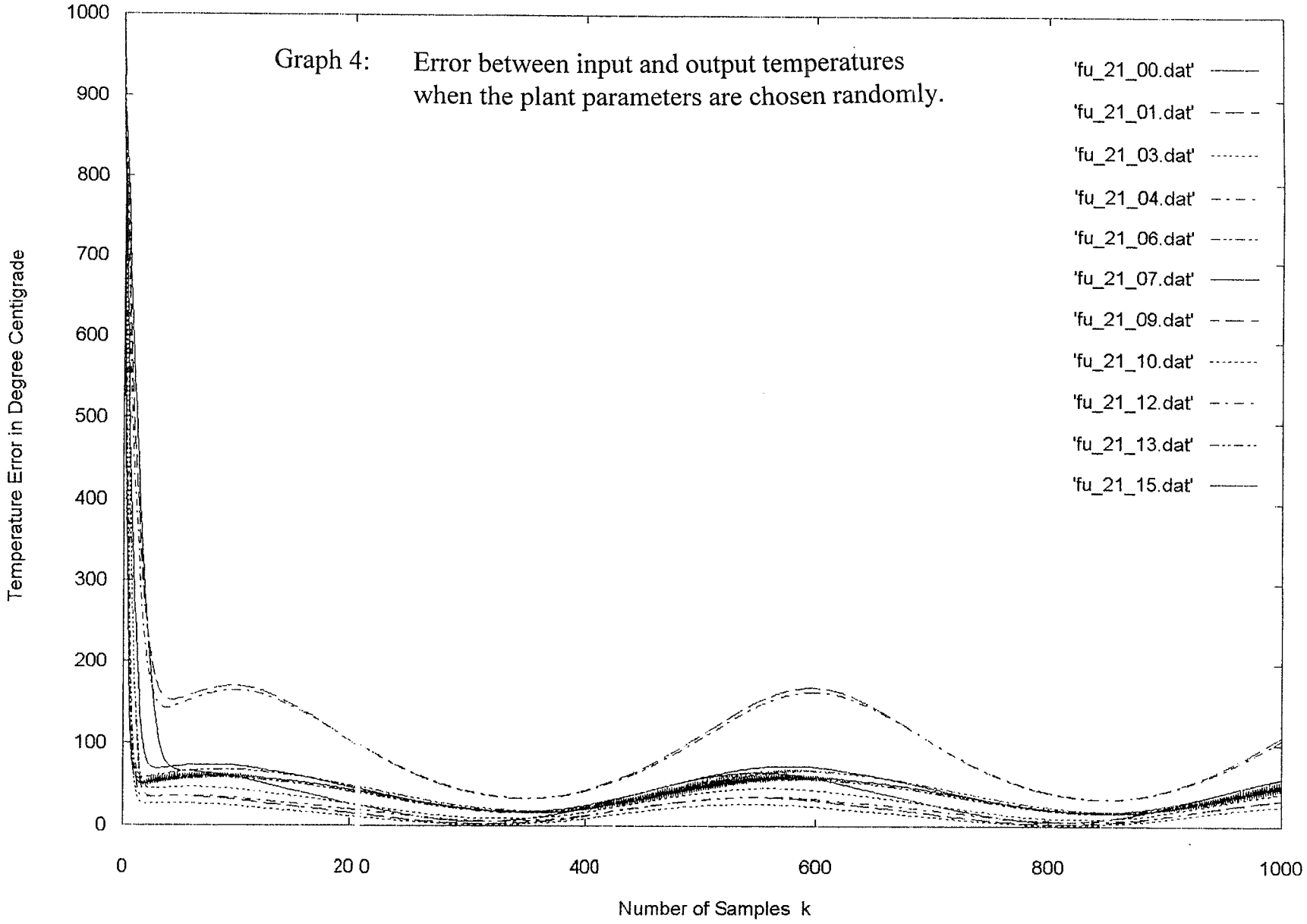


Graph 2: Initial response to a step change in reference temperature demand.





Graph 4: Error between input and output temperatures when the plant parameters are chosen randomly.



Graph 4: Error between input and output temperatures when the plant parameters are chosen randomly.