173

FINAL TECHNICAL REPORT

ON

Cooperative Agreement NCC 3-109

TEMPERATURE AND MELT SOLID INTERFACE CONTROL DURING CRYSTAL GROWTH

Starting Date : May, 1 1989
Ending Date : November, 14 1989

Prepared by

Dr. Celal Batur
Department of Mechanical Engineering
The University of Akron
Akron-Ohio 44325-3903

July 1990

1. Abstract

This technical report summarizes findings on the adaptive control of a transparent Bridgman crystal growth furnace. The task of the process controller is to establish a user specified axial temperature profile by controlling the temperatures in eight heating zones. The furnace controller is built around a computer. Adaptive PID (Proportional Integral Derivative) and Pole Placement control algorithms are applied. The need for adaptive controller stems from the fact that the zone dynamics changes with respect to time. The controller tested extensively on the Lead Bromide crystal growth. Several different temperature profiles and ampoule's translational rates are tried.

The second objective of this study is to determine the feasibility of solid liquid interface quantification by image processing. The interface is observed by a color video camera and the image data file is processed to determine if the interface is flat, convex or concave.

2. Objectives and expected results

The objective of this experimental study is twofold :

A. Design, build, and validate a computer based temperature controller for a transparent Bridgman crystal growth furnace.

B. Quantify the solid liquid interface of the crystal by image processing techniques.

The transparent furnace consists of eight electrically heated zones which are not insulated from each other. Due to the lack of insulation, heat exchange takes place between different zones. The process controller is designed in such a way that the zone temperature is kept at the desired level, despite the heat exchange effects of the neighboring zones. This calls for an adaptive controller design. The

adaptability is implemented in a sense that the controller tunes itself to changing dynamics of the heating zone. The change in dynamics is also caused by the ampoule which moves vertically within the furnace. The controller is formed by a current dynamic model of the heating zone. This model takes into account the effects of heat exchange between the heating zones. This undesirable heat exchange can be considered as the process disturbance. The task of the adaptive controller is then to keep the zone temperature at the reference level despite the effects of changes in the zone dynamics and the disturbances.

The second objective of this study is to quantify the shape of the solid liquid interface while the crystal is growing. Quantification is understood in a sense that the interface position is determined with respect to some fixed reference coordinate frame. This information can be used for the following purpose.

a - The effect of process parameters on the crystal growth can be studied. In a typical furnace, these parameters are the axial furnace temperature profile and the translational motion of the ampoule.

b - The interface profile can be used as a feedback control signal. The desired interface shape can be compared with the actual interface profile and the resulting error signal is employed as input to process controller.

c - The position of the interface, determined through image processing, permits us to determine the crystal growth rate.

Quantification of the interface is performed by the color image processing techniques.

3.Introduction

One common way to grow crystal is to subject the molten material to a temperature gradient. This requires keeping a

predetermined axial and radial temperature profile around the material despite the effects of unknown process disturbances. To maintain continuous solidification, either the material or the temperature gradient is moved axially. **Figure 1** shows typical temperature profiles surrounding the material. At any given point, the temperature is a function of axial position h, radial position r, circumferential position c, and the time t. This distributed nature of the temperature can be described by a multivariable function $T(h,r,c,t)$. The time dependency of the temperature is due to the following factors:

- the translational motion of the ampoule with respect to furnace,
- the inevitable heat exchange between different heating zones, and
- the duty cycle modulation of AC the heater drive systems, particularly at low power settings.

The temperature control problem can now be described as follows:
Determine the electrical heating power to individual heating zones and the translational motion velocity of the ampoule such that at each point within the ampoule the temperature is kept at the desired level i.e.

$$T(h,r,c,t) = T_d(h,r,c,t)$$

Previous research has shown that the shape of the solid liquid interface is strongly correlated to the imposed temperature gradient and the translational motion velocity of the ampoule. Furthermore, the macroscopic shape of the interface affects the crystalline perfection of the material. It is known that [Chang and Wilcox 1974], minimization of thermal stresses can be achieved by maintaining a planar interface between the solid and the liquid phases of the material. Such planar interface is desirable because crystalline imperfections are also

minimized. [Taghavi and Duval 1989] have studied the effects of furnace temperature profile and translational velocity on the interface shape, and have also considered the inverse problem of the desired temperature profile that would result in a planar interface. These studies have given insight into the problem of achieving planar interface during crystal growth . Most importantly, the ability to design a furnace arrangement where the temperature profile can be tailored during the growth of the crystal is found to be highly desirable and advantageous for obtaining a planar interface. This has led into the concept of transparent multi-zone Bridgman furnace.

Experimental observations confirm that the same factors affect the shape of interface, namely :

1 the axial temperature profile alongside the ampoule

2.the radial temperature profile surrounding the interface, and

3.the translational velocity of the ampoule.

[Wang,With and Corrutters al 1984] experimented with a single zone vertical Bridgman type furnace using Gallium doped Germanium as the material. They considered the energy flow, the temperature distribution and the translational speed as their main parameters. They verified that the macroscopic growth rate was always transient and the interface remained concave with respect to solid at the translational speed of 4 micro meter/sec. They further concluded that the main parameters effecting the crystal growth process are :

i) the establishment of a well defined and symmetrical temperature profile around the ampoule, and

ii) the provision for the growth interface quantification and control

The Mellen EDG furnace [Parsey and Thiel 1985] can emulate the heat flow characteristics of the horizontal Bridgman-Stockbarger furnace without requiring the motion of the ampoule. The use of the Mellen furnace in horizontal

configuration for a large diameter Gallium Arsenide crystal has been investigated by [Bourret et al 1987]. They have found that the crystalline perfection is improved by increasing the axial temperature gradient over the solid and decreasing the radial temperature gradient over the melt.

Quantification of solid-liquid interface by image processing has been performed by [Batur et al 1990a, 1990b]. This powerful technique can locate the position of interface and determines its concavity or convexity for transparent furnace and materials. This topological information about the interface can serve the following purpose :

- It provides continuous on line information for the position and the shape of the interface. Therefore the effects of different temperature gradients and the translational speed on the interface can be quantified in real time.
- It can be used as the feedback signal in conjunction with the classical temperature feedback from each zone.

In conclusion, material structural properties, partially observed by the shape of solid-liquid interface, are complex, time dependent phenomena of the temperature profiles and the ampoule's translational velocity. An experimental furnace model explaining the interactions between different heating zones and the effect of the translational velocity is the first fundamental step to understand the process. The purpose of the proposed investigation is to find this model in real time and based on the model, implement an adaptive control algorithm for temperatures and interface profile.

4. Crystal growth control

Control of crystal growth should ideally be based on the desired structural properties of a given material. For example, if the acousto-optical properties of the material

are critical then the control law should be designed around these parameters. On line measurement of the solid's refractive index, for example, provides such a feedback signal for the control algorithm. However if the desired structural properties can not be directly measured because of cost or furnace design, then two alternative options exist. If the relation between the desired state and the physically realizable measurement is known, then this relation can be used to find the desired measurement. If this relation is unknown, then the next best measurement, closest to the desired observation should be provided for the controller.

The shape of the interface is one measurement which is closely related to the structural properties of the material. X-ray imaging of the ampoule, for example, may provide information to determine the solid-liquid interface [Fripp et al 1988]. The differences in X-ray intensities corresponding to solid and melt can be exploited to determine the topology of the interface. X-ray imaging is an elaborate and equipment intensive process. If there is a significant difference in gray levels or colors between the solid and the melt, then optical imaging can also locate the position and the shape of the interface, [Batur et al 1990a,1990b]

Perhaps the most practical furnace control methodology is to measure and control the temperatures which define the necessary thermal gradient for the growth of the crystal. Although this control technique is not based on the measurement of the desired structural properties of the material, nevertheless it is preferred for the following reasons :

- As outlined in the introduction, analytical studies have shown that the shape of the interface can be related to the axial and radial temperature surrounding the interface.

- Temperature measurement technology is well established in comparison with for example optical or on line material stress measurements.

In the next section we will consider the temperature control problem.

4.1 Single input single output adaptive temperature control

This mode of control starts with single input single output model of the heating zone. If the heat energy input and the zone temperatures are denoted by u and y respectively then, for each zone, the assumed model is in the following form:

$$y(t)= a_1\ y(t-1)+a_2\ y(t-2)+a_n\ y(t-n)+ b_1\ u(t-1)+b_2\ u(t-2)+b_n\ u(t-n)+v(t)$$

$$v(t)= c_1\ v(t-1)+c_2\ v(t-2)+c_n\ v(t-n)+e(t)$$

where the term v(t) accounts for the heat exchange interactions between the neighboring zones and e(t) is white noise. For adaptive control of temperatures, estimates of process parameters (a_i,b_i,and c_i) are obtained by the Least Squares or the Generalized Least Squares Technique. At each sampling time, these estimated values are substituted into the control law as if they are the true values for the unknown parameters (a_i,b_i,and c_i).

[Batur et al 1990a 1990b] have applied single input single output adaptive temperature control to an eight zone transparent furnace. Process parameters (a_i,b_i) are obtained by the sequential Least Squares Technique and two different control laws namely adaptive PID and adaptive pole placement controllers have been applied. Detailed derivations of these adaptive control laws are given in the **Appendix 1.**

4.2 Multi input multi output adaptive temperature control

Figure 2a shows structure of the multivariable controller. Process outputs are the measured zone temperatures. Process inputs are the heating powers and the translational velocity of the ampoule with respect to the furnace. Due to the multizone configuration , there are inevitable interactions between the neighboring heating zones. The input output dynamics of the system can be represented by the following first order multivariable model.

$$\begin{aligned} y_1(t) = {} & a_{11}\, y_1(t-1)+a_{12}\, y_2(t-1)+\ldots+a_{1,8}\, y_8(t-1)+ \\ & b_{11}\, u_1(t-1)+b_{12}\, u_2(t-1)+\ldots+b_{1,9}\, u_9(t-1)+ \\ & c_{11}\, e_1(t-1)+c_{12}\, e_2(t-1)+\ldots+c_{1,8}\, e_8(t-1) \end{aligned}$$

$$\begin{aligned} y_2(t) = {} & a_{21}\, y_1(t-1)+a_{22}\, y_2(t-1)+\ldots+a_{2,8}\, y_8(t-1)+ \\ & b_{21}\, u_1(t-1)+b_{22}\, u_2(t-1)+\ldots+b_{2,9}\, u_9(t-1)+ \\ & c_{21}\, e_1(t-1)+c_{22}\, e_2(t-1)+\ldots+c_{2,8}\, e_8(t-1) \end{aligned}$$

$$\begin{aligned} y_8(t) = {} & a_{8,1}\, y_1(t-1)+a_{8,2}\, y_2(t-1)+ \quad +a_{8,8}\, y_8(t-1)+ \\ & b_{8,1}\, u_1(t-1)+b_{8,2}\, u_2(t-1)+\ldots+b_{9,9}\, u_9(t-1)+ \\ & c_{8,1}\, e_1(t-1)+c_{8,2}\, e_2(t-1)+ \quad +c_{8,8}\, e_8(t-1) \end{aligned}$$

These equations can be expressed in a much more compact form by the use of vector matrix representation. Combining all input, output and disturbance terms into appropriate vectors, the multi input, multi output representation takes the following form.

$$y(t) = A(z^{-1})y(t)+B(z^{-1})u(t-1)+[I+C(z^{-1})]e(t) \qquad (1)$$

where the z^{-1} is the delay operator defined as, for example,

$y(t-1)= z^{-1}\, y(t)$.

Input output and disturbance vectors are explicitly given as

$$u(t) = [u_1, u_2, \ldots u_9]^T$$
$$y(t) = [y_1, y_2, \ldots y_8]^T$$
$$e(t) = [e_1, e_2, \ldots e_8]^T$$

where T is the transpose operator and u_9 corresponds to the translational speed. The A, B and C of (1) are appropriate matrices representing the dynamics of the system and the noise. For example, the A matrix is

$$A(z^{-1}) = \begin{matrix} a_{11}\, z^{-1} & \ldots & a_{1,8}\, z^{-1} \\ a_{21}\, z^{-1} & \ldots & a_{2,8}\, z^{-1} \\ a_{8,1}\, z^{-1} & \ldots & a_{8,8}\, z^{-1} \end{matrix}$$

The rationale behind this model are as follows:

a- Open loop step response of each individual zone essentially follows an exponential curve suggesting first order dynamics.

b- Interaction between the heating zones can be taken into account with the off diagonal terms of the A,B,C matrices. Note that this multivariable model is drastically different from that of the single input single output. Zone interactions are explicitly modelled in the multivariable case as the off diagonal terms of A,B, and C matrices. The single input single output model on the other hand tries to accommodate this interactions by the added noise term to model. If there is negligible interaction, then A,B and C matrices become diagonal as , for example

$$A(z^{-1}) = \{a_{ii} z^{-1}\}_{8*8} \qquad i=1,2..8.$$

If the disturbances are not correlated i.e. C={0} then the identification of model parameters A and B can be performed by the sequential Least Squares parameter estimation

techniques. In the case of correlated disturbances, C={0}, The Maximum Likelihood or Extended Least Squares identification procedures must be used [Ljung and Soderstrom 1983] , [Goodwin and Sin 1984].

Once the feedback control system performance has been specified, the control action for each zone can be determined. In crystal growth furnaces, due to the slow motion of the ampoule with respect to the furnace, the dynamics of the furnace changes in time. Therefore A,B, and C matrices describing the dynamics are not actually time invariant. However these changes are not expected to be fast. This is because the relative speed of the ampoule is slow (on the average 2 cm/day) . Therefore a self tuning multivariable temperature control algorithm can be applied. Process parameters A, B, and C of (1) can be estimated by an on-line Generalized Least Squares algorithm and the following performance indices can be minimized by the controller.

4.2.1 Multivariable minimum variance controller

The minimum variance controller minimizes

$$V= (1/N) \{ W_1(T_d(k)-T(k-1-m))+W_2(u(t))\} \quad (2)$$

where V is a scalar function of input u(t), output T(k), and the desired process output $T_d(k)$, and (k) is the time index. The functions W1(.) and W2(.) define the emphasis that one puts on the output error [$T_d(k)-T(k-1)$] and the control signal u(t). Common choices are:

1. $W_2(u(t)) = u_1^2(t)+\ldots+u_9^2(t)$

2. $W_2(u(t)) = [u_1(t)-u_1(t-1)]^2+ \ldots+[u_9(t)-u_9(t-1)]^2$

3. $W_1(T_d(k)-T(k-1))=[T_{d1}(t)-T_1(t-1)]^2+\ldots+[T_{d8}(t)-T_8(t-1)]^2+$

where the choice (1) considers the heating power and penalizes excessive power sent to heaters whereas (2) performs the same action for changes in heating power. Inclusion of $W_2(.)$ into the performance index stems from the fact that for a typical crystal growth furnace, the radiative energy received by the ampoule may be a considerable portion of the total energy (convective + radiation). For example, in one experiment involving an eight zone furnace, it has been observed that the heating wire surface temperature can approach 800 C. At the emissivity of 0.8 , approximately 40% of the radiation energy will transmit through the quartz ampoule and reach the material.
$W_1(.)$ shown in the third choice makes sure that controller tracks the desired temperatures in a mean square sense.

4.2.2 Pole placement control algorithm

Ignoring disturbances, the process dynamics of Eqn.1 can be expressed as

$$y(t) = P(z^{-1})u(t)$$

If the controller dynamics is written as

$$u(t) = C(z^{-1})\ [T_d(t)-T(t)]$$

then the closed loop system, as far as the error is concerned, will be given as

$$[T_d(t)-T(t)] = \{I- [I+P(z^{-1})C(z^{-1})]^{-1}\ P(z^{-1})]\ C(z^{-1})]\}\ r(t)$$

where I is an 8*8 unity matrix. In pole placement design the controller matrix $C(z^{-1})_{9*8}$ is chosen such that the error recovery follows a user specified pattern.

The reason behind these different choices is that each design emphasizes on certain aspects of the control. The minimum variance controller provides the prediction for the "predictable" part of the disturbances and tries to eliminate the disturbances before their effects actually take place. The pole placement design however, shapes the dynamics of error recovery. Here the disturbances are not explicitly predicted.

In conclusion, the self-tuning control algorithm,

- experimentally identifies A,B, and C matrices of the process dynamics in real time, and
- generates a control action for each zone in such a way that the performance index in Eqn. (2) is minimized.

In this report, only the experimental results associated with the single input single output adaptive control will be presented. **Figure 2b** shows the structure of the single input single output adaptive controller and its associated hardware. Multi input output experimental work is currently being performed.

5. Electro dynamic gradient control (EDG)

Electro dynamic gradient controller translates the temperature gradient, instead of the ampoule. In principle, this can be achieved by changing the desired temperature in each zone so that the effect of translating the ampoule can be duplicated by the changes in the desired temperature profile. There are two main requirements for the success of this procedure.

1. When the desired zone temperatures are changed, the actual zone temperatures should settle on the new levels as soon as possible. Furthermore, there should be no

temperature overshoot during this transition. Due to continuous changes in the desired temperatures, this is no longer a temperature regulator problem. In controls terminology, this is known as a servo problem where the desired levels change with respect to time.

2. No matter how sophisticated the controller is, the fine vertical motion of the critical temperature of the imposed temperature gradient is effectively bounded by the number of heating zones surrounding the ampoule. Increasing the number of zones will permit to move this point in very fine increments, therefore minimizing the effects of step temperature changes within the ampoule.

The proposed adaptive control schemes will also work under this servo control condition. Due to slow translational changes in the imposed temperature profile , process dynamics, therefore A,B and C matrices are not expected to change fast. The trade-off between the fast settling and the low overshoot can be adjusted by simply changing weights in the performance indices.

6. Interface Quantification Problem

There are essentially four reasons to quantify the interface between the solid and the melt:

1. Given the shape of the interface one can determine the concavity or convexity of the solid with respect to the liquid.

2. It is possible to determine the relation between the interface growth velocity and the ampoule's translational speed.

3. The effects of the imposed temperature profile on the shape of the interface can be determined.

4. Most importantly, the shape of the interface can be used as a feedback signal by manipulating on the translational speed and the axial temperature profile.

The interface quantification through imaging has been attempted before. For example [Bachman and Kirsh 1970] and have used video signals to locate the melt-solid interface. [Fripp et al 1988] have devised an an X-ray imaging technique . This procedure works on the difference in densities between the transmitted images of the solid and liquid phases.

Figure 3 shows the hardware features of the image processing system. The color video camera looks at the solid liquid interface within the ampoule. The location of the interface within the image frame does not significantly change in time. In fact , from the camera point of view the relative speed v of the interface is determined by

v= speed of ampoule - crystal growth speed

For slow translation rate, the crystal growth rate matches the speed of the ampoule and therefore (v) is small and the interface practically remains within the viewing area of the camera.

The image is digitized with a resolution of 8 bits/color and on a three color (RGB) basis. The task of the image processing algorithm is to

i) locate the interface,

ii) quantify the degree of concavity or convexity ,and

iii)determine the relative speed of the interface ,

in almost real time so that the interface growth can further be manipulated by changing the translational speed of the ampoule.

The procedure to accomplish the interface location can be summarized as follows :

1. obtain the histogram of the area of interest within the image,
2. perform histogram based segmentation,
3. perform dilation or median based filtering to eliminate the noise in the segmented image and, finally

4. apply edge detection procedures to find the interface between the solid and the melt .[Rosenfeld and Kak 1982] , [Jain 1989] , [Ekstrom 1984].

7.Experimental results

The schematic of the transparent furnace is shown in **Figure 4.** Each one of the four zones located in the middle have a nominal resistance of approximately 8 ohms. Outer zones have 5 ohms resistance. Temperatures in each zone are measured by two thermocouples, one for the temperature control and the other for an independent On-Off controller for safety. Extra thermocouples are also placed radially and axially around the interface to monitor the temperature distribution, however these are not used by the temperature controller. Temperatures are measured by programmable transducers. Currently a 0-700 C range corresponds to a 0-10 V span. However the range can be made narrower for increased resolution without affecting the span. On the computer side , the data acquisition board has 12 bits providing temperature reading resolution of 0.17 C. Heating actuators are zero crossing , duty-cycle modulated solid state switches . Each can deliver 30 A. at 110 V. Details of the computer side of the instrumentation can also be found in **Figure 4.**

Figure 5 shows the temperature profiles when step changes are introduced in the reference temperatures. In this case, the adaptive controller is single input single output pole placement controller. We have found that it is intuitively more appealing to use the pole placement controller. This is because, the desired dynamic response of the controller can be easily fixed by placing closed system poles at desired locations. In this example the desired closed system poles are located at

$z_1 = 0.9$, $z_2 = 0.0$. **Table 1** shows the estimates of the parameters representing the zone dynamics.

The overall performance of the controller is close to the design specifications. For steady state operations (all reference temperatures are constant), each zone temperature has controller error with zero mean and 0.1 C variance. For the dynamic response, the temperature control system follows the changes in the desired temperatures according to the performance index.

Figure 6 shows the main result of the interface quantification procedure. The interface growth rate, which is a critical process parameter, is automatically determined by the image processing technique. The flow chart for finding this interface is shown in **Figure 7.** Currently this procedure takes about 5 seconds on an IBM PC AT based computer. This time is slightly longer than controller sampling time which is 2 seconds. For the existing hardware, one practical solution to this problem is to perform the interface quantification less frequently than 5 seconds. This approach can be justifiable due to the fact that the dynamics of the interface is slower than that of the furnace.

The listing of the temperature control and the image quantification program is given in the **Appendix 2.**

8. Conclusion

A Bridgman type transparent crystal growth furnace is controlled by two different adaptive controllers. Changes in the furnace dynamics and the interaction between the neighboring heating zones are estimated by an on line Least Squares algorithm. The pole placement and the PID control algorithms are extensively tested on the growth of Lead Bromide crystals. Both controllers performed according to their expected levels of performance. However it is found that the pole placement controller is much more flexible to

implement any desired closed loop feedback dynamics. On the other hand, the adaptive PID controller only executes the Ziegler-Nichols based performance index.

The second objective of this study is to determine the position and the shape of the solid liquid interface by image processing. The interface is observed by a color video camera and the image data file is processed to determine if the interface is flat, convex or concave.

10.References

1. Singh N.B., Duval W.M.B., Rosenthal B.W " Characterization of directionally solidified lead chloride", Journal of Crystal Growth. Vol. 89, 1988, pp 80-85.

2. Taghavi K, Duval W.M.B "Inverse heat transfer analysis of Bridgman Crystal Growth" International Journal of Heat and Mass Transfer. Vol. 32 No.9 pp. 1741-1750

3. Chang E.C.,Wicox R.W." Control of interface in the vertical Bridgman-Stockbarger Technique" J. of Crystal Growth Vol 21, pp135-140 ,1974

4. Wang,C.A.,Witt,A.F.,Carruthers,J.R "Analysis of crystal growth characteristics in a conventional vertical Bridgman configuration" J. of Crystal Growth 66 1984 299-308

5. Parsey,J.M.,Thiel,F.A "A new apparatus for the controlled growth of single crystals by horizontal Bridgman techniques" J. of Crystal Growth 73. pp. 211-220 1985.

6. Stockbarger,D.C "The production of large single crystals of Lithium Floride" P.S.I Volume 7. pp 133-136 Vol.7 March 1936.

7. Bourret,E.D.,Guitron J.B,Haller E.E "Evaluation of the Mellen EDG furnace for growth of large diameter GaAs single crystals in a horizontal configuration" J. of Crystal Growth 1985 (1987) pp 124-129

8. Astrom K.J ,Hagglund T "Automatic tuning of PID controllers" ISA publications 1988.

9. Bachman K.J.,Kirsch H.T Journal of crystal growth 7,290.

10. Archibald L.F.,Debnam J.W., Berry F.R."Imaging of directional solidification interfaces" NASA Technical Briefs. February 1988

11. Goodwin G.C,Sin K.S"Adaptive filtering prediction and control" John Wiley 1984.

12. Ljung L.,Soderstrom T."Theory and practice of recursive identification" 1983 Cambridge Mass: MIT Press.

13. Jain A.K "Fundamental of digital image processing" Prentice Hall 1989.

14. Rosenfeld A.,Kak A.C. "Digital picture processing" Vol 1&2 Academic Press 1982

15. Ekstrom M.P "Digital Image Processing Techniques" Academic Press 1984

16. Ohta Y.,Kanade T.,Sakai T "Color information for region segmentation" Computer graphics and image processing 13, pp 222-241 1980

17. Batur C., Sharpless R., Duval W.M.B., Rosenthal B., Singh N.B. " Solid-liquid interface profile control for transparent multizone bridgman type crystal growth furnaces" ISA 90 International conference. October 14-19 1990 (a). New Orleans, Louisiana. To be published in the proceedings.

18. Batur C., Sharpless R., Duval W.M.B., Rosenthal B., Singh N.B. " Crystal production control by image processing based feedback control system" To be published in the Proceedings of the 1990 (b) ASME Conference.

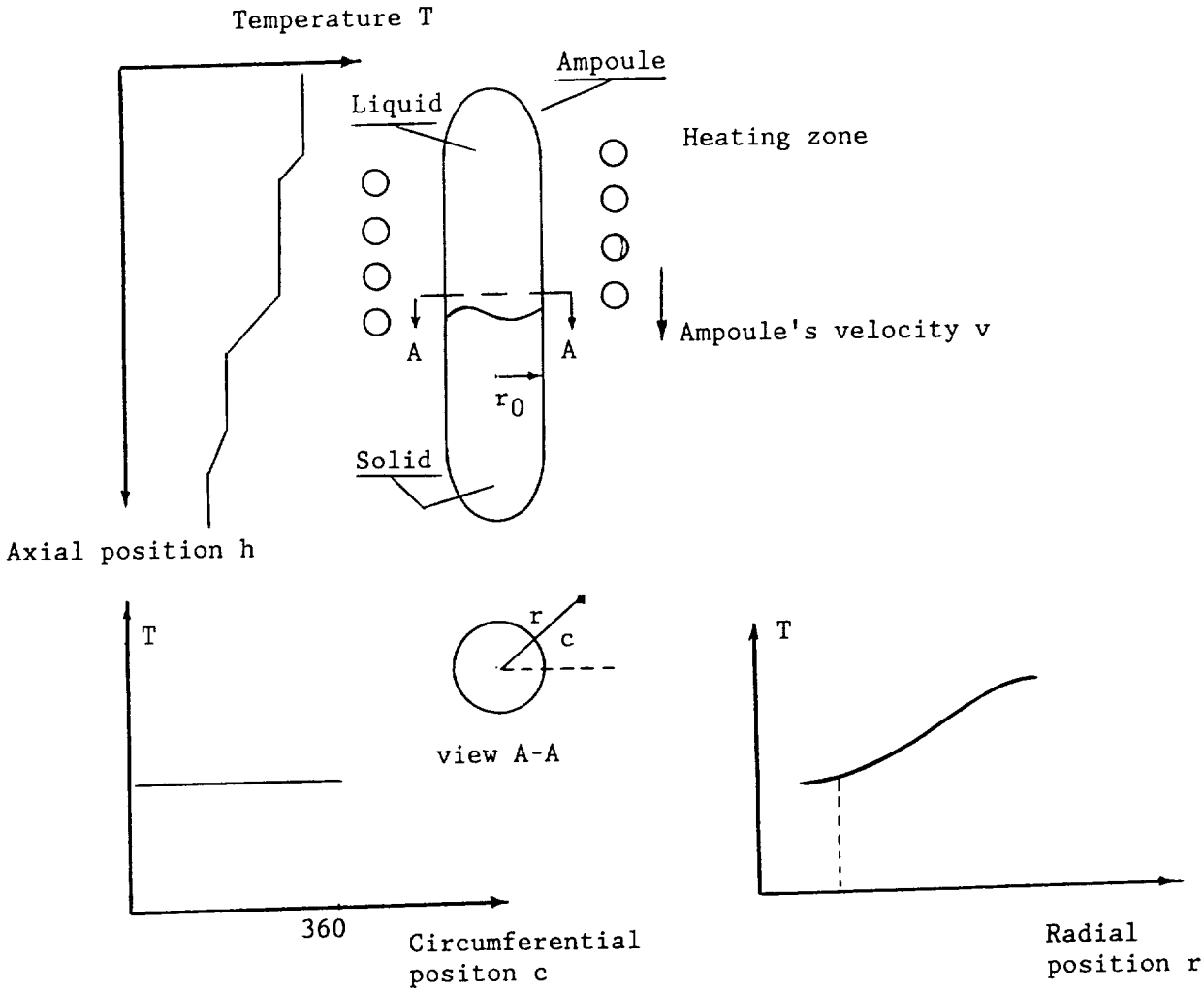


Figure 1. Distributed nature of the temperature.
At any point, the temperature is a function of h,r,c and the time t.

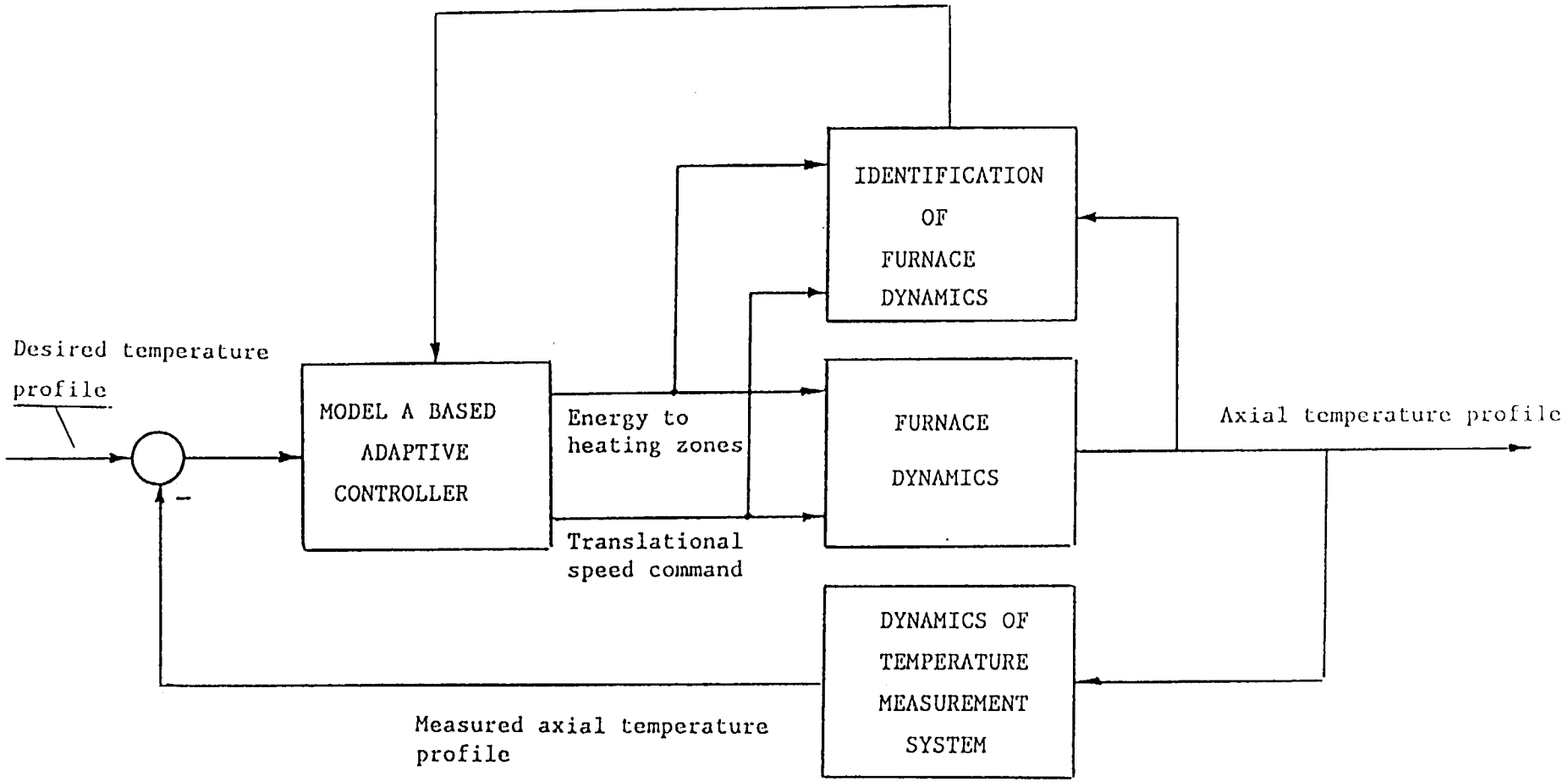


Figure 2a. Adaptive multivariable temperature control system

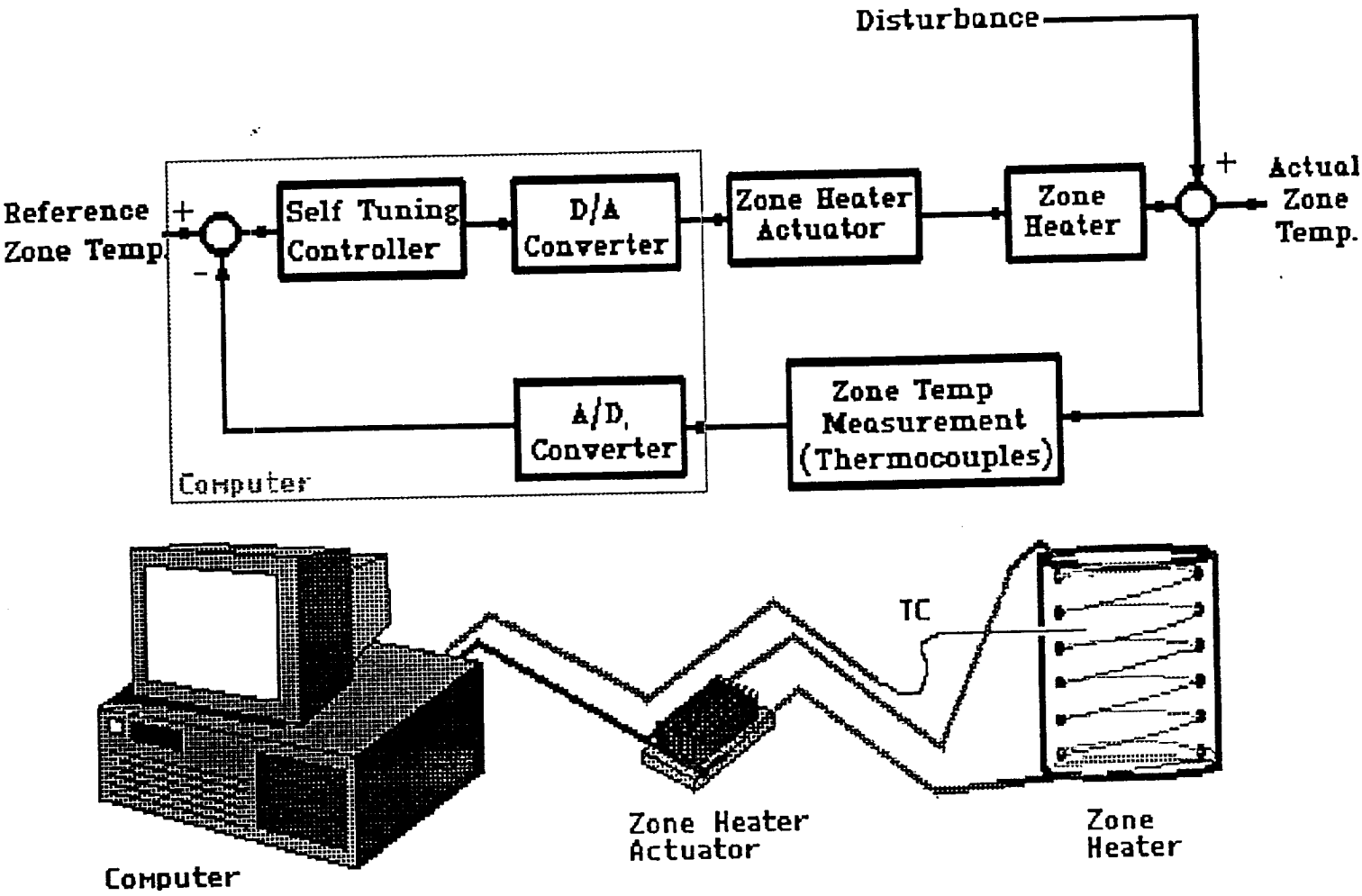


Figure 2b. Adaptive single input single output control system

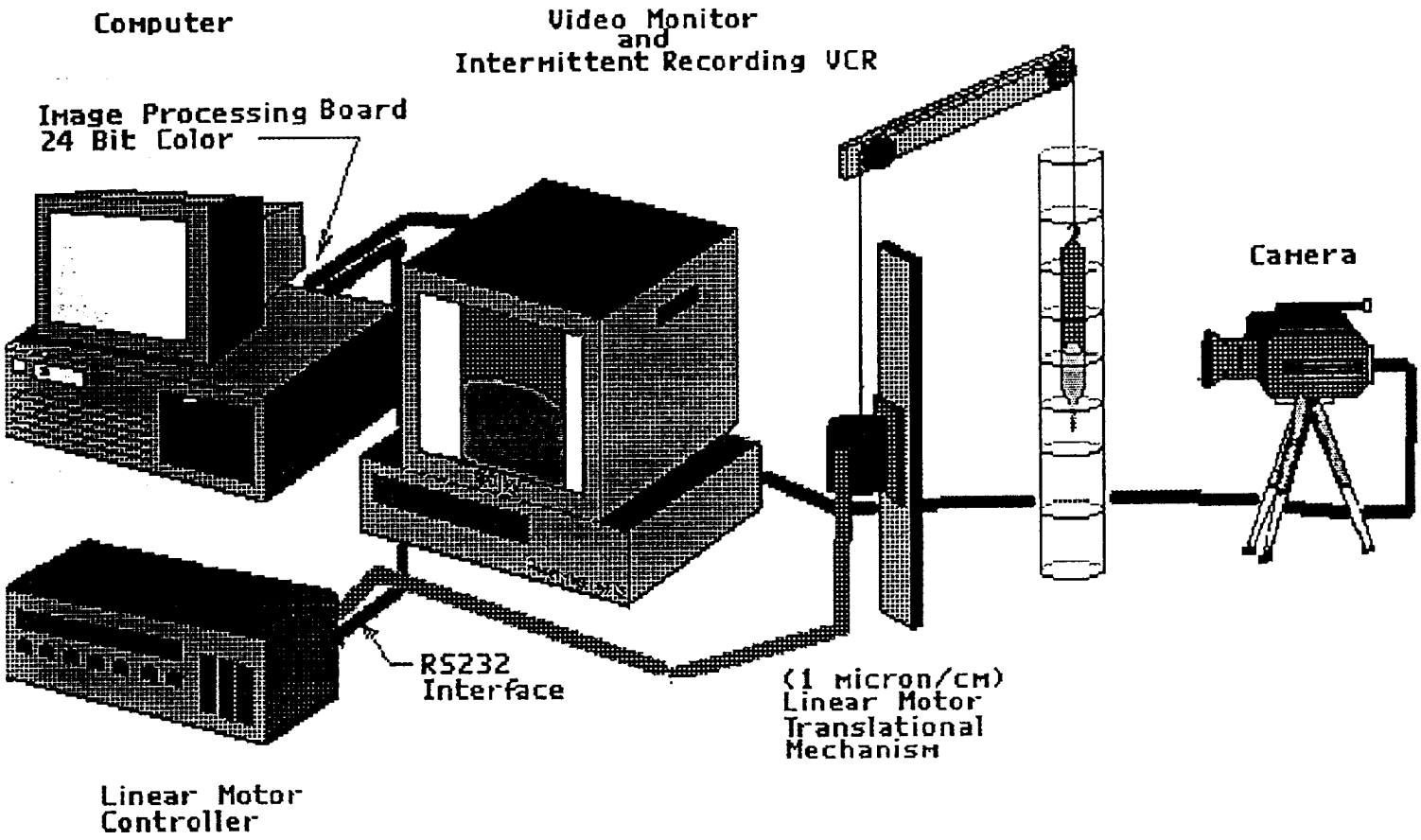
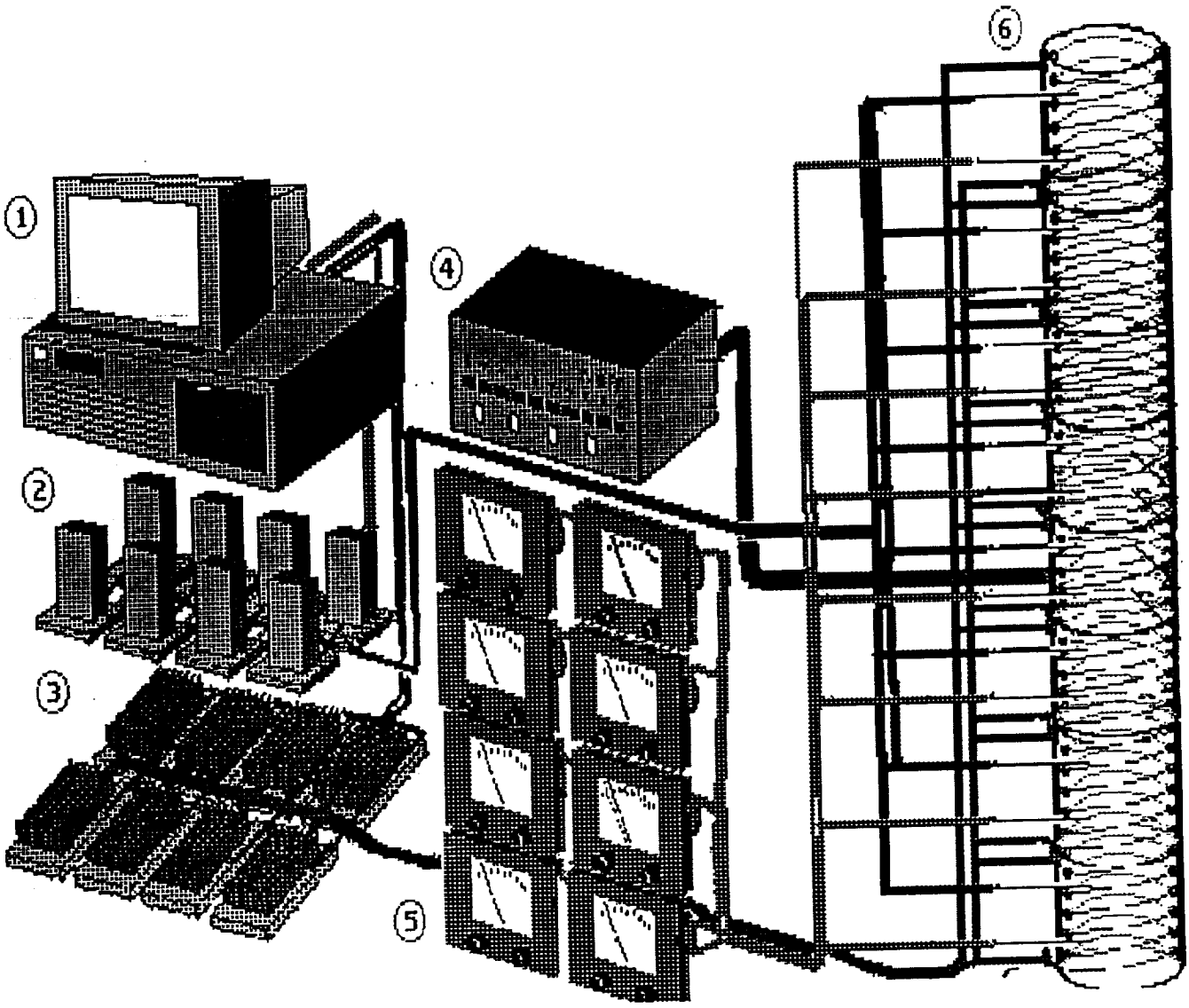


Figure 3. Image processing hardware



(1) Computer controller with A/D and D/A boards
(2) Thermocouple Tranducers
(3) Linear Actuators
(4) Independent Data Aquisition System
(5) API Meters, for emergency furnace shutdown.
(6) Furnace

Figure 4. Transparent furnace and the control system hardware

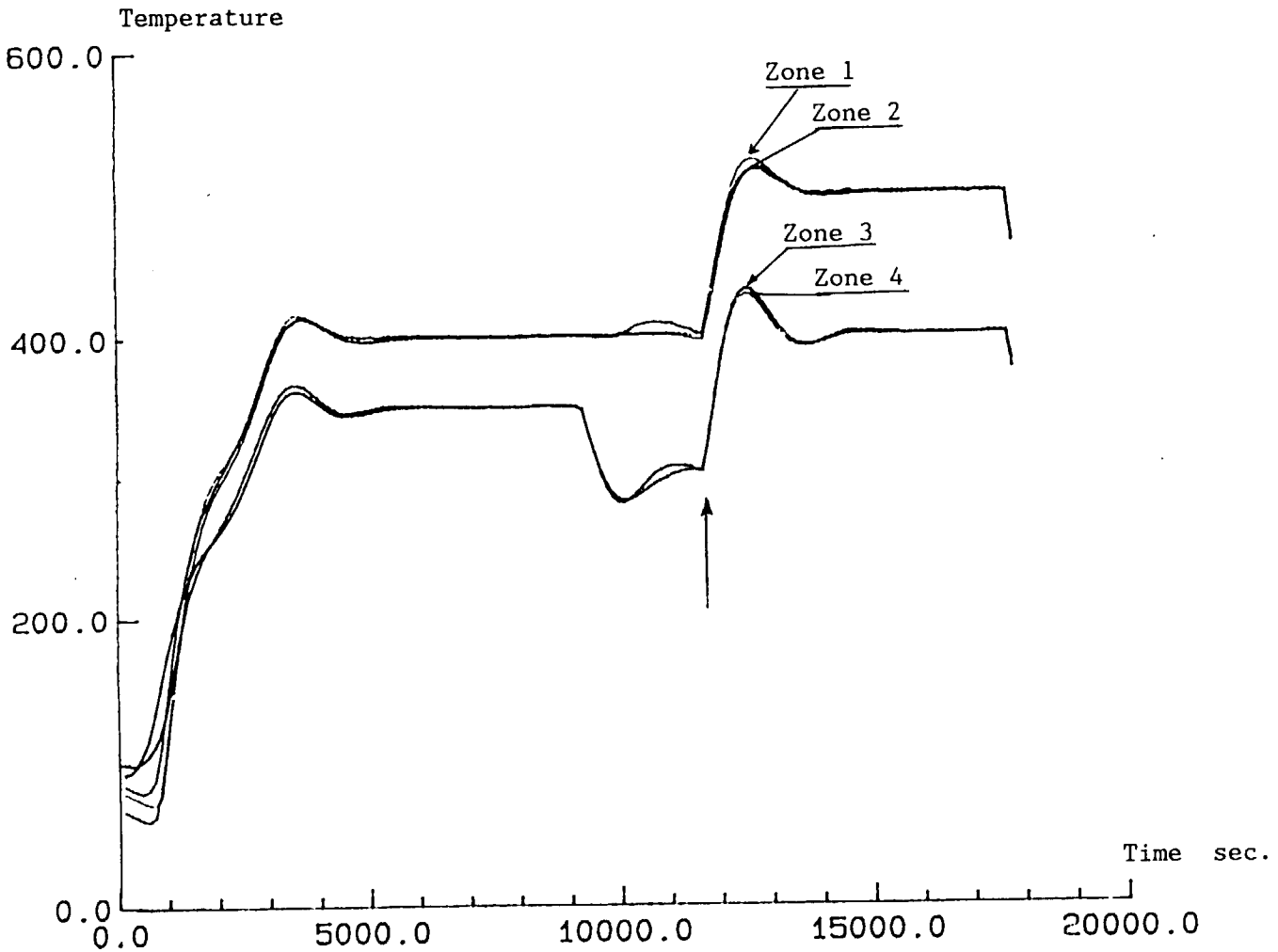


Figure 5. Responses of the furnace to step changes under computer control

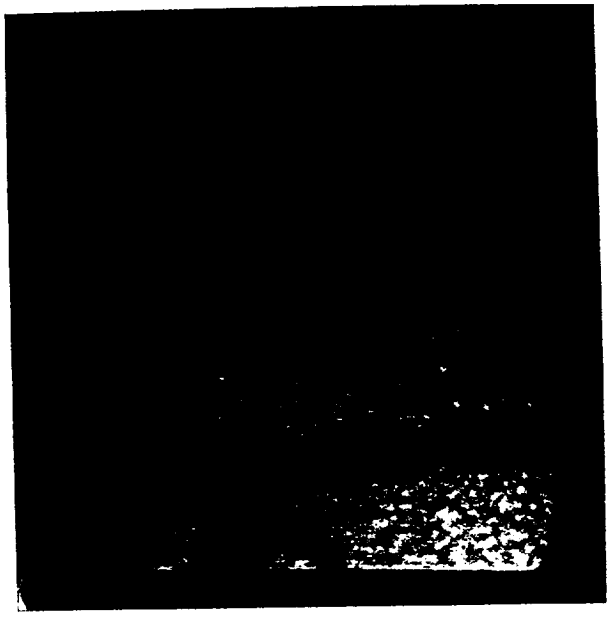


Figure 6a Unprocessed image

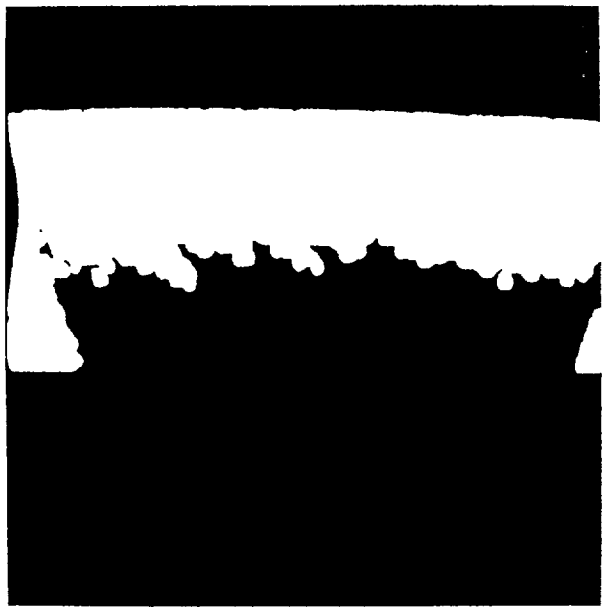


Figure 6c Noise is filtered out from image 6b.

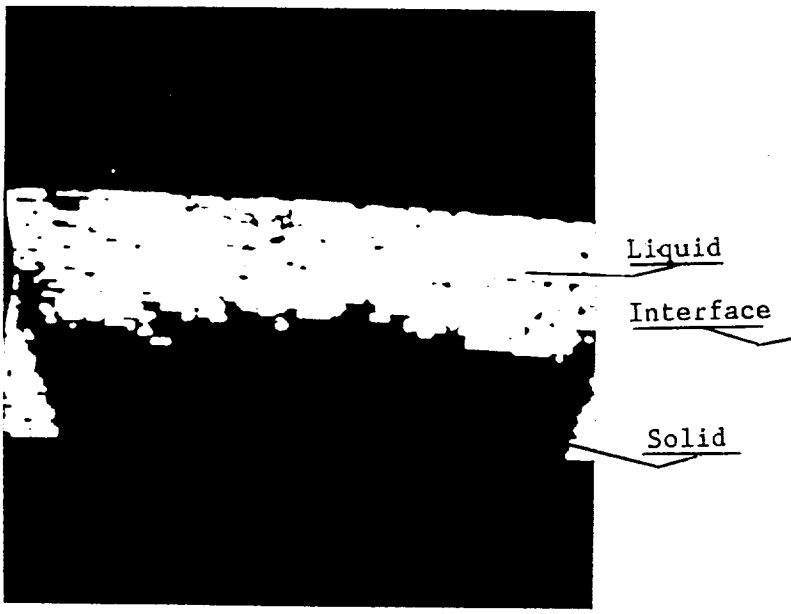


Figure 6b Histogram based image segmentation

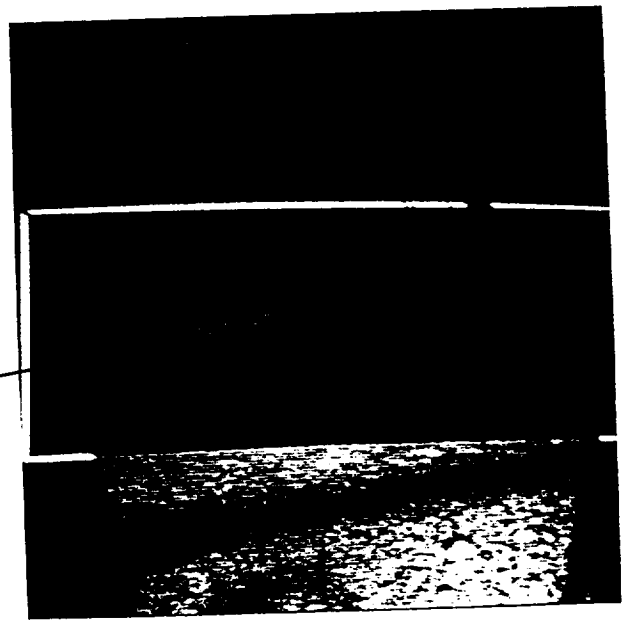


Figure 6d Gradient based interface detection

Figure 6 Stages of preliminary interface quantification

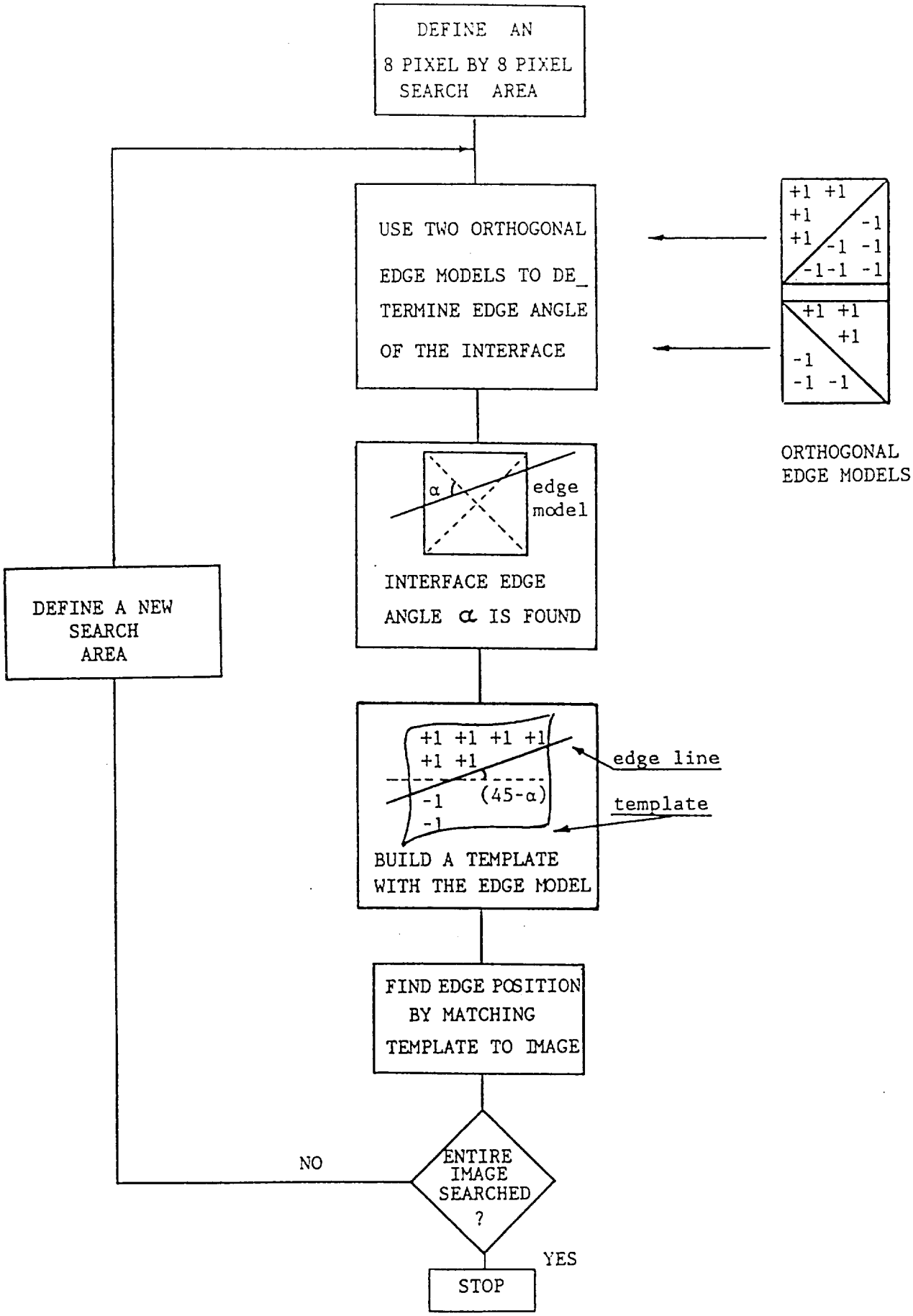


Figure 7. Flow chart for the determination of interface

Zone	a	b
1	-0.9900	0.0171
2	-0.9905	0.0151
3	-0.9917	0.0123
4	-0.9911	0.0138
5	-0.9905	0.0149
6	-0.9918	0.0120
7	-0.9894	0.0187
8	-0.9920	0.0146

Table 1. Dynamics of each zone identified in open loop experiment

Process model is $y(t)= -a\ y(t-1)+b\ u(t-1)+n(t)$

APPENDIX 1

A.1 Process Modeling

For the following process structure

$$\begin{aligned} y(t) &= a\,y(t-1) + b\,u(t-1) + e(t) \\ &= [y(t-1)\ u(t-1)] \begin{bmatrix} a \\ b \end{bmatrix} + e(t) \\ &= \phi^T \theta + e(t) \end{aligned} \tag{A1}$$

Process dynamics θ can be identified sequentially by, [Goodwin 1984]

$$\theta(t) = \theta(t-1) + \frac{P(t-1)\,\phi(t-1)}{1 + \phi^T(t-1)\,P(t-1)\,\phi(t-1)} [y(t) - \phi^T(t-1)\,\theta(t-1)]$$

where

$$P(t) = P(t-1) + \frac{P(t-1)\,\phi(t-1)\,\phi^T(t-1)\,P^T(t-1)}{1 + \phi^T(t-1)\,P(t-1)\,\phi(t-1)}$$

with the initial choices θ(0), P(0) = α I where α is a large number indicating little confidence on the initial choices.

A.2 Self-Tuned PID Controller

Following [Astrom and Hagglund 1988], the controller is given by

$$\textit{Proportional action: } P = K[b\,r(t) - y(t)] \tag{A3}$$

$$\textit{Derivative action: } \frac{Td}{N}\frac{dD}{dt} + D = -KT_d\frac{dy}{dt} \tag{A4}$$

$$\textit{Integral action: } \frac{dI}{dt} = \frac{K}{T_i}[r(t) - y(t)] \tag{A5}$$

where b, N are appropriate constants, Td is the derivative time constant, K is the controller gain and ($1/T_i$) is the gain associated with the integral action. If the sampling time is T then (A3-A5) can be discretized to

$$P(k) = K[b\,r(k) - y(k)] \tag{A6}$$

$$D(k) = \left(\frac{T_d}{T_d + NT}\right) D(k-1) - \left(\frac{KT_d N}{T_d + NT}\right)[y(k) - y(k-1)] \tag{A7}$$

$$I(k) = I(k-1) + \frac{KT}{T_i}[r(k) - y(k)] + \frac{T}{T_t}[u(k) - v(k)] \tag{A8}$$

where the last term in (A8) avoids the integration wind-up by subtracting the controller output determined by the controller, i.e.

$$v(k) = P(k) + D(k) + I(k) \tag{A9}$$

from the actual control output

$$u(k) = Saturation\ (v(k)) \tag{A10}$$

which goes to the process in the case of saturation. In (A8), T_t is a user specified parameter which effectively determines the wind-up recovery.

Once the process parameters (a, b) are identified, the process gain (k) and the time constant (τ) can be determined from

$$\frac{y(k)}{u(k)} = \frac{bz^{-1}}{1 - a\,z^{-1}} \quad \Leftrightarrow \quad \frac{Y(z^{-1})}{u(z^{-1})} = Z\left[\frac{1 - e^{-sT}}{s}\,\frac{k}{\tau s + 1}\right] = k\,\frac{(1 - e^{-T/\tau})z^{-1}}{1 - e^{-T/\tau}z^{-1}} \tag{A11}$$

which yields

$$b = k(1 - e^{-T/\tau}) \ , \ a = e^{-T/\tau} \tag{A12}$$

or

$$k = \frac{b}{1+a} \quad , \quad \tau = -T/\ell na \tag{A13}$$

Following the Ziegler and Nichols tuning rules, the controller parameters can be determined by

$$K = 1.2\,\frac{\tau}{kT} \quad ; \quad T_i = 2T \quad ; \quad T_d = T/2$$

where it is assumed that the process delay time is equal to the sampling period.

A.3 Self-Tuned Pole Placement Controller

The pole placement controller is assumed to be in the following form

$$u(k) = \frac{1}{R(z^{-1})}\left[T(z^{-1})\,r(k) - S(z^{-1})\,y(k)\right] \tag{A14}$$

where R (z^{-1}), S (z^{-1}) and T (z^{-1}) are polynomials to be determined. Writing the process dynamics from (A1) as

$$\frac{y(k)}{u(k)} = \frac{bz^{-1}}{1-az^{-1}} = \frac{B(z^{-1})}{A(z^{-1})} \tag{A15}$$

The dynamics of the feedback control system is given by

$$y(k) = \frac{BT}{AR + BS}\,r(k) \tag{A16}$$

where for convenience the argument (z^{-1}) in the polynomials has been dropped. If the desired characteristic polynomial for the feedback control system is given as P, then from (A16), it follows that

$$AR + BS = FP \tag{A17}$$

where F is an observer polynomial. To guarantee the integral effect, the polynomial R is designed to have an integrator, i.e.

$$R = Q(1 - z^{-1}) \tag{A18}$$

Under this condition (A17) becomes

$$(1 - az^{-1})(1 - z^{-1})(1 + qz^{-1}) + bz^{-1}(s_0 + s_1 z^{-1}) = (1 + fz^{-1})(1 + p_1 z^{-1} + p_2 z^{-2}) \tag{A19}$$

Equating coefficients associated with the equal powers of z gives,

$$q = \frac{-fp_2}{a} \quad ; \quad s_0 = \frac{f + p_1 + a + 1 - (fp_2/a)}{b} \quad ; \quad s_1 = \frac{fp_1 + p_2 - a - (-1 - a)(fp_2/a)}{b} \tag{A20}$$

under this solution and from (A16) and (A17) the closed loop feedback control system can be represented by

$$y(k) = \frac{B(z^{-1})T(z^{-1})}{F(z^{-1})P(z^{-1})} r(k) \tag{A21}$$

if

$$T(z^{-1}) = F(z^{-1}) \frac{P(1)}{B(1)}$$

then (A21) becomes

$$y(k) = \frac{B(z^{-1})P(1)}{P(z^{-1})B(1)} r(k) \tag{A22}$$

which yields

$$\bar{y} = \bar{r} \tag{A23}$$

APPENDIX 2. Control and image processing software

```
#include<stdio.h>
#include<ctype.h>
#include<dos.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<tardev.h>
#include "com.h"
#define live 1
#define width 0.75        /* ampule width in inches */

int Day=0,PlotI=301;

main()
{

   unsigned long            TimeIntoTest(unsigned long StartTime);
   int                      sure(char[20]);
   char                     GetRef();
   int                      ATD(int Channel);
   void                     d2a(int dio,int channel,int base);
   void        ZeroOut(void);
extern         void         DrawAxis(void);
   void                     PlotLine(float Y[8],float U[8],float Phi[8][2],int NPORT);
   void                     cl(void);
   void                     menu(int,int,int,int);
   void                     clearshow(void);
   char          getref(long int SITime,long int RTime);
   void                     ccom();
   void                     getline(unsigned char m[512],unsigned char *no);
   void                     heunkel(unsigned char searchln[512],unsigned char *no,unsigned char refln[512]);
   void                     drawln(unsigned char v[256],unsigned char len);


   struct time now;
   struct date dnow;
   float       c[8][2]={{-5.4395,0.9867},{1.9618,1.01},{-6.7497,0.9917},{8.9301,0.9869},
               {-1.6935,0.9979},{0.6859,0.9744},{-1.7250,1.0032},{2.2191,0.9964}},
               Phi[8][2],Theta[8][2],P[8][2][2],PxPhi[2],Y[8],U[8],r[8],ref[8],ry[8],
               den,res,Umax,delh,input,
               s0,s1,rlu,rbar,pole1,pole2,alpha,d,incden,UDelay[8],YDelay[8],
               temp;
   float tspeed,ampulescale;

   char                     Status,cmd,filen[12],comstring[100];

   unsigned long            StartTime,RTime,SITime,DataTime;
   unsigned char i,j,k,Done,NPORT,PlotI,ys,xs,Simulate,Save,ReDraw,showo,
                            datai,identify,h,m,s,hs,showid,interface,tempc,
                            searchln[512],lnlen,refln[512],reflnlen,refoffset,trans;
   int                      movi,dwell,tdist,delh2;


   FILE *f;
```

```
   Simulate    = 0;
ReDraw      = 0;
showo       = 0;
showid          = 0;
identify    = 1;
directvideo = 0;
interface   = 0;
trans           = 0;
textcolor(3);

Umax   = 600.;
NPORT  = 7;
Save   = 0;
Done   = 0;
delh   = 5.;
delh2  = 100*delh;
datai  = 1.;
PlotI  = 301;
RTime  = 0;
SITime = 0;

pole1 = -1.5;
pole2 = 0.56;
d     = 1+pole1+pole2;
alpha = -0.5;
incden= 0.00001;

GraphInit(-1);
SetPageMode(4);
if(live) SetBLiveMode();
else     GetPic("furn256.tga",0,0,0);

for(k=0;k<=NPORT;k++)
{           for(i=0;i<=1;i++)
                    for(j=0;j<=1;j++)
                    {           Phi[k][j]    = 1.;
                                Theta[k][0]  = -0.9;
                                Theta[k][1]  = 0.06;
                                P[k][i][j]   = 0.;
                                if(i==j)  P[k][i][j] = 1000.;
                     }
            UDelay[k] = 0.;
            YDelay[k] = 0.;
            r[k]      = 1.;
            Y[k]      = 1.;
            U[k]      = 0.;
}

                                                                    /* get settings */

DrawAxis();
menu(Save,showo,showid,interface);


                                                                    /* zero outputs */

if(!Simulate) ZeroOut();
```

```
    do
    {
            StartTime=TimeIntoTest(0);
                                                        /* READING TIME         */
            SITime+=1;
                                                        /* REFERENCE TEMP       */
            Status=getref(SITime*delh,RTime);
            switch(Status){
                    case 'U':
                            for(j=0;j<=NPORT;j++) ref[j]=ry[j]+(r[j]-ry[j])*SITime/RTime*delh;
                            gotoxy(35,9);
                            h=(RTime-SITime*delh)/3600.;
                            m=(RTime-SITime*delh-h*3600.)/60.;
                            s=(RTime-SITime*delh-h*3600.-m*60.);
                            cprintf("REMAINING RAMP TIME -> %02d:%02d:%02d",h,m,s);
                            break;
                    case 'S':for(j=0;j<=NPORT;j++) ref[j]=r[j];break;}

                                                        /* READING TEMPERATURES */
                                                        /* SIMULATED PROCESS    */
            if(Simulate)
            for(i=0;i<=NPORT;i++)       Y[i] = -0.9*Phi[i][0]+0.088823*Phi[i][1];

                                                        /* ACTUAL PROCESS       */
            if(!Simulate)
            for (i=0;i<=NPORT;i++)
                {
                Y[i] = c[i][0]+c[i][1]*(ATD(i)+151.66)/6.0667;
                if (Y[i]>1000.)
                                {
                                gotoxy(20,15);cprintf("Thermocouple Problem !!");
                                ZeroOut();
                                Done=1;
                                }
                }
/*          cprintf("\n");*/

                                                         /* DETERMINE OUTPUT */
            for(j=0;j<=NPORT;j++)
            {
                    r1u  = -1.0*alpha*pole2/(Theta[j][0]+incden);
                    s0   = (alpha+pole1+1.0-Theta[j][0]-r1u)/(Theta[j][1]+incden);
                    s1   = ((alpha*pole1+pole2)-(Theta[j][0]-1.)*r1u+Theta[j][0])/(Theta[j][1]+incden);
                    rbar = ref[j]*(1.+alpha)*d/(Theta[j][1]+incden);

                    U[j] = (1.0-r1u)*U[j]+r1u*UDelay[j]-s0*Y[j]-s1*YDelay[j]+rbar;

                    if (U[j]>=Umax)             U[j] = Umax;
                    if (U[j]<0)                 U[j] = 0;
            }

                                                /* SEND POWER       */
            if(!Simulate && !Done){
            for(j=0;j<=5;j++)           d2a(U[j],j,512);
            for(j=0;j<=1;j++)           d2a(U[j+6],j+4,528);}
```

```
                                        /* SAVE DATA       */
        if(Save && TimeIntoTest(DataTime)>=datai*100)
        {
                DataTime=TimeIntoTest(0);
                gettime(&now);
                fprintf(f,"%02d:%02d:%02d.%02d",now.ti_hour,now.ti_min,now.ti_sec,now.ti_hund);
                for(i=0;i<=NPORT;i++)       fprintf(f," %7.2f",Y[i]);
                if(interface)
                {
                        if(live)
                        {           GrabFrame();
                                    SetDispMode();
                        }
                        else        GetPic("furn256.tga",0,0,0);
                        heunkel(searchln,&lnlen,refln);
                        for(i=0;i<lnlen;i+=5)
                        {           temp=(searchln[256+i]-refln[256+i+refoffset]+0.)*ampulescale;
                                    fprintf(f,"%6.3f",temp);
                        }
                        if(live) SetBLiveMode();
                        else        GetPic("furn256.tga",0,0,0);
                }
                fprintf(f,"\n");
        }
                                                        /* IDENTIFY SYSTEM */
        if(identify && r[3]>1)
         for(k=0;k<=NPORT;k++)
         {          for(i=0;i<=1;i++)
                    {           PxPhi[i]=0.;
                                for (j=0;j<=1;j++)
                                        PxPhi[i]+=P[k][i][j]*Phi[k][j];
                    }

                    den=0.;
                    for(i=0;i<=1;i++)
                                den += Phi[k][i]*PxPhi[i];
                    den += 0.0001;

                    for(i=0;i<=1;i++)
                                for (j=0;j<=1;j++)
                                        P[k][i][j] -=PxPhi[i]*PxPhi[j]/den;
                    res=0.;
                    for(i=0;i<=1;i++)
                                res += Phi[k][i]*Theta[k][i];
                    res =Y[k]-res;

                    if(  fabs(res)>0.0  )
                                for( i=0;i<=1;i++ )
                                        Theta[k][i]+=PxPhi[i]/den*res;
                    if (Theta[k][1]<0 || Theta[k][1]>2) Theta[k][1]=0.06;
                    if (Theta[k][0]>0 || Theta[k][0]<-1) Theta[k][0]=-0.9;
         }
                                                        /* PLOT DATA       */
PlotLine(Y,U,Phi,NPORT);
    while(TimeIntoTest(StartTime)<delh2)
```

```
        while(TimeIntoTest(StartTime)<delh2)

                                        /* KEY PRESSED ?   */
        if(kbhit())
        {
                cmd=toupper(getch());
                switch(cmd)
                {
                        case 'T' : if(sure("CHANGE TEMP")){
                                        for(j=0;j<=NPORT;j++) {
                                        badscan:gotoxy(35,9);
                                                if(!movi) scanf(" %c",&tempc);
                                                if(j==0) cprintf("ENTER ZONE %d NEW TEMP: ",j+1);
                                                else    cprintf("ENTER ZONE %d NEW TEMP (%5.1f): ",j+1
,r[j-1]);

                                                movi=scanf(" %f",&temp);
                                                if(movi) r[j]=temp;
                                                ry[j]=Y[j];
                                                cl();}
                                        gotoxy(35,9);
                                        cprintf("ENTER RAMP TIME (hrs): ");
                                        scanf("%f",&temp);RTime=temp*3600.;
                                        SITime=0.;
                                        cl();}
                                        break;
                        case 'E' : if(sure("EXIT"))
                                        Done=1;
                                        break;
                        case 'K' : if(sure("START/CHANGE TRANSLATION")){
                                        trans=1;gotoxy(35,9);
                                        installcomdrivers (1);                  /* Set up for COM1:
*/
                                        setbaud (2400);                                 /* 2400 Baud
N 8 1                   */
                                        cprintf("ENTER THE TRANS DISTANCE (+/- cm): ");
                                        scanf("%d",&tdist);cl();
                                        cprintf("ENTER THE TRANS SPEED (CM/DAY): ");
                                        scanf("%f",&tspeed);cl();
                                        if (tdist<0) {movi=-1;tdist*=-1;}else movi=1;
                                        dwell=8778./tspeed;
                                        while (dwell<500){
                                                if(movi<0) {movi--;dwell=8778./tspeed*-movi;}
                                                else       {movi++;dwell=8778./tspeed*movi; }
                                        }
                                        com_printf ("LOAD;\n\r");
                                        com_printf ("10MODE M_INC   NULL    NULL            ;\n\r");
                                        com_printf ("15MATH   Q1    =          0            ;\n\r");
                                        com_printf ("20LOOP         %d                      ;\n\r",tdist);td
ist=9843./movi;if(tdist<0) tdist*=-1;
                                        com_printf ("30LOOP         %d                      ;\n\r",tdist);
                                        com_printf ("40MOVI         %d       0      0       ;\n\r",movi);
                                        com_printf ("45DISP  LNUM    BLANK    BLANK   BLANK ;\n\r");
                                        com_printf ("50MATH  Q1  =   Q1       +        1     ;\n\r");
                                        com_printf ("55WRITE    1    0    0    Q1            ;\n\r");
                                        com_printf ("60DWELL        %d   MILLISECONDS       ;\n\r",dwell);
                                        com_printf ("70ENDLOOP                              ;\n\r");
                                        com_printf ("80ENDLOOP                              ;\n\r");
                                        com_printf ("90DONE                                 ;\n\r");
                                        com_printf ("*;");
                                        uninstallcomdrivers ();

                                        }
```

```
                case 'A' : if(!Save && sure("START AQUISITION")){
                                badfile:cl();
                                        cprintf("ENTER THE FILE NAME: ");
                                        scanf("%s",filen);
                                        f=fopen(filen,"w");
                                        if(f==NULL) goto badfile;
                                        gettime(&now);
                                        getdate(&dnow);
                                        fprintf(f,"File Started  %02d/%02d/%02d at %02d:%02d:%02d.%02d\n",dn
ow.da_mon,dnow.da_day,dnow.da_year,
                                                now.ti_hour,now.ti_min,now.ti_sec,now.ti_hund);
                                        Save=1;cl();gotoxy(35,9);
                                        cprintf("ENTER THE SAMPLE INTERVAL (s): ");
                                        scanf("%d",&datai);cl();
                                        DataTime=TimeIntoTest(0);
                                        menu(Save,showo,showid,interface);}
                        case 'L' : if((Save && !interface && sure("RECORD INTERFACE PROFILE"))
                                        ||(Save && interface && sure("Change SEARCH and/or REFERENCE Line(s)"))){
                                        if(live){GrabFrame();SetDispMode();}

                                        if(!interface || sure("Change SEARCH Line")){
                                                clearshow();gotoxy(1,1);
                                                cprintf("Enter an Approximate SEARCH Line\n\n\r");
                                                cprintf(" -mark points from LEFT to RIGHT\n\r");
                                                cprintf(" -Use SPACE to mark points       \n\r");
                                                cprintf(" -Press RETURN when done");
                                                getline(searchln,&lnlen);
                                                ampulescale=width*2.54/(lnlen+0.);}
                                        if(!interface || sure("Change REFERENCE Line")){
                                                clearshow();gotoxy(1,1);
                                                cprintf("Enter the REFERENCE Line\n\n\r");
                                                cprintf(" -mark points from LEFT to RIGHT\n\r");
                                                cprintf(" -Use SPACE to mark points       \n\r");
                                                cprintf(" -Press RETURN when done\n\n\r");
                                                cprintf(" *Be sure this line is AT LEAST AS\n\r");
                                                cprintf("  LONG AS THE SEARCH LINE");
                                                getline(refln,&reflnlen);
                                                refoffset=refln[i]-searchln[0];}
                                                cl();clearshow();interface=1;
                                                menu(Save,showo,showid,interface);}
                                                if(live) SetBLiveMode();
                                                break;
                        case 'C' : if(Save && sure("STOP AQUISITION")){
                                                fclose(f);
                                                Save=0;
                                                menu(Save,showo,showid,interface);}
                                                break;
                        case 'O' : if(!showo && sure("SHOW OUTPUT")){
                                                showo=1;
                                                showid=0;
                                                clearshow();
                                                menu(Save,showo,showid,interface);}
                                                break;
                        case 'I' : if(!showid && sure("SHOW IDENTIFIED PARAMETERS")){
                                                showid=1;
                                                showo=0;
                                                clearshow();
                                                menu(Save,showo,showid,interface);}
                                                break;
```

```
                    case 'H' : if(showo && sure("HIDE OUTPUT")){
                                                clearshow();
                                                showo=0;
                                                menu(Save,showo,showid,interface);}
                                                break;
                            case 'J' : if(showid && sure("HIDE IDENTIFIED PARAMETERS")){
                                                clearshow();
                                                showid=0;
                                                menu(Save,showo,showid,interface);}
                                                break;
                }
        }
        for(j=0;j<=NPORT;j++)
        {
                YDelay[j]    = Y[j];
                UDelay[j]    = Phi[j][1];
                Phi[j][1]    = U[j];
                Phi[j][0]    = -Y[j];
        }

        if(showo)
        {
                gotoxy(4,1);cprintf("Zone Ttarg   Tact    P.O  ");
                for(i=0;i<=NPORT;i++)
                {
                        gotoxy(5,2+i);
                        cprintf("%d  %6.1f %6.1f %8.4f",i+1,r[i],Y[i],U[i]/4025.*100.);
                }
        }
        if(showid)
        {
                gotoxy(6,1);cprintf("    A[1]        B[1]  ");
                for(i=0;i<=NPORT;i++)
                {       gotoxy(5,2+i);
                        cprintf(    "  %8.4f   %8.4f",Theta[i][0],Theta[i][1]);
                }
        }
                                                                        /* DELAY          */
        if(SITime==3) delh=TimeIntoTest(StartTime)/100.;
        } while(!Done);

        if(!Simulate) ZeroOut();
        if(Save) fclose(f);
        closegraph();
}

unsigned long TimeIntoTest(unsigned long StartTime)
{
    struct time now;
    long TimeNow;
    gettime(&now);
    TimeNow=(long)now.ti_hour*360000+(long)now.ti_min*6000+(long)now.ti_sec*100+now.ti_hund*1;
    if (TimeNow<StartTime  && StartTime !=0) return(TimeNow*24*360000-StartTime);
    return(TimeNow-StartTime);
}
```

```
char getref(long SITime,long RTime)
{               if (SITime  < RTime  && RTime>0 )       return('U');
   else return('S');
}

int ATD(int Channel)
{
   unsigned int  Base,wordl,wordh;
   unsigned char   xl,xh;
   Base=768;
   outport(Base+2,Channel);
   outport(Base,0);
/* while(inport(Base+8)>>7)==1);*/
   delay(100);
   xl=inportb(Base);xh=inportb(Base+1);
   wordl=xl+0.;wordh=xh+0.;
/* cprintf("%d %d \r\n",wordl,wordh);*/
   return((wordh<<4)+(wordl>>4));
}

void d2a(int dio,int Channel,int Base)
{
   int xh,xl;
   xh=dio/256.;
   xl=dio-256.*xh;
   outportb(Base+2*Channel,xl);
   outportb(Base+1+2*Channel,xh);
}

void ZeroOut()
{
   int j;
   for(j=0;j<=5;j++)  d2a(0,j,512);
   for(j=0;j<=1;j++)  d2a(0,j+4,528);
}

void cl()
{
   int i;
   gotoxy(35,9);
   for (i=1;i<48;i++) cprintf(" ");
   gotoxy(35,9);
}

void clearshow()
{  int i,j;
   for (i=1;i<=9;i++)
   {           gotoxy(1,i);
               for (j=1;j<35;j++) cprintf(" ");
   }
}

int  sure(char s[12])
{  char cmd='x';
   cl();
   gotoxy(35,9);
   cprintf("%s (y/n) ? ",s);
   while( cmd!='Y' && cmd!='N') cmd=toupper(getch());
   cl();
   return((cmd=='Y') ? 1 : 0);
}
```

```
void getline(unsigned char xf[512],unsigned char *no)
{   long color[6],t=0;
    int i,key,xa;
    unsigned char j,x,y;

    j=0;
    x=50;y=50;
    do{
    for(i=-1;i<2;i++)  {GetPix(&color[i+1],x+i,y,0);PutPix(&t,x+i,y,0);}
    for(i=-1;i<2;i+=2) {GetPix(&color[i+4],x,y+i,0);PutPix(&t,x,y+i,0);}

    while(i && i!=32 && i!=13)
                              i=getch();
    if(i==0){
    for(i=-1;i<2;i++)  PutPix(&color[i+1],x+i,y,0);
    for(i=-1;i<2;i+=2) PutPix(&color[i+4],x,y+i,0);
    key=getch();
    switch(key)
    {           case 75:x-=3;break; /* left  */
                case 77:x+=5;break; /* right */
                case 72:y+=3;break; /*  up   */
                case 80:y-=5;break; /* down  */
    }}

    else if (j==0 && i!=13) {xf[0]=x;xf[256]=y;j++;xa=0;}
    else if (i!=13){key=x-xf[xa];if (key<0) key*=-1;xf[key+xa]=x;xf[256+key+xa]=y;
    for(j=1;j<key;j++)
    {           xf[j+xa]=xf[xa]+j*key/(xf[key+xa]-xf[xa]+0.);
                xf[256+j+xa]=xf[256+xa]+(xf[256+key+xa]-xf[256+xa])/(xf[key+xa]-xf[xa]+0.)*(xf[j+xa]-xf[xa]);
                PutPix(&t,xf[j+xa],xf[256+j+xa],0);
    }
    xa+=key;}
    }while(i!=13);
    *no=xa;
}
void drawln(unsigned char v[256],unsigned char len)
{   unsigned char i;
    long t=31<<5;
    for(i=0;i<len+1;i++)
                PutPix(&t,v[i],v[256+i],0);
}
void menu(int save,int showo,int showid,int identify)
{
    int i,xmin=20,ymin=21;
    char str[11][40]={"'T' -> Change Temperature References",
                              "'E' -> Exit",
                              "'A' -> Acquire Temperature Data",
                              "'C' -> Close Data File",
                              "'O' -> Show Outputs",
                              "'H' -> Hide outputs",
                              "'I' -> Show Identification Parameters",
                              "'J' -> Hide Identification Parameters",
                              "'L' -> Change Quant Ref/Search Lines",
                              "'L' -> Quantify Interface",
                              "'K' -> Start / Change Translation"};

    setviewport(280,0,639,100,1);
    setcolor(BLUE);
    setfillstyle(LTSLASH_FILL,BLUE);
    bar3d(10,10,340,90,0,2);
    setcolor(WHITE);
```

```
        for (y=0;y<N;y++)
                for (x=0;x<N;x++)
                        d[N*y+x] = x*sinalpha+(N-y+1)*cosalpha;

    for(i=0;i<N*N;i++)
                for(j=i+1;j<N*N;j++)
                        if(d[o[j]]<d[o[i]])
                        {       mini = o[j];
                                o[j] = o[i];
                                o[i] = mini;
                        }

    for(i=0;i<N*N;i++)
    {       sum = 0.;
            for(j=0;j<=i;j++)
                        sum += col[o[j]]-mean;
            if(sum>max || i==0){ maxi = i;max = sum;}
    }

    if(line){
    for (x=0;x<N;x++)
                for (y=0;y<N;y++)
                        if(N*y+x>maxi)
                        {       mini=searchln[256+k];
                                searchln[256+k]=yg+y;
                                color=0;        PutPix(&color,xg+x,yg+y,0);
                                color=(long)255<<16;    PutPix(&color,xg+x,refln[xg+x+256-refln[0]],0);
                                if(x==N-1) yg=searchln[256+k]-N/2.;
                                y=N;k++;
                        }}
    else{
    for (x=0;x<N;x++)
                for (y=0;y<N;y++)
                {       if(N*y+x<=maxi) {color=31<<5;PutPix(&color,xg+x,yg+y,0);}
                        else {color=0;PutPix(&color,xg+x,yg+y,0);}
                        if(N*y+x>maxi && x==N-1)
                        {       ymin=ymin+y-N/2.;
                                y=N;
                        }
                }}
    }
}
void PlotLine(float Y[8],float U[8],float Phi[8][2],int NPORT)
{
    int         xs,ys,i,j,c[8]={2,3,4,15,13,12,11,14},color;
    float       pos[8]={1.5,4.5,6.5,8.5,10.5,12.5,15.5,18.5};
    static int PlotI,ReDraw,yold[8];

    if (PlotI>263)                              /* Erasing TEMP v Time */
    {       setviewport(56,150,319,323,1);
            clearviewport();
            PlotI=0;
            setlinestyle(4,0x8080,1);               /* Grid Lines */
            setcolor(4);
            for(i=1;i<=7;i++)
            {       line(-1,175-i*25,262,175-i*25);
                    yold[i]=0;
            }
            setcolor(15);
            setlinestyle(0,2,1);
    }
```

```
        setviewport(55,150,319,323,1);
    setlinestyle(4,0xffff,1);
    color=getcolor();
    for(j=0;j<=NPORT;j++)                    /* Plotting TEMP vs TIME */
                {
                ys = 175-25./100.*(Y[j]+2)+1;
                xs = PlotI;
                setcolor(c[j]);
                line(xs-1,yold[j],xs,ys);
                yold[j]=ys;
                }
    ReDraw=0;
    setcolor(color);
    for(j=0;j<=NPORT;j++) if(fabs(Y[j]+Phi[j][0])>0.5) ReDraw=1;

    if(ReDraw || PlotI==0)                   /* Plotting POSITION vs DISP */
    {
    setviewport(371,150,639,314,0);
    clearviewport();

    setlinestyle(4,0xffff,1);
    for(j=0;j<=NPORT;j++)
    {           xs = 35./100.*(Y[j]+0.15);
                ys = 5+pos[j]*2.54*32./10.;
                if(j>0) lineto(xs,ys);
                else    moveto(xs,ys);
    }
    setlinestyle(4,0x8080,1);                    /* Grid Lines */
    setcolor(10);
    for(i=1;i<=7;i++)
                line(35*i,0,35*i,165);
    setcolor(15);
    }
    PlotI+=1;
}
```

```
        outtextxy(120,ymin-9,"Action Menu");
    setcolor(WHITE);          outtextxy(xmin,ymin+0*10+50,str[0]);
    setcolor(RED);                    outtextxy(xmin,ymin+1*10+50,str[1]);
    setcolor(WHITE);

    if (!save)                outtextxy(xmin,ymin+30,str[2]);
    else                              outtextxy(xmin,ymin+30,str[3]);
    if (!showo)               outtextxy(xmin,ymin+10,str[4]);
    else                              outtextxy(xmin,ymin+10,str[5]);
    if (!showid)              outtextxy(xmin,ymin+20,str[6]);
    else                              outtextxy(xmin,ymin+20,str[7]);
    if (save && identify)     outtextxy(xmin,ymin+0,str[8]);
    if (save && !identify)    outtextxy(xmin,ymin+0,str[9]);
                                      outtextxy(xmin,ymin+40,str[10]);


}

void heunkel(unsigned char searchln[512],unsigned char *no,unsigned char refln[512])
#define N 8
#define M N*N
{

    unsigned char o[M],col[M],xg,yg,x,y,i,j,maxi,mini,ymin,k=0;
    double                    sinalpha,cosalpha,tannum,tanden,d[M],pi;
    long                      color;
    float                     mean,sum,max=0;
    int r,g,b,c,t;

    pi=acos(-1.);
    yg=searchln[256]-N/2.;

    for(xg=searchln[0];xg<=searchln[*no] && xg>=searchln[0];xg+=N)
    {

    mean=0.;
    for (y=0;y<N;y++)
                for (x=0;x<N;x++)
                {           GetPix(&color,xg+x,yg+y,0);
                            col[N*y+x] = ((color)>>8)&255);
                            mean           += col[N*y+x];
                            o[N*y+x]   = N*y+x;
                }

    mean   /= (N*N+1.);
    tannum  = 0.;
    tanden  = 0.;
    for (y=0;y<N;y++)
                for (x=0;x<N;x++)
                {     if(x<y)               tannum += col[N*y+x];
                else if(x!=y)             tannum -= col[N*y+x];
                     if(x+y>N-1)             tanden += col[N*y+x];
                else if(x+y!=N-1)            tanden -= col[N*y+x];
                }

    if(tanden!=0.)
    {           sinalpha = sin(atan2(tannum,tanden)-pi/4.);
                cosalpha = cos(atan2(tannum,tanden)-pi/4.);
    }
    else {sinalpha = 1.;cosalpha = 0.;}
```

Lorene Albergottie
Grants Officer
NASA Lewis Research Center
Cleveland Ohio 44135

Subject: NCC3-109, final report

Dear L. Albergottie

A. 3 copies of the Final Technical Report are delivered to the NASA Technical officer.

B. 2 copies ~~to~~ are sent to
NASA Scientific and Technical Information Facility
P.O. Box 8757
BWI Airport, MD 21240

In your letter, regarding ~~to~~ item 2, can you please send me the Standard Form 272. I believe that items 3 to 5 do not apply. ~~Because all the items bought are currently installed at NASA.~~ All equipment bought are installed at NASA. Please advise on items 3 to 5.

~~I apologize for the delay~~

Also, Enclosed please ~~find~~ find a copy of the report :- I did not ~~forget~~ if you needed one

Sincerely

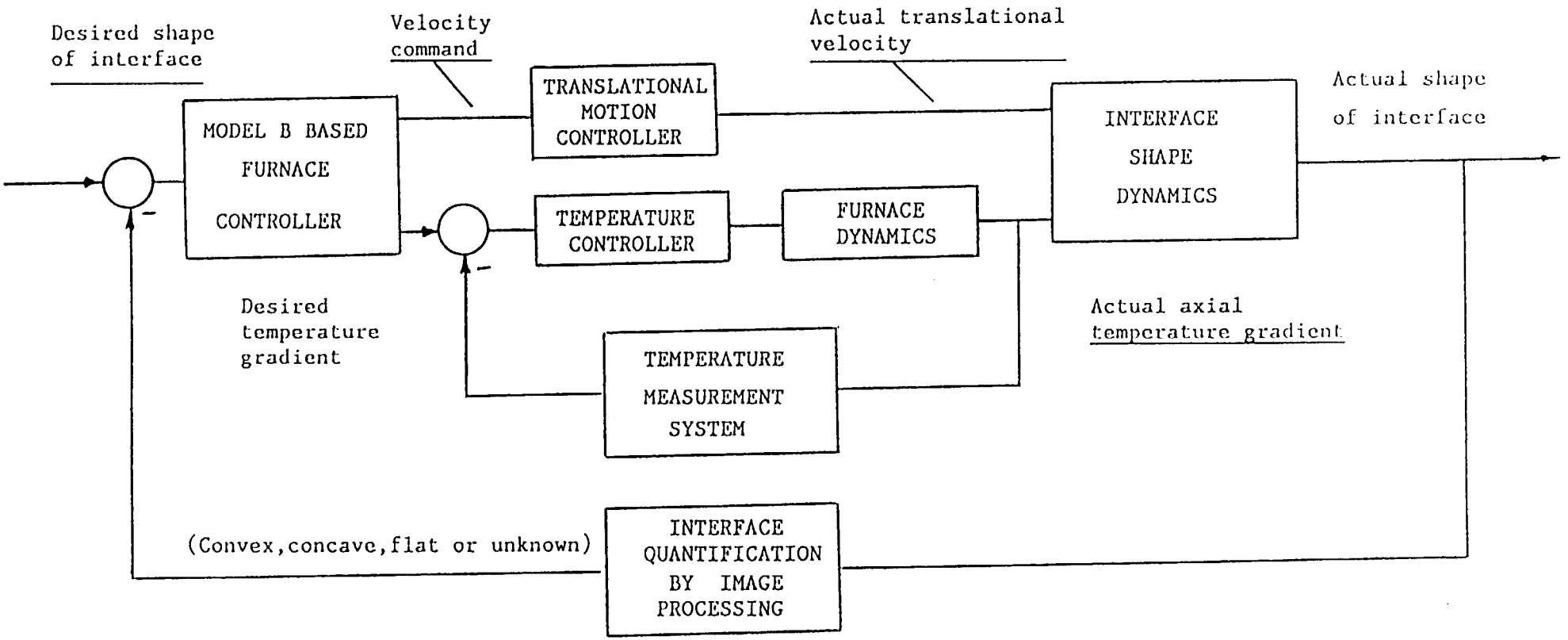


Figure 2. Interface shape control
(applicable only for transparent furnace and materials)