THE UNIVERSITY OF CALGARY

Real-Time Computer Control for Wafer in Rapid Thermal Processing

by

Ying Diana Yu

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGEREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

March 1997

ABSTRACT

This study explores real-time computer control for a wafer in Rapid Thermal Processing (RTP).

The general purpose of this thesis is to design and to implement two real-time software applications for wafer RTP temperature control. One application emulates the actual wafer RTP system and the other one acts like a digital controller to control the temperatures across the wafer to near uniformity when following a fast temperature trajectory. The design of this digital control system is also one of the topics discussed in this thesis.

These two applications are designed to be separately installed in two personal computers (PCs) and to communicate only through input and output (I/O) boards. Therefore, the emulator application can be replaced by actual wafer RTP system without changing the controller application.

In this thesis, the implementation is done within the Unix environment as a study case. The suggestion of installation of these two software applications into two PCs, and the design of user graphic interfaces are also discussed.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Dr. Steven Norman, for guiding, encouraging and supporting me during the entire course of this work. I am very grateful for the financial assistance that Dr. Norman and the department offered me during the course of my studies.

I would also like to extend my appreciation to Dr. W.C. Chan and Dr. M. Sideris for serving on my examination committee.

Lastly, I would like to thank my parents and my friends. To my parents, Baogui Yu and Hongren Cheng, I am deeply thankful for your loving and support over all my years as a student. To my friend Boon Phiaw Tan, I thank you for giving me so much help. To Huang Shu, Jian He, Mingming Liu, Wenli Tan and Arthur Wong thanks for being there when I needed you.

DEDICATIONS

To Mom & Dad

for loving me and guiding me

in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND NOTATIONS

Symbol	Section	Meaning
A_i:	2.2	exposed surface area of element i
C:	2.2	diagonal matrix contenting heat capacities of wafer elements
C_p:	2.2	specific heat
I:	2.2	the number of wafer elements
J:	2.2	the number of lamp zones
K^{cond}:	2.2	tri-diagonal matrix used to describe conductive heat transfer
K^{conv}:	2.2	diagonal matrix used to describe convective heat transfer.
K^{rad}:	2.2	diagonal matrix used to describe radiative heat transfer.
L:	2.2	matrix of $L_{i,i}$
$L_{i,j}$:	2.2	the coefficient for radiative transfer from lamp zone j to wafer element i
N:	2.2	the number of sensors
P:	2.2	lamp power vector
P_j:	2.2	the lamp power from lamp zone j
P^{dyn}:	3.4	lamp power vector which adds enough additional heat to obtain a rate of temperature increase

P^{nom}:	2.4	nominal lamp power setting vector
P^{FB}:	3.3	feedback lamp power calculated by a controller
P^{OL}:	3.3	optimal open-loop lamp power
P^{ss}:	3.4	lamp power vector which balances the loose of the heat from the wafer by radiation and convection
P^{unif}:	3.4	lamp power vector used to calculate P^{dyn}
Q:	3.7.1	state weighing matrix
R:	3.7.1	input weighing matrix
T:	2.2	wafer temperature vector
T_i:	2.2	the temperature at center position of wafer element i
T^{des}:	3.1	desired temperature vector
T_{GAS}:	2.2	gas temperature vector
T_{gas}:	2.2	gas temperature
T^{nom}:	2.4	nominal temperature vector
T^{S}:	2.4	sensor temperature vector
T^{ss}:	2.3	steady-state temperature vector
T^{ss}_{ind}:	2.3	initial guess steady-state temperature vector
V:	3.7.1	sensor noise covariance weighting matrix
W:	3.7.1	process noise covariance weighting matrix

Z:	2.2	wafer thickness
h:	3.2	sampling period
h_i:	2.2	the area-weighted average convective heat transfer coefficient over the exposed surface of element i
$k_{i,i-1}$:	2.2	the thermal conductivity at the inner boundary of wafer element i
$k_{i,i+1}$:	2.2	the thermal conductivity at the outer boundary of wafer element i
m_i:	2.2	mass of wafer element i
q:	2.2	net heat flow vector
q_i:	2.2	the net heat flow into wafer element i
q_i^{rad}:	2.2	the sum of radiation emitted and absorbed by the wafer element i
q_i^{conv}:	2.2	the sum of thermal conductivity at the inner and outer boundaries of wafer element i
q_i^{cond}:	2.2	heat transfer between the wafer element i and the surrounding gas
r_i^{out}:	2.2	outer radius of wafer element i
r_i^{in}:	2.2	inner radius of wafer element i
r_i^{cent}:	2.2	center radius of wafer element i
ε_i:	2.2	the average total emissivity over exposed surface of wafer element i

ρ:	2.2	the mass density for the wafer
σ:	2.2	the Stefan Boltzmann constant
τ:	2.5	the time delay coefficient for lag factor
τ_{PI} :	3.6.2	reset time for PI feedback controller
$0_{i,j}$:	2.6.1	*i*-by-*j* zero matrix
$1_{i,j}$:	3.6.1	*i*-by-*j* matrix of ones
I_i:	2.2	*i*-by-*i* identity matrix
(A, B, C, D) :	2.6.1	state-space realization to represent transfer function for a linear system

Chapter 1 Introduction

The economics of computer control has been changed drastically by the microprocessor in the reduction of cost and the improvement in reliability. This has meant that computer-based systems are the first choice in many applications. The major costs of computer control are now no longer the computer hardware, but the system design costs and the cost of software. As a consequence, attention is shifting towards greater standardization of design of software products, and to the development of improved techniques for design and for software construction and testing.

The topic of this thesis lies in the area of the design and implementation of a real-time computer control system for a silicon wafer in rapid thermal processing (RTP). In this chapter, we will briefly introduce:

- the development of modern control theory which provides the possibility of designing real-time computer control systems
- the history of real-time computer control
- what is rapid thermal processing
- what to control in the wafer RTP
- the framework of the design and the implementation of real-time computer control for the wafer RTP
- problems to solve in this thesis
- outline of the thesis

1.1 Modern Control Theory

The complexity of mathematical models of modern industrial dynamic behavior (time-varying, nonlinear, multiple-input and multiple-output etc.), has meant that the classical control theory could no longer handle the control problems.

In response to these increased demands, modern control theory has been developed since the early 60s. It is a new approach to the analysis and design of complex control systems, for the necessity of meeting increasingly stringent requirements on performance, the increase in system complexity, easy access to large-scale computers, etc. It has played vital roles in industry applications.

Compared to the classical control theory, the modern control theory has two characteristics:

- It uses "State-Space Theory," which is the base of the modern control theory. The state-space model is contrasted with transfer function, in that the former reveals the system state variables, which hide inside the system in the time-domain, while the latter is based on the input-output relations of the system in the frequency-domain. The state-space approach to system analysis is best suited for reducing the complexity of the mathematical expressions, as well as resorting to computers for most of the tedious computations necessary in the analysis. Modern control theory is applicable to multi-input and multi-output (MIMO) systems, nonlinear systems, and distribution systems which are extremely different and can not have classical control theory applied to them.

- Applying to the design with computer of implement in adaptive control, hybrid control and optimal control.

1.2 Real-Time Computer Control

Due to the dramatic development in digital computers and microelectronics, the approach to analysis, design, and implementation of control systems is changing drastically. The application of digital computers to industrial control began in the late 1950s. The first digital computer specifically used for process control was investigated in 1958 by a group of engineers from Thomson Ramo Woolridge (TRW) and Texaco. A computer-controlled system for the polymerization unit was designed based on the RW-300 computer and went on-line March 12, 1959. The system controlled 26 flows, 72 temperatures, 3 pressures, and 3 compositions [ÅsW84]. In the following thirty years, the computer control system used in the process industry was increased one million times. More important, however, is that the reliability improved ten times, and the cost reduced ten times. Recently, the development of computer software and high level languages made it possible to solve many of the on-line, complicated logic problems.

In many applications, a computer control system must sense the environment and directly influence it through action. Its communication devices are connected by physical devices to processes which are external to the computer. These external processes are operated in their own time scales. Such systems are subjected to the real-time constraints of the environments in which they operate. For example, an autonomous vehicle operating in the real world needs a control system that responds quickly enough to avoid collision with

obstacles or other vehicles. Real-time refers to systems in which the order of computation is determined by the passage of time or by events external to the computer; and the results of the particular calculation may depend upon the value of the variable 'time' at the instance of execution of the calculations or the time taken to execute the computation. The real-time system can be mainly divided into clock-based systems, sensor-based systems or interactive systems.

Real-time control has become indispensable to the operation of the astronomical telescopes, manufacturing, automobiles, vision systems, and other applications since the 70s [Bel95].

Early real-time systems were operated in relatively simple, well-characterized environments. Such "traditional" real-time systems had a set of repeated tasks with known execution times and arrival patterns. Researchers are now extending real-time systems to more complex applications. Systems must adapt to the operational modes of the physical process they monitor and control, because resource limitations make it impossible to schedule and guarantee all behaviors that might be needed in a particular domain. For example, a real-time system controlling avionics must deal with different tasks and execution characteristics during take-off, cruising, and landing stages of operation [MHADSP95]. Real-time systems research has focused on developing low-level operating system mechanisms to support predictable execution of traditional periodic control tasks that have only minor data dependencies.

A real-time digital control computer or controller computer can be thought of as a three-stage pipe: data acquisition from sensor, data processing to generate control/ display

commands, and output the results to actuators/display devices.

1.3 Rapid Thermal Processing

Rapid thermal processing (RTP) is a recent single-wafer processing technology which is used to perform thermal manufacturing operations involved in integrated circuit (IC) fabrication. It is a versatile approach suitable for several different processing functions, such as rapid thermal annealing (RTA), rapid thermal cleaning (RTC), rapid thermal chemical vapor deposition (RTCVD), rapid thermal oxidation (RTO), and rapid thermal Nitridation (RTN) [GRS91].

In current mainstream IC processing, heat treatment is achieved by means of large, hot-wall ovens in which wafers are processed simultaneously. Processing times for wafer batches range from tens of minutes to many hours for a single process step. Speed is limited by the large masses of the oven walls. In RTP, only the relatively small thermal mass of the wafer itself is heated to, and cooled from, the processing temperature. The walls of the reaction chamber are water-cooled and remain at room temperature. The windows are air-cooled or water-cooled. Consequently, process steps may need only ten seconds for completion.

A wide variety of arrangements for heating wafers in RTP systems with infrared and/or visible light have been designed and built [Nor92]. Figure 1.1 represents a diagram of a generic RTP system. The wafer is supported by three quartz pins which contact the wafer near its edge (omitted from figure) and heated by lamps or lamp array, which is separated from the chamber by a transparent window. Light passes from lamp array to the wafer through the upper window. The lower window can be used for remote measurement of wafer

temperature or for application of ultraviolet light to the wafer. Power requirements for the lamp array are typically from several to several tens of kW [Nor92].

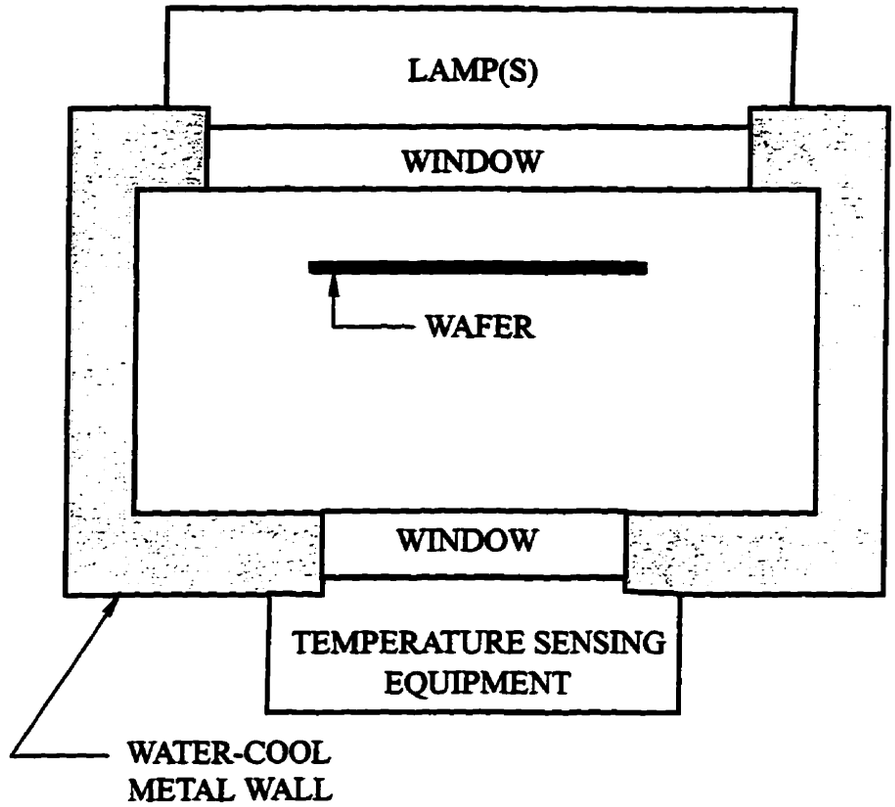


Figure 1.1 : Generic RTP system

The advantage for the RTP system is the lower thermal budget, better process repeatability, and shorter high temperature processing time for each wafer. However, since the wafer is very far from being in thermal equilibrium with its surroundings, the problem of temperature nonuniformity over the wafer is much more acute than in hot-wall oven processing.

1.4 Wafer Temperature Control in RTP

During the IC manufacturing, silicon wafers undergo a number of processing steps typically at high temperatures (Figure 1.2 shows a typical trajectory for wafer in RTP), under various atmospheric conditions, and high levels of heat power. However, the control techniques required to provide real-time monitoring and control with the desired accuracy are still under development.

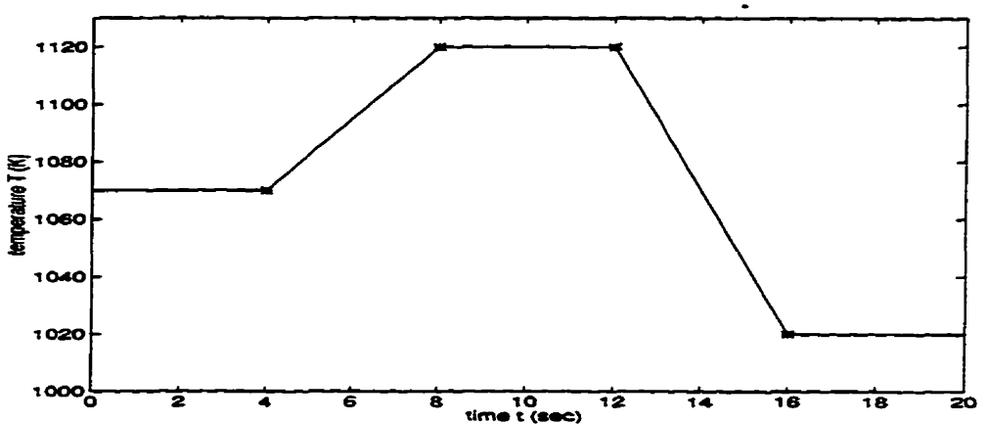


Figure 1.2 : A typical temperature trajectory for a wafer in RTP

One of the major problems is the control of the wafer temperature since it is required to maintain near uniformity temperature distribution over the wafer at all times, while following fast temperature trajectories. A lot of research has been done to solve the nonuniformity problem [GRS91], [CAKLL91], [Nor92], [ApS92], and [SMSK94]. In this thesis, temperature uniformity control is still one of the topics. Several control systems will be designed to compare the results.

Currently, wafer temperature control can be done through scalar control or multivariable control, depending on the system. Multivariable control has been proven to be much

better than scalar control in [Nor92] and [ApK92]. So, the system with multiple, independently controllable lamps and multiple sensors are used in this thesis. The actuator is shown in Figure 1.1.

It is necessary to measure the temperatures at various points across the wafer during RTP for multivariable control. The most popular approach is the use of pyrometers located outside the chamber. Recently, there have been some new techniques developed such as double-pass infrared transmission [CuS95] and multiwavelength imaging pyrometer [BMKM95]. However, the measurement of the wafer temperature problem is beyond this thesis and good measurement is assumed.

1.5 Design and Implementation

Modern control theory and computer technology allow us to solve many industry problems in real-time. In this thesis, we are going to design and implement real-time software applications to control the temperatures across the wafer in RTP. A wafer RTP emulator will be designed because in some research cases, the actual RTP system does not exist.

Figure 1.3 shows the structure of the system design. In this figure, the emulator computer will be used as an emulator to emulate the wafer RTP dynamics system. The controller computer will be used as a digital controller to set lamp powers. The outputs from the emulator computer are the digital values of real-time wafer temperatures measured by sensors, and these outputs will be passed to the controller computer through input and output (I/O) boards (which convert digital value to analog value (D/A) and analog value to digital value (A/D)). The outputs from the controller computer are the digital values of real-time lamp

powers calculated by the digital controller, and these outputs will be sent to the emulator computer through I/O boards.

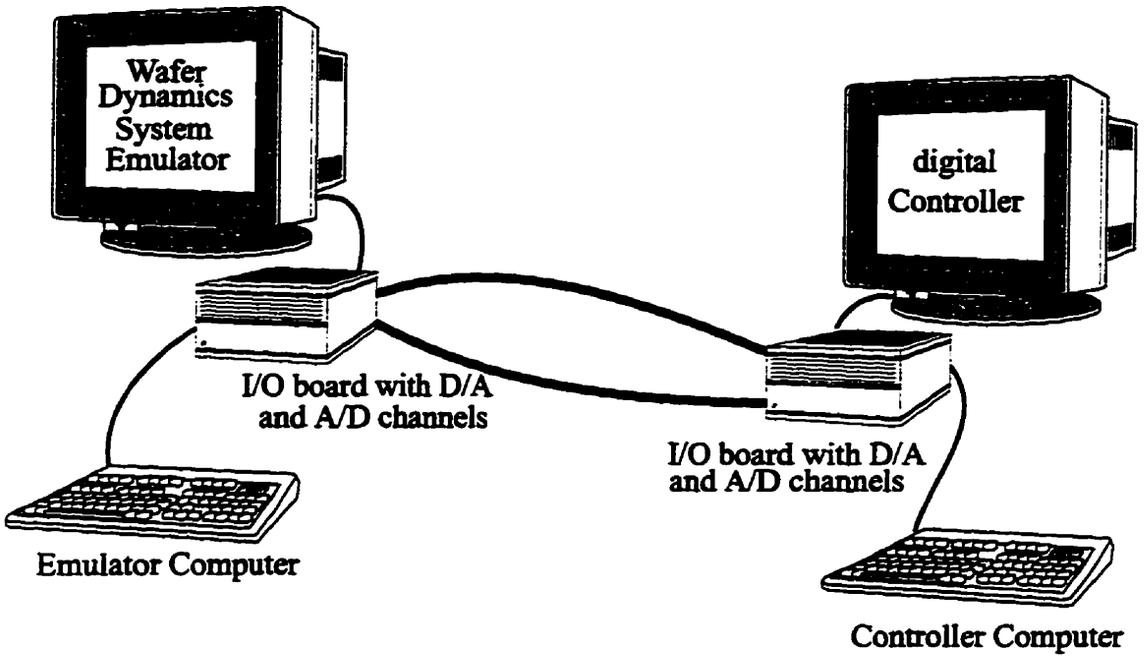


Figure 1.3 : Structure of the system design and implement of a wafer in RTP

The real-time software design of the above system will include two applications. One is for the emulator and the other is for the controller. These two applications will be separately implemented in two personal computers. The only connections between them are signals through I/O boards. This is because we can easily replace the wafer RTP emulator with an actual wafer RTP system without changing the controller application. Each application can be briefly described as four parts: retrieve user inputs and initialize the application; run real-time portion to read inputs, update outputs and send outputs.

The C programming language and Matlab software package will be used to implement the applications.

1.6 Thesis Outline

For industry processes, it is difficult to get an exact mathematical model for the process behavior. This causes difficulties in the design of control systems. Fortunately, in wafer RTP system, many jobs have been done in the model design [Lor88], [Nor92],[GRS91], [ApS92], and [CPKX94]. The model is functioning well, although it still has slight errors depending on the different processing environment. One of the purposes of this thesis is to design feedback controllers to correct the model errors.

Currently, finding a simple and effective way to control the temperature uniformity inside of the wafer is still a research problem. So, how to design the control systems and how to implement them in the real-time software application are the main problems that have to be resolved. Also, clear and convenient graphic user displays of the wafer RTP and control systems will help the process of manufacturing.

This thesis will address the above problems and its chapters are outlined below:

Chapter 2 presents the thermal models, which are represented as vector-matrix formula for the wafer nonlinear model and state-space formula for the wafer linear model.

Chapter 3 is concerned with the design of the linear MIMO temperature control system, which includes open-loop control and feedback control.

Chapter 4 covers the real-time software application designs, implementation study under Unix, suggestions of installing applications into separate PCs and design ideas of Graphic User Interfaces (GUIs) for the wafer RTP and controller systems.

Chapter 5 is about the simulation results of model test and controller functionality

test.

Chapter 6 contains some general discussions about the results in the thesis, and some suggestions for future research.

Appendix A provides a summary of symbols and notations used in the thesis.

Appendix B gives the user guide about how to use the software applications (within the Unix environment) designed in Chapter 4.

Chapter 2 Thermal Models for an RTP System

In this chapter, we discuss nonlinear and linear models used to calculate the temperatures across a wafer as functions of lamp powers, and other associated parameters inside a Rapid Thermal Processor (RTP). The linear model is obtained from the nonlinear model by introducing nominal wafer temperatures and corresponding nominal lamp powers. Each of these models contain two parts: the ideal RTP system generated from thermal dynamic equations, and the lag factor added to slow down the lamp power changes. The nonlinear model is used for simulations and is represented in vector-matrix form. The linear model is used for design of the controllers and is represented in state-space form. The calculation of steady-state temperature is also introduced in this chapter.

2.1 Introduction of RTP System

A wide variety of arrangements for heating single wafer in RTP systems have been designed and built in industry [Nor92].

Generally, the wafer is supported by three quartz pins (omitted from Figure 1.1) which contact the wafer near its edge. The chamber walls of the system are water-cooled and the windows are air or water-cooled. Ideally, all of the inner surfaces of the reaction chambers remain at room temperature during the wafer processing. The wafer can be heated

from one or both sides. The heat source is from a powerful lamp or lamp array (from several to several tens of kW), which is visible, and infrared radiation. The arrangement of the lamps is variable (it can be in concentric rings, in a hexagonal packing or other ways). Temperature measurement can be at one or many points on the wafer during the RTP. The most popular temperature measurement technique is to put one or more pyrometers outside the chamber close to the bottom window (for single side heating systems).

In this thesis, a system called "S", built in [Nor92], is adopted and the simplified schematic is shown in Figure 2.1. (The lamp array is idealized to be black-body lamp zones set in the ceiling of the chambers; the quartz pins, gas inlets and outlets, doors or temperature sensing equipment, and other necessary components are omitted from the diagram).

In the S system, the wafer is heated from the top side only. The upper window allows light to pass to the wafer, and the bottom window is used for remote measurement of wafer temperatures at different points. (Several sensors are located outside of the chamber). The lamp array is organized in three circularly symmetric zones. The system is a cylindrical coordinate system where the origin of the coordinate system is the center of the wafer bottom surface, and the z-axis of the coordinate system coincides with the central axis of the wafer. The interior of the chamber is axisymmetric and is coaxial with the wafer.

There are some assumptions made for the above RTP system. The models obtained in the later sections are based on these assumptions:

- the effect of the quartz pins supporting the wafer is neglected
- the wafer is perfectly cylindrical
- gas inlet for the chamber is a "shower-head" symmetric about the wafer axis to

avoid non-axisymmetric effect on the wafer temperature distribution, and other non-axisymmetry effects are avoided or neglected as well.

- the temperature distribution is axisymmetric and the wafer is thin enough that axial (z-axis) thermal gradients can be neglected.
- furthermore, the wafer will be broken into annular zones. In each of the annular zones, the temperature is assumed to be uniform (such an approach is often used in radiative heat transfer applications and has been used for the RTP system in Lord [Lor88]).

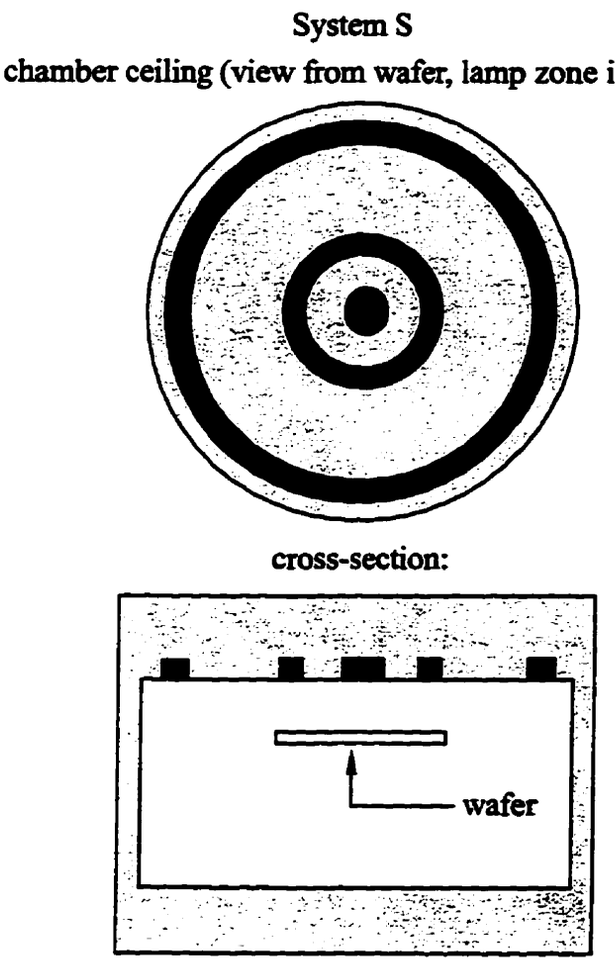


Figure 2.1 : Schematic diagram for system S

2.2 Modeling of Ideal RTP System

The first step in the analysis of a dynamic system is to derive its model. The RTP system can be analyzed through modeling and simulation of the thermal environment. The main goal of any theoretical modeling of a RTP system is to predict the wafer temperature as a function of time and position on the wafer. There are three different ways heat can flow from one substance to another: conduction, convection and radiation.

The approach used here is to break the wafer into annular zones (which is adopted from [Nor92]) as shown in Figure 2.2. The z-axis is strongly enlarged for clear representation.

Break the wafer with radius R and thickness Z into I concentric elements. Let r_i^{out} be the outer radius of element i, let r_i^{in} be the inner radius of element i, and let r_i^{cent} be the center radius of element i.

$$0 < r_1^{out} < r_2^{out} < \ldots < r_I^{out} = R \tag{2.1}$$

$$r_i^{in} = r_{i-1}^{out} \qquad i > 1 \text{ and } i \le I \tag{2.2}$$

$$\left(r_i^{cent}\right)^2 = \frac{\left(r_i^{in}\right)^2 + \left(r_i^{out}\right)^2}{2} \tag{2.3}$$

Then for element i the mass m_i is given by:

$$m_i = \rho\pi\left(\left(r_i^{out}\right)^2 - \left(r_i^{in}\right)^2\right)Z \tag{2.4}$$

where ρ is the mass density, and the exposed surface area A_i of element i is given by:

$$A_i = 2\pi\left(\left(r_i^{out}\right)^2 - \left(r_i^{in}\right)^2\right) \text{ for } i = 1, \ldots, I-1 \tag{2.5}$$

and

$$A_I = 2\pi\left(\left(r_I^{out}\right)^2 - \left(r_I^{in}\right)^2\right) + 2\pi RZ \quad \text{for } i = I \tag{2.6}$$

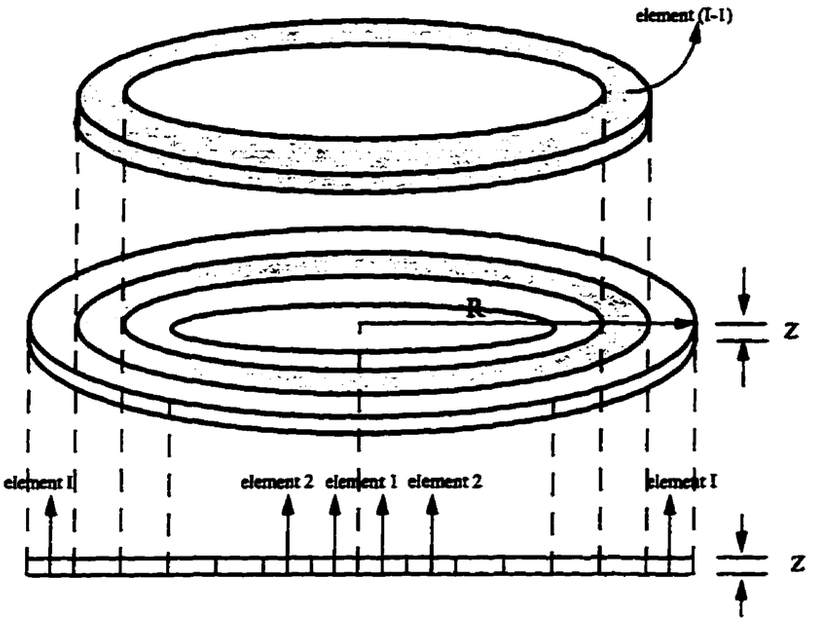


Figure 2.2 : Wafer is broken into annular zones

The net heat flow q_i into each wafer element i is given by:

$$q_i = q_i^{rad} + q_i^{cond} + q_i^{conv} \tag{2.7}$$

where q_i^{rad} is the sum of radiation emitted by the wafer element i (defined as q_i^{em}) and radiation absorbed by the wafer element i (defined as q_i^{ab}), where q_i^{cond} is the sum of thermal conductivity at the inner boundary of element i and at the outer boundary of element i, and where q_i^{conv} is the heat transfer between the wafer element i and the surrounding gas.

The temperature at each element "center" position r_i^{cent} is defined as T_i, and the time derivative of T_i is given by

$$\dot{T}_i = \frac{q_i}{m_i C_p(T_i)} \tag{2.8}$$

where C_p is the specific heat of the wafer and is strongly dependent on temperature. (Note in continuous time, "dot" denotes time derivative.)

More details on how to calculate the heat flow into wafer element i are presented below.

Let us define J as the number of total lamp zones, P_j as the lamp power from lamp zone j, and T_{gas} as the input gas temperature. We assume the relation between power emitted by a source, and power absorbed by a wafer element is linear. Define $L_{i,j}$ as the fraction (value is from one to zero) for radiative transfer from lamp zone j to wafer element i. For example, assuming $L_{i,j} = 0.025$, and lamp zone j is emitting a total of 1kW, the wafer element i will absorb 25W of that 1kW. Then, when the same lamp zone is emitting a total of 2kW, the same wafer element absorbs 50W from the 2kW. The value of $L_{i,j}$ is strongly dependent on lamp zone positions. In the same way, we define $W_{i,i'}^e$ as the fraction for radiative transfer from wafer element i' to wafer element i.

Therefore, radiative heat flow q_i^{rad} is given by:

$$q_i^{rad} = q_i^{em} + q_i^{ab} \tag{2.9}$$

where

$$q_i^{em} = -\varepsilon_i \sigma A_i T_i^4, \tag{2.10}$$

and

$$q_i^{ab} = \sum_{j=1}^{J} L_{i,j} P_j + \sum_{i'=1}^{I} W_{i,i'}^{e} q_{i'}^{em}. \tag{2.11}$$

From Equation (2.10), ε_i is the average total emissivity over the exposed surface of the element *i*. A black-body surface has an emissivity of exactly one; the emissivity of a gray surface is between zero and one. σ is the Stefan-Boltzmann constant.

Conductive heat flow q_i^{cond} is given by:

$$q_i^{cond} = -2k_{i,i-1}\frac{T_i - T_{i-1}}{r_i^{cent} - r_{i-1}^{cent}}\pi r_i^{in} Z - 2k_{i,i+1}\frac{T_i - T_{i+1}}{r_{i+1}^{cent} - r_i^{cent}}\pi r_i^{out} Z \tag{2.12}$$

where $k_{i,i-1}$ is the thermal conductivity at the inner boundary of element *i*, and $k_{i,i+1}$ is the thermal conductivity at the outer boundary of element *i*. If element *i* is the innermost or outermost element, the conduction occurs at only one boundary.

Convective heat flow q_i^{conv} is given by:

$$q_i^{conv} = -h_i A_i (T_i - T_{gas}) \tag{2.13}$$

where h_i is the area-weighted average convective heat transfer coefficient over the exposed surface of element *i*. The value of h_i depends on the pressure and gas flow characteristics, and often needs to be experimentally determined.

Let T, T^n, q, T_{GAS}, and P denote vectors as given by

$$T = \begin{bmatrix} T_1 \\ \vdots \\ T_I \end{bmatrix}, \quad T^n = \begin{bmatrix} T_1^n \\ \vdots \\ T_I^n \end{bmatrix}, \quad q = \begin{bmatrix} q_1 \\ \vdots \\ q_I \end{bmatrix}, \quad T_{GAS} = \begin{bmatrix} T_{gas} \\ \vdots \\ T_{gas} \end{bmatrix}, \text{ and } \quad P = \begin{bmatrix} P_1 \\ \vdots \\ P_J \end{bmatrix} \tag{2.14}$$

where T is the temperature I-vector of the wafer elements, n is any integer, q is the net heat flow into the wafer elements ($q \in \Re^{I \times 1}$), T_{GAS} is the gas temperature I-vector for the wafer elements, and P is the lamp power J-vector of lamp zones.

Let us also define some matrices as:

$diag(d_1, \ldots d_n)$: diagonal matrix as $\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$

I_n: n-by-n identity matrix.

C: diagonal matrix containing heat capacities of wafer element where $C = diag(m_1 C_p(T_1), m_2 C_p(T_2), \ldots, m_I C_p(T_I))$

C^{-1}: the inverse of the square matrix C.

$L \in \Re^{I \times J}$: matrix of $L_{i,j}$

$W^e \in \Re^{I \times I}$: matrix of $W^e_{i,i}$

$K^{rad} \in \Re^{I \times I}$: diagonal matrix, used to describe radiative heat transfer, where

$$K^{rad} = -\left(I_I + W^e\right) diag\left(\varepsilon_1 \sigma A_1 T_1^4, \varepsilon_i \sigma A_2 T_2^4, \ldots, \varepsilon_I \sigma A_I T_I^4\right)$$

$K^{conv} \in \Re^{I \times I}$: diagonal matrix, used to describe convective heat transfer, where

$$K_{conv} = diag(h_1 A_1, h_2 A_2, \ldots, h_I A_I)$$

$K^{cond} \in \Re^{I \times I}$: tri-diagonal matrix used to describe conductive heat transfer,

$$\text{where } K^{cond}_{i,i} = -\left(\frac{2\pi r_i^{in} Zk_{i,i-1}}{r_i^{cent} - r_{i-1}^{cent}} + \frac{2\pi r_i^{out} Zk_{i,i+1}}{r_{i+1}^{cent} - r_i^{cent}} \right)$$

$$\text{if } 2 \le i \le I-1,$$

$$K^{cond}_{1,1} = \frac{2\pi r_1^{out} Zk_{1,2}}{r_2^{cent} - r_1^{cent}} \quad \text{if } i = 1,$$

$$K^{cond}_{I,I} = \frac{2\pi r_1^{out} Zk_{1,2}}{r_I^{cent} - r_{I-1}^{cent}} \quad \text{if } i = I,$$

$$K^{cond}_{i-1,i} = \frac{2\pi r_i^{in} Zk_{i,i-1}}{r_i^{cent} - r_{i-1}^{cent}} \quad \text{if } 2 \le i \le I,$$

$$K^{cond}_{i+1,i} = \frac{2\pi r_i^{out} Zk_{i,i+1}}{r_{i+1}^{cent} - r_i^{cent}} \quad \text{if } 1 \le i \le I-1, \text{ and}$$

$$K^{cond}_{i,i'} = 0 \quad \text{if } |i-i'| > 1$$

Then the ideal wafer RTP model can be written in vector-matrix form as

$$\dot{T} = C^{-1} q \tag{2.15}$$

and

$$q = K^{rad} T^4 + K^{cond} T + K^{conv} (T - T_{GAS}) + LP \tag{2.16}$$

The radiation transfer from the chamber walls is small and is neglected since the wall is water cooled and the wall temperatures are close to room temperature. Also the capacitive effects of a thick window are neglected. Since the associated time constant of the window is two orders of magnitude slower than those of the wafer and the lamps, the window heating

is considered as a slowly varying disturbance [CPKX94].

In the real system, only few sensors are available, so not all the wafer element temperatures are measured. We define N as the total number of sensors. Let N-by-I matrix C_y be the matrix formed from the identity matrix by selecting rows corresponding to elements whose temperatures are being measured. Therefore, the nonlinear ideal wafer RTP model is given as

$$\dot{T} = C_y C^{-1} q \tag{2.17}$$

2.3 Steady-State Temperature

Computing steady-state temperature is quite important during the process, since it will be used both for controller designs and initial wafer temperatures in simulation.

Suppose if there is a given lamp power P, we wish to know the corresponding steady-state temperature profile $T^{ss}(P)$. A nonlinear equation relating P and T^{ss} is given by Equation (2.16). Since the derivative of steady-state temperature is zero ($\dot{T} = 0$), the heat flow q should be equal to zero too (from Equation (2.15)). Starting with an initial guess, we could use Newton's method to converge iteratively to the solution.

Let us start with the initial guess temperature, T^{ss}_{ind}, where *ind* is an iteration count. Submit this initial guess temperature into Equation (2.16), the heat flow q_{ind} is given by:

$$q_{ind} = K^{rad} T^{ss\ 4}_{ind} + K^{cond}\left(T^{ss}_{ind}\right) T^{ss}_{ind} + K^{conv}\left(T^{ss}_{ind} - T_{GAS}\right) + LP. \tag{2.18}$$

Using the fact, we could have:

$$q - q_{ind} = \frac{dq}{dT_{ind}^{ss}}\left(T_{ind+1}^{ss} - T_{ind}^{ss}\right), \tag{2.19}$$

where $\frac{dq}{dT_{ind}^{ss}}$ is a *I*-by-*I* matrix of partial derivatives. (Note that it is not exactly Newton's method because a term involving the derivative of $K^{cond}(T)$ is neglected, see [Nor92] for more details).

Set $q = 0$ and obtain,

$$0 - q_{ind} = \frac{dq}{dT_{ind}^{ss}}\left(T_{ind+1}^{ss} - T_{ind}^{ss}\right). \tag{2.20}$$

If $\frac{d}{dT_{ind}^{ss}} q \neq 0$, we can solve Equation (2.20) for T_{ind+1}^{ss} as

$$T_{ind+1}^{ss} = T_{ind}^{ss} - \left(\frac{dq}{dT_{ind}^{ss}}\right)^{-1} q_{ind}, \tag{2.21}$$

where

$$\frac{dq}{dT_{ind}^{ss}} = 4K^{rad} diag\left(\left(T_{ind}^{ss}\right)^3\right) + K^{cond}\left(T_{ind}^{ss}\right) + K^{conv}, \tag{2.22}$$

therefore, we can have,

$$T_{ind+1}^{ss} = T_{ind}^{ss} - \left\{4K^{rad} diag\left(\left(T_{ind}^{ss}\right)^3\right) + K^{cond}\left(T_{ind}^{ss}\right) + K^{conv}\right\}^{-1} q_{ind}. \tag{2.23}$$

Next we repeat this procedure with T_{ind}^{ss} replaced by T_{ind+1}^{ss} and keep repeating the above process. In general, if the *n*th approximation is T_{ind+n}^{ss} and if $\frac{d}{dT_{ind+n}^{ss}} q \neq 0$, then the next approximation is given by

$$T^{ss}_{ind+n+1} = T^{ss}_{ind+n} - \left(\frac{dq}{dT^{ss}_{ind+n}}\right)^{-1} q_{ind}, \tag{2.24}$$

if T^{ss}_{ind+n} becomes closer and closer to T^{ss} as n becomes large, then

$$\lim_{n\to\infty} T^{ss}_{ind+n} = T^{ss}. \tag{2.25}$$

2.4 Linearization

Many control laws are developed based on linear system models. The designs of two controllers in Chapter 3 are based on a linear model which is obtained from a nonlinear ideal RTP system.

A linearized model for an ideal RTP system of Equation (2.15) is given by Norman [Nor92]. This system is a linearization about a nominal temperature profile T^{nom} ($T^{nom} \in \Re^{I\times 1}$) and a nominal lamp setting vector P^{nom} ($P^{nom} \in \Re^{J\times 1}$).

Figure 2.3 shows the linearization model for an ideal RTP system, where the outputs of the system are only the sensor temperatures.

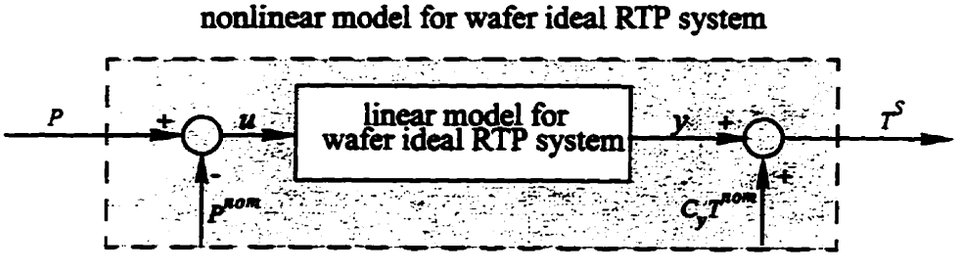


Figure 2.3 : Linearization model for ideal RTP system

In Figure 2.3, u is the J-vector of deviations from P^{nom}, and it can be represented as

$$u = P - P^{nom}. \tag{2.26}$$

The states of the system x ($x \in \Re^{I\times 1}$) is the vector of deviations from T^{nom}, and it is,

$$x = T - T^{nom}. \tag{2.27}$$

The sensed output y is a N-vector of measurements of temperature errors at some wafer elements, $C_y T^{nom}$ is the corresponding N-vector of nominal temperatures at these wafer elements, and T^S ($T^S \in \Re^{N\times 1}$) is the sensed wafer temperature vector at these elements.

A state equation describing the linearized system is

$$\dot{x} = Ax + B_u u \tag{2.28}$$

where A is I-by-I matrix,

$$A = C^{-1}\left(4K^{rad}\left(T^{nom}\right)^3 + K^{cond}\left(T^{nom}\right) + K^{conv}\right); \tag{2.29}$$

B_u is I-by-J matrix,

$$B_u = C^{-1}L. \tag{2.30}$$

(recall L is the matrix of fraction for radiative transfer from lamp to wafer element.) More details about L can be found in Norman [Nor92].

The output equations corresponding to the linear state Equation (2.28) is

$$y = C_y x. \tag{2.31}$$

2.5 Lag Factor

In the physical RTP system, it takes time to change the lamp filament temperatures. To faithfully represent this situation, one can add a lag factor in the ideal RTP system. The RTP system is then described by a lagged model.

Since we use a linear model for controller designs and a nonlinear model for the simulations, we need to consider different ways to add the lag. For the linear model, the lag factor is added to the deviation lamp power u and is shown at the top model of the Figure 2.4, where J-vector u^{lf} denotes the outputs from the lag fact. For the nonlinear model, the comparison of the lamp power P and the nominal lamp power P^{nom} is unnecessary, so the lag factor is put directly to P and is shown at the bottom model of the Figure 2.4, where J-vector P^{lf} denotes the outputs from the lag factor in the nonlinear model.

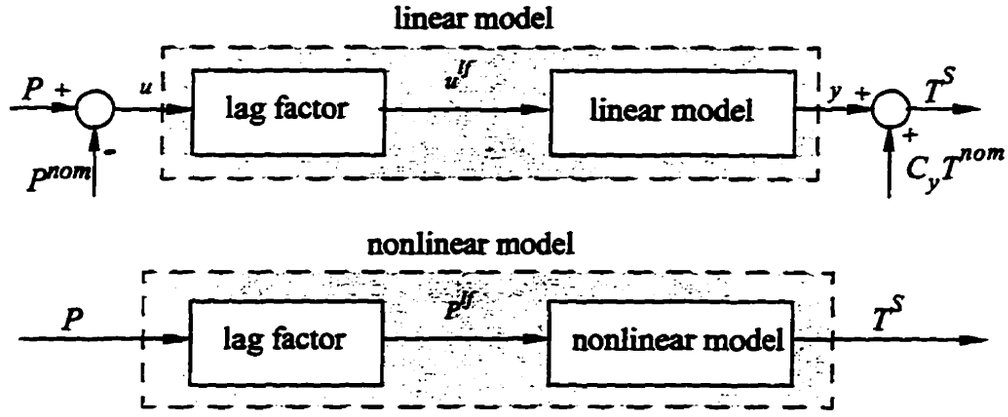


Figure 2.4 : Block diagrams of the wafer thermal models

To represent the effect of the lag, we use inertia factor,

$$\frac{1}{\tau s + 1}, \tag{2.32}$$

where τ is the time delay coefficient. A more detailed representation of the lag factor in Figure 2.4 is then given in Figure 2.5, where we use the linear model as the example, and each subsystem is a single-input and single-output (SISO) system. (u_i denotes the element i of vector u and u_i^{lf} denotes the element i of vector u^{lf}).

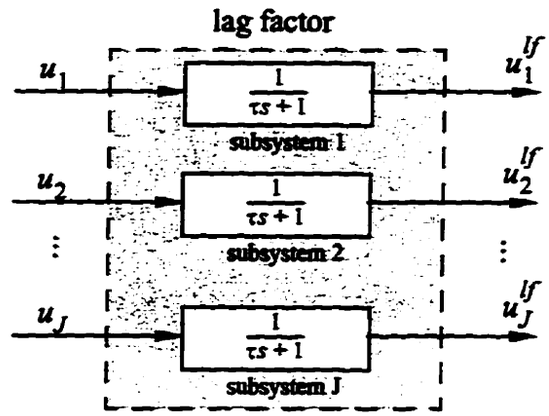


Figure 2.5 : Diagram of lag block

To see the effect of the inertia factor, first set a unit step input u_i, the output u_i^{lf} as the function of t, is given by:

$$u_i^{lf} = \left(1 - \frac{1}{\tau} e^{-\frac{1}{\tau}t}\right). \tag{2.33}$$

The step responses with several different values of τ are shown in Figure 2.6.

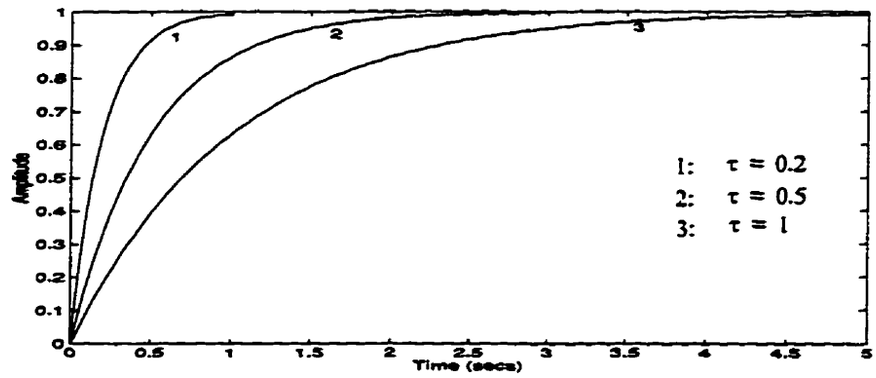


Figure 2.6 : Step responses with different values of τ

Then set ramp function input u_i ($u_i = t$ for $t \geq 0$, and $u_i = 0$ for $t < 0$), the output u_i^{lf}, is given by

$$u_i^{lf} = t + \tau e^{-\frac{1}{\tau}t} - \tau. \tag{2.34}$$

The various trajectories of u_i^{lf} are shown in Figure 2.7 based on the different values of τ

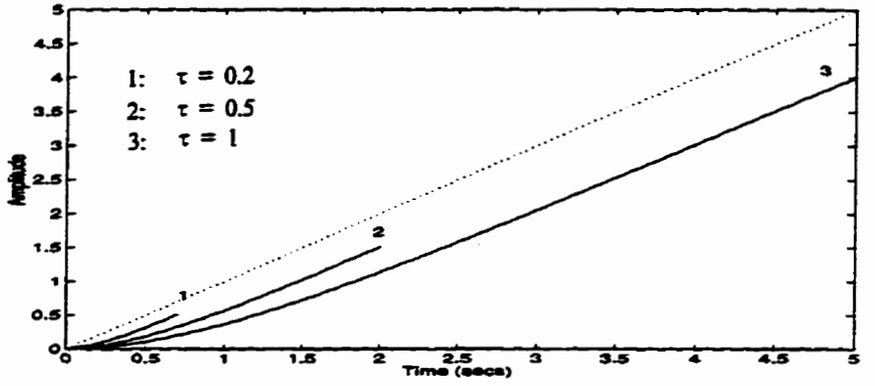


Figure 2.7 : Trajectories of u_i^{lf} with different values of τ

2.6 Lagged Models of the RTP System

In this section, we introduce two lagged models for the wafer RTP system. The first is the lagged linear model and the second is the lagged nonlinear model. Each model includes the ideal RTP system and lag factor.

2.6.1 Linear Model of the RTP System

The state-space model for the ideal linear wafer RTP system is represented by Equation (2.28) and Equation (2.31). If the lag factor is added, we have to replace the vector signal u with u^{lf} in Equation (2.28).

For the lag factor, we could obtain the state-space model from analyzing the single SISO subsystem. From the transfer function (Equation (2.32)), we could have

$$\dot{u}_i^{lf} = -\frac{1}{\tau}u_i^{lf} + \frac{1}{\tau}u_i. \tag{2.35}$$

Define states x_i^{lamp}, where $x_i^{lamp} = u_i^{lf}$. We can obtain the state-space equations for the SISO subsystem as

$$\dot{x}_i^{lamp} = a_i x_i^{lamp} + b_i u_i, \tag{2.36}$$

$$u_i^{lf} = c_i x_i^{lamp}, \tag{2.37}$$

where $a_i = -\frac{1}{\tau}$, $b_i = \frac{1}{\tau}$, and $c_i = 1$.

By combing all the subsystems in Figure 2.5, we can obtain the state-space equations for the lag factor,

$$\dot{x}^{lamp} = A_{add}x^{lamp} + B_{add}P. \tag{2.38}$$

$$u^{lf} = C_{add}x^{lamp}. \tag{2.39}$$

where

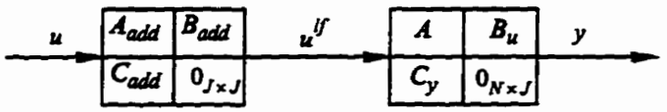
$$x^{lamp} = \begin{bmatrix} x_1^{lamp} \\ \vdots \\ x_J^{lamp} \end{bmatrix}, \quad A_{add} = diag(a_1, a_2, ..., a_J), \quad B_{add} = diag(b_1, b_2, ..., b_J),$$

$$u^{lf} = \begin{bmatrix} u_1^{lf} \\ \vdots \\ u_J^{lf} \end{bmatrix} \text{ and } C_{add} = diag(c_1, c_2, ..., c_J).$$

To simplify presentation for a state-space realization (A,B,C,D), we use the packed notation, $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$, to represent its transfer matrix $D + C(sI - A)^{-1}B$ in continuous time or

$D + C(zI - A)^{-1}B$ in discrete time.

Let $0_{i \times j}$ be a i-by-j zero matrix (i and j are any positive integers). The linear model in Figure 2.4 can be represented as follows:



From Equation (2.28) and Equation (2.39), we can have

$$\dot{x} = Ax + B_u C_{add} x^{lamp}. \tag{2.40}$$

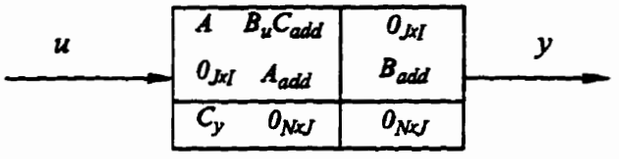
By combining Equation (2.40) and Equation (2.38), we can obtain the linear state equation as

$$\begin{bmatrix} \dot{x} \\ \dot{x}^{lamp} \end{bmatrix} = \begin{bmatrix} A & B_u C_{add} \\ 0_{J \times I} & A_{add} \end{bmatrix} \begin{bmatrix} x \\ x^{lamp} \end{bmatrix} + \begin{bmatrix} 0_{I \times J} \\ B_{add} \end{bmatrix} u \tag{2.41}$$

and the output equation with the combined state vector $\begin{bmatrix} x \\ x^{lamp} \end{bmatrix}$ as

$$y = \begin{bmatrix} C_y & 0_{N \times J} \end{bmatrix} \begin{bmatrix} x \\ x^{lamp} \end{bmatrix}. \tag{2.42}$$

Then, the RTP system can be shown in one block as



2.6.2 Nonlinear Model of the RTP System

The vector-matrix form of a nonlinear ideal wafer RTP system is represented by

Equation (2.16) and Equation (2.17). If the lag factor is added, we have to replace the vector signal P with P^{lf} in Equation (2.16).

Let us define p^{lamp} as the state vector, where $p^{lamp} = P^{lf}$. From Equation (2.38) and Equation (2.39), we can have

$$\dot{p}^{lamp} = A_{add}p^{lamp} + B_{add}P, \tag{2.43}$$

$$P^{lf} = C_{add}p^{lamp}. \tag{2.44}$$

From Equation (2.16), and Equation (2.17), we can have

$$\dot{T} = C_y C^{-1}\left(K^{rad}T^4 + K^{cond}T + K^{conv}(T - T_{GAS}) + LP^{lf}\right). \tag{2.45}$$

By substituting Equation (2.44) into Equation (2.45), and combining that equation with Equation (2.43), we can obtain the vector-matrix equation for the nonlinear wafer RTP system as

$$\begin{bmatrix} \dot{p}^{lamp} \\ \dot{T} \end{bmatrix} = \begin{bmatrix} A_{add}p^{lamp} + B_{add}P \\ C_y C^{-1}\left(K^{rad}T^4 + K^{cond}T + K^{conv}(T - T_{GAS}) + LC_{add}P^{lamp}\right) \end{bmatrix}. \tag{2.46}$$

Once mathematical models of the RTP system are obtained, the analysis can be continued. In the next chapter, we will focus on the design of the discrete-time controllers.

Chapter 3 Real-Time Control System

In this chapter, the designs of several discrete-time feedback controllers are presented. The plant model used is discretized as the wafer linear model given by Equation (2.41) and Equation (2.42).

3.1 Control Strategy

A block diagram of the basic real-time control system used in this thesis is shown in Figure 3.1, where T^{des} is a scalar signal. (Recall P is a lamp power J-vector, T^S is a sensor temperature N-vector, J is the number of lamp zones and N is the number of sensors.)

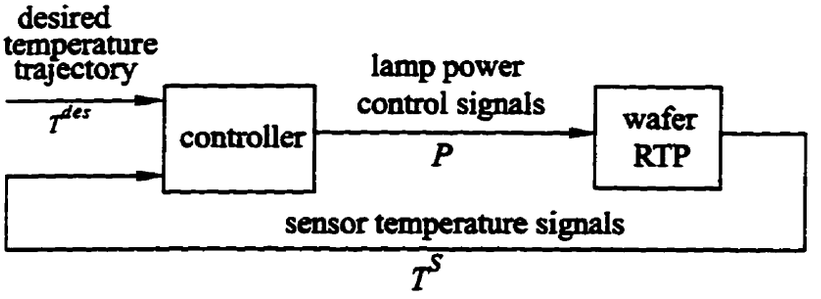


Figure 3.1 : Block diagram of the real-time wafer RTP and control systems

A real time multivariable strategy is used to control the uniformity and repeatability of wafer temperature in RTP. This strategy is based on a physical model of the process where the model parameters are estimated using an experimental design procedure. The control law design method is used to automatically compute the lamp powers to a multi-zone array of concentric heating zones to make wafer temperatures follow the desired tra-

jectories and achieve wafer temperature uniformity. Control actions are made in response to real-time feedback information provided by temperature sensing at multiple points across the wafer and the pre-specified desired temperature trajectory.

Control design on a particular system involves many choices, like:

- sensors and their placement
- actuators and their placement
- control law.

All of these are important in development of a complete control system, but in this chapter we focus on the control law used to achieve spatial temperature uniformity. The controllers we design under the control law should:

- be able to make the temperatures at all points on the wafer closely follow desired temperature trajectories that are entirely specified before the wafer process steps begin;
- be able to ensure such trajectory-following despite uncertainty about many system variables such as wafer dimensions and surface conditions, the exact temperatures of components of the reaction chamber, chamber gas composition and its effect on convective cooling, and so on;
- guarantee satisfactory trajectory-following for a wide range of desired trajectories.

3.2 Discretization

The lamp power settings in the system are controlled by a digital computer, which

takes samples of the temperatures at discrete time instants. Suppose the sampler is ideal, which is represented as *S*. The lamp power signals will most likely be zero-order-hold (which is represented as *H*) signals, that is, signals which are constant between sample instants. We use a broken line to represent the discrete-time system to distinguish it from the continuous-time system. We use the notation $\cdot(k)$ to represent the sampled signals during time $kh \le t < (k+1)h$, where h is the sampling period and k can be taken as any nonnegative integer value. Therefore, the controller in Figure 3.1 is represented in Figure 3.2.

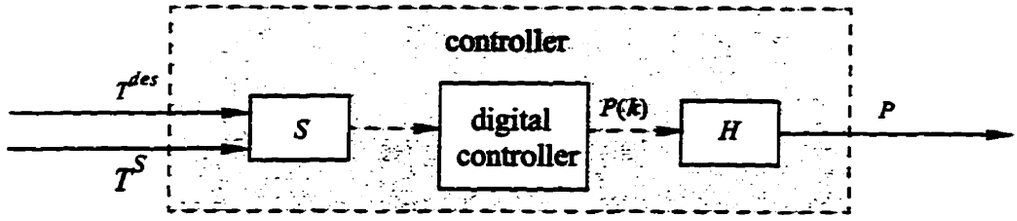


Figure 3.2 : Block diagram of the digital controller

Since we are going to design digital controllers, it makes sense to transform continuous-time wafer model into discrete-time model, which has exactly the same behavior as the continuous-time system at the sampling instants. The approach here is to put zero-order-hold at the inputs and ideal sampler at the outputs of the wafer RTP (Figure 3.3).

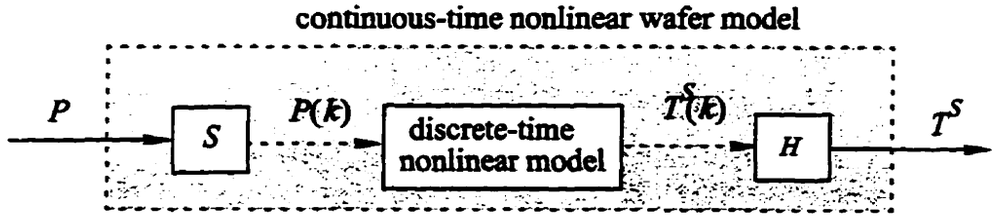


Figure 3.3 : Discrete-time nonlinear wafer model

Two controllers designed in Section 3.7 and Section 3.8 are discrete-time controllers

based on the discrete-time linear model. Therefore, we also need a discrete-time linear model for design purposes (Figure 3.4).

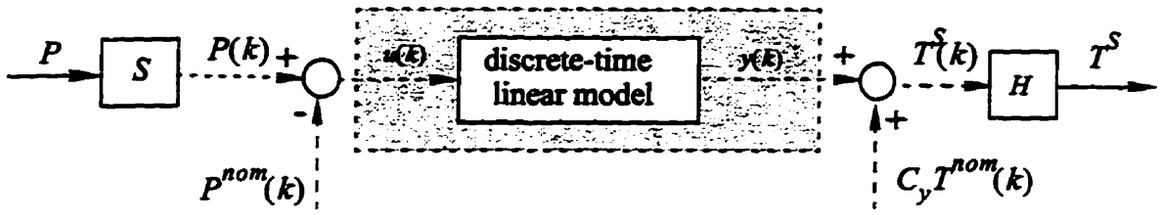


Figure 3.4 : Discrete-time linear wafer model

Several ways are available to discretize a continuous-time system. c2d (this is a Matlab function) is used here to map the state-space data for the wafer RTP in Equation (2.41) and Equation (2.42) to discrete-time state-space data.

To simplify the representation, let us also define the matrices in Equation (2.41) and Equation (2.42) as

$$A_s = \begin{bmatrix} A & B_u C_{add} \\ 0_{J\times I} & A_{add} \end{bmatrix}, \quad B_s = \begin{bmatrix} 0_{I\times J} \\ B_{add} \end{bmatrix} \quad \text{and} \quad C_s = \begin{bmatrix} C_y & 0_{N\times J} \end{bmatrix}. \tag{3.1}$$

Therefore, the mapped discrete-time state-space data are given as

$$A_d = e^{hA_s}, B_d = \int_0^h e^{\tau A} d\tau B_s, \text{ and } C_d = C_s. \tag{3.2}$$

The state-space equations of the discrete-time linear model can be written as

$$x(k+1) = A_d x(k) + B_d u(k), \text{ and} \tag{3.3}$$

$$y(k) = C_d x(k). \tag{3.4}$$

3.3 Control Modules

A model-based control algorithm was developed to automatically manipulate the lamp powers in response to errors between the wafer temperature measurements (sensor temperature signals) and the desired temperature trajectory. The issues that have to be considered when developing this control strategy include,

- time needed to damp the temperature errors,
- control effort required to damp out the temperature errors
- repeatability of the lamp power-temperature trajectory profiles,
- uniformity during the process ramps and holds,
- time needed to get the control system running on a system,
- robustness of the control strategy to slow process variations, modifications to the equipment design, or modifications to the sensors,
- number and interpretation of parameters available to the operator for ease of tuning controller in the field.

Based on the model development, the control law is developed. Among many algorithms, we chose both open-loop transient control and feedback solution to meet these objectives. The methodology to design this controller can be described as follows.

- A nonlinear model is employed and derived for application of the control system design. The nonlinear model is then linearized at a sampling instant. This linear model is placed in discrete time state-space format, and the state-space equations are given by Equation (3.3), and Equation (3.4).

- The studies of wafer temperature control in the RTP system (Nor [92]) has indicated that multivariable control of the lamp powers is more successful than the scalar control. So the control systems are designed as MIMO systems.

A more detailed block diagram for the control strategy in discrete-time is shown in Figure 3.5.

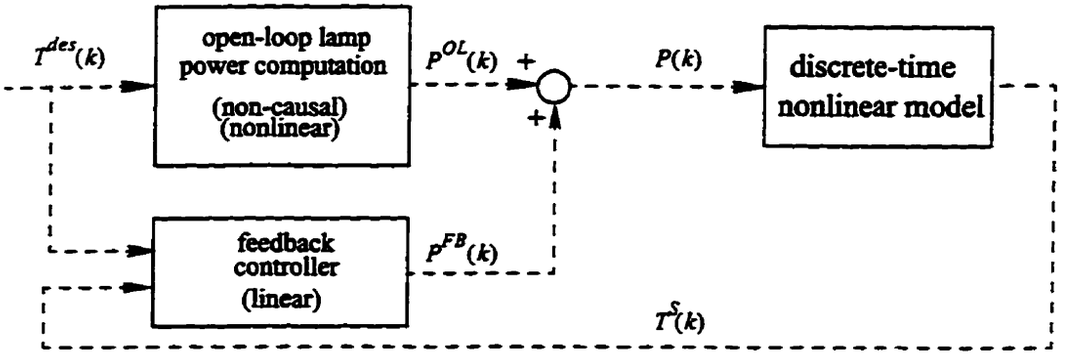


Figure 3.5 : Control system block diagram (include the wafer RTP system)

From the above block diagram, we can see that the control system has two parts. One is the open-loop power computation, which is non-causal and nonlinear (see Section 3.4 for more details). The other is the linear feedback controller (see Section 3.5 for more details), which has two sampled inputs. One is the sensor signals $T^{S}(k)$, and the other is the desired process temperature $T^{des}(k)$. The outputs of the control system also have two parts. One is the open-loop lamp power $P^{OL}(k)$, which is from the open-loop transient control, and the other is the feedback lamp power $P^{FB}(k)$, which is adjusted by the feedback controller.

3.4 Optimal Design for Open-Loop Transient Control

In this subsection, we formulate an optimal open-loop design problem to find the optimal lamp power setting for each lamp zone which results in the best achievable temperature uniformity while tracking a pre-specified trajectory.

We first derive a steady-state optimization criterion. Then we consider a dynamic optimization criteria: ramp-up and cool-down. Finally, an optimization problem is formulated for optimal lamp power settings by combining the steady-state and dynamic optimization criteria.

We start by examining the heat loss profiles over the wafer under various processing conditions and various temperatures. The basic idea behind this method is to compensate for the heat losses by manipulating the heat flux of lamp rings. Thus it follows that if we can perfectly compensate for the heat loss over the wafer, we can achieve perfect temperatures. It should be noted that the near-uniform steady-state hold is essential to uniform processing. At one sampling period, the goal would be to find the lamp power setting at each lamp zone that minimizes the worst case difference between $T_i^{ss}(k)$, the steady-state temperature of element i with these lamp power settings, and $T_i^{des}(k)$, desired temperature of wafer element i (Equation (3.5)),

$$\max_i \left| T_i^{ss}(P,k) - T_i^{des}(k) \right| = \left\| T^{ss}(P,k) - T^{des}(k) \right\|_\infty \tag{3.5}$$

One way to approximately get the values of optimal lamp powers for each sampling period, the optimal lamp power J-vector $P^{OL}(k)$ is given by

$$P^{OL}(k) = P^{ss}(k) + P^{dyn}(k) \tag{3.6}$$

where a J-vector $P^{ss}(k)$ balances the loss of the heat from the wafer by radiation and convection, and a J-vector $P^{dyn}(k)$ adds enough additional heat to obtain a rate of temperature increase close to the derivative of desired temperature at all points on the wafer.

The quantity of $P^{ss}(k)$ is evaluated by interpolation in a pre-computed table of values of $P^{ss}(T)$ at a grid of temperature T space, of approximately, 50K apart over the working range of the RTP system,

$$P^{ss}(k) = P^{ss}\left(\frac{T^{des}(k) + T^{des}(k+1)}{2}\right), \tag{3.7}$$

and $P^{dyn}(k)$ is given by

$$P^{dyn}(k) = \left(\frac{T^{des}(k+1) - T^{des}(k)}{h}\right)\frac{C_p\left(\frac{T^{des}(k+1) + T^{des}(k)}{2}\right)}{1\frac{W}{kg}}P^{unif}, \tag{3.8}$$

where J-vector P^{unif} is chosen such that for each wafer element i,

$$\frac{1}{m_i}\left[L_{i,1} \cdots L_{i,j}\right]P^{unif} \approx 1\frac{W}{kg}.$$

From Equation (3.7) and Equation (3.8), we can see that the computations of the optimal lamp powers are nonlinear and non-causal. The non-causality is caused by the fact that the computations depend on the forward information of the desired temperature. However, this is not a problem since the desired temperature is pre-specified.

Note that it is very hard and unnecessary to find the exactly optimal lamp powers for the practical controller since the wafer model will have slight errors. Therefore, we will add

the feedback controller in the design.

3.5 Feedback Control

During RTP, each wafer sees a slightly different process environment and the model also has a slight error because of uncertainty about many system variables. A feedback control system is one approach to handle these problems. Therefore, if the wafer RTP model has a slight difference from the actual system, the feedback controller can automatically adjust the feedback lamp powers to achieve the process requirements. This feedback system can be used to speed up transient response, improve steady-state characteristics, provide disturbance rejection and decrease sensitivity to parameter variations.

The control system should be able to track a wide range of desired temperature trajectories. It is inconvenient to have the nominal temperatures close to the desired temperatures during wafer RTP. The feedback controllers design in this chapter only need one set of nominal lamp powers and corresponding nominal temperatures. They will automatically set the feedback lamp powers to achieve near uniformity across the wafer at short time. The largest difference between the nominal temperature and desired temperature can be about a hundred Kelvin degrees.

We have designed several common types of feedback controllers for the wafer RTP temperature control. The simulation results will be discussed in Chapter 5. These feedback controllers are Proportional controller (P), Proportional-Integral controller (PI), Linear Quadratic Gaussian controller (LQG) and LQG controller with one additional error Integrator state (LQGI).

3.6 Designs of P and PI Controllers

Before we go to the complicated design, we would like to see how the simple controllers work. In this section, we introduce two feedback controllers: P and PI controllers. In our design cases, we only consider one situation in which to simplify and show the designs. This situation is to assume that:

- the number of sensors is equal to the number of lamp zones
- the feedback gains are applied as diagonal matrices

We use a non-linear wafer RTP model for these two controllers since the designs are based on trial-and-error wafer temperature simulations. The total number of sensors (or lamp zones) will form the size of the feedback gain.

3.6.1 P Controller

The proportional feedback control (only feedback gain controller) is one of the simplest controllers. The general form of proportional control is $u_c = K_P e$, where u_c is the control output(s), e is the control input(s) and K_P is the feedback gain (which can be scalar or matrix).

Under the assumptions made above, the general term of K_P can be simply applied as $K_P = diag([g_1, ..., g_N])$, where we use N to represent the number of sensors or actuators. We can view this proportional controller as amplifiers with "knob"s to adjust the gains up or down. Each lamp zone is independently controlled by its own amplifier.

The values of these coefficients are selected based on the maximum lamp powers of

corresponding lamp zones. We apply the largest coefficient to the corresponding lamp zone which has the largest maximum power. Initially, we have to use experience to guess values for these coefficients, then we do the simulations, and change the values a little, one at a time. Step by step, we pick the coefficients which have the acceptable temperature trajectories. Figure 3.6 represents the block diagram the control system with P controller.

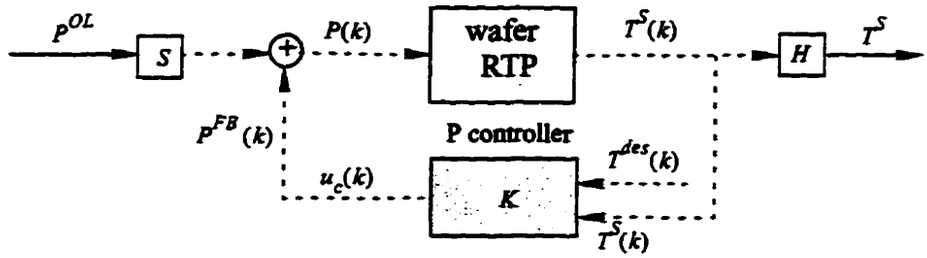


Figure 3.6 : Block diagram of P control system

In Figure 3.6, The control gain K is slightly different from K_p. This is because we have introduced different input signals for this controller. The inputs for K_p should be the errors between desired temperature and sensor temperatures in order for the plant to track the specified process trajectory. However, the inputs we have are (at k's sampling period), $\begin{bmatrix} T^{des}(k) \\ T^S(k) \end{bmatrix}$, so that we can introduce the scalar signal T^{des} to the vector signal T^S. Let us define $1_{m\times n}$ as m-by-n matrix of ones. The feedback gain K ($K \in \Re^{N\times(N+1)}$) could be represented by K_p as

$$K = K_P\begin{bmatrix} 1_{N\times 1} & -I_N \end{bmatrix}. \tag{3.9}$$

To simplify the presentation, we will use F to represent the item $\begin{bmatrix} 1_{N\times 1} & -I_N \end{bmatrix}$ in Section

3.6.2.

After designing the P controller, we need to place it back in the real-time system. Figure 3.7 shows the system we will use to implement in Chapter 4.

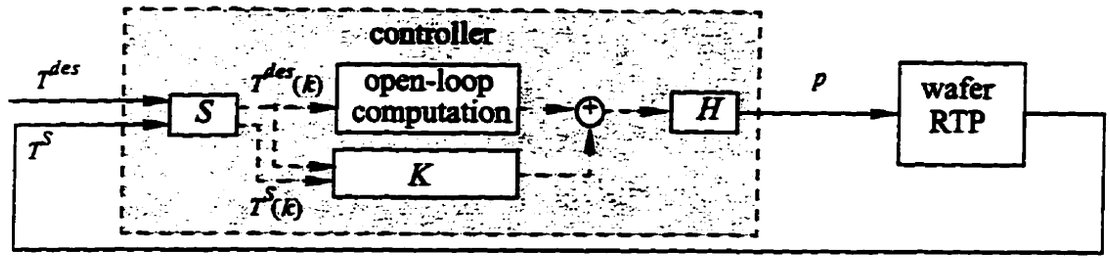


Figure 3.7 : Real-time computer control system (P controller)

The proportional feedback control can reduce error responses to disturbances, but it may have a steady-state offset (or droop) in response to a constant reference input, and may not be entirely capable of rejecting a constant disturbance. In addition, proportional feedback increases the speed of response, but enlarges the transient overshoot. For higher-order systems, large values of the proportional feedback gain will typically lead to instability.

3.6.2 PI Controller

The primary reason for integral control is to reduce or eliminate constant steady-state errors, but this benefit typically comes at the cost of worse transient response. The PI controller is one of the most practical and widely used controllers in industry. The reasons for its widespread use are that it requires very little knowledge of the plant dynamics, and the methods of determining the controller parameters are well known.

The design can be carried out by discretization continuous-time transfer function $G(s)$ to discrete-time transfer function $G(z)$. Based on the assumptions made on page 40

and the design for P controller in Section 3.6.1, we can have PI controller as,

$$G(s) = K_P + \frac{1}{\tau_{PI}s}K_I ,$$

where τ_{PI} is the time for the integrator outputs to reach K_I with input of unities. (K_I has the same size as K_P). The block diagram of PI controller in the continuous-time can be shown in Figure 3.8. (mux is used to group several scalar and vector signals into a formed vector signals).

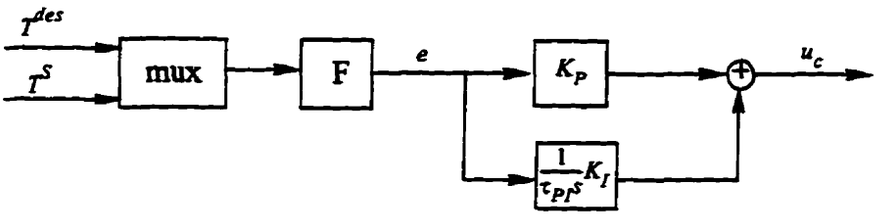


Figure 3.8 : Block diagram of continuous-time PI controller (I)

Since K_I has the same size as K_P and K_P is a diagonal matrix, we can reform the controller as shown in Figure 3.9.

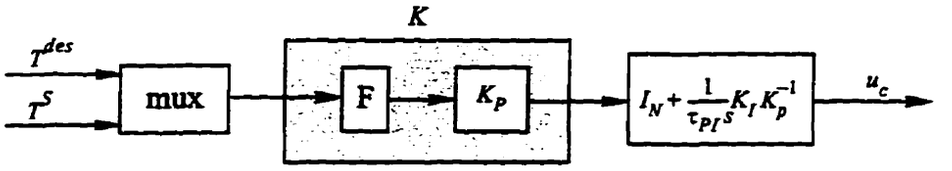


Figure 3.9 : Block diagram of continuous-time PI controller (II)

From Figure 3.9, $\frac{1}{\tau_{PI}}K_I K_p^{-1}$ is a n-by-n diagonal matrix (use $\frac{1}{\tau_i^{res}}$ to represent the diagonal coefficients for this matrix). Define K_I as $K_I = diag(k_1, ..., k_N)$. Therefore we can have $\tau_i^{res} = \frac{g_i}{k_i}\tau_{PI}$. The small τ_i^{res} value gives the fast response with a big overshoot.

A number of methods of mapping $G(s)$ to $G(z)$ are available. The method used

here is to map the $\frac{1}{s}$ to $T_{sample}\frac{z}{z-1}$, where T_{sample} is the sampling rate.

The block diagram for the wafer RTP with PI controller in discrete-time is similar to Figure 3.6. The different part is that the PI controller replaces the P controller (Figure 3.10). We form the integral part as one block called INTG.

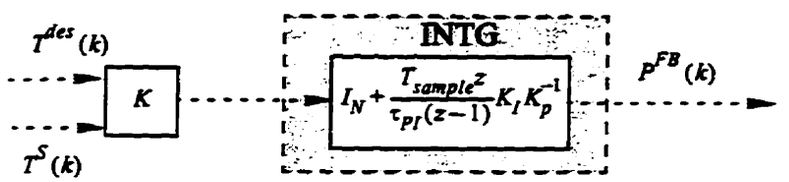


Figure 3.10 : Structure of the discrete-time PI controller

To design this particular control loop, τ_i^{res} has to be adjusted to arrive at acceptable performance.

As P controller, we place PI controller back to the real-time system as Figure 3.11.

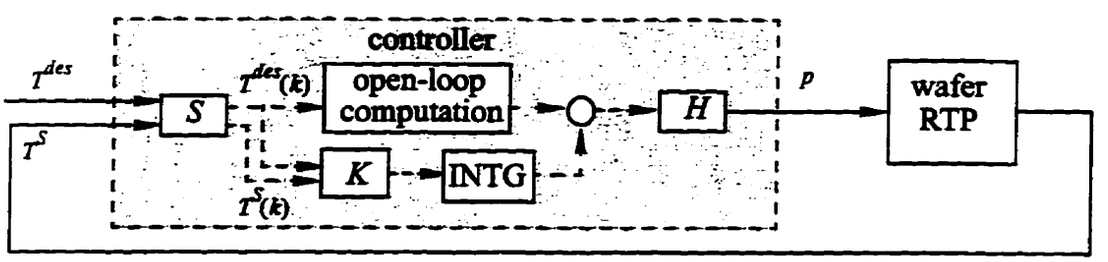


Figure 3.11 : Real-time computer control system (PI controller)

3.7 LQG Controller Design

Linear Quadratic Gaussian (LQG) control methodology is an optimal control design methodology minimizing a quadratic cost function subject to linear constraints with Gaussian noise on the inputs and outputs. The controller and estimator gains of an LQG compensator are obtained from the solutions of the controller Riccati equation and the estimator

Riccati equation. The LQG control methodology can be applied directly here because the linear model for wafer RTP is available.

3.7.1 LQG Controller for the Wafer RTP

The discrete-time linear model for wafer RTP is given as Equation (3.3) and Equation (3.4). Based on that model, we add process noise ($w(k)$) in each channel of the system inputs, and measurement noise ($v(k)$) in each channel of the system outputs. Both noises are assumed to be zero-mean white Gaussian noise for the design purpose. We do not know how the process noise affects the wafer RTP, but we can make the assumption that the process noise has the same effect to the system as the linear model inputs have. This assumption is reasonable if we can prove the controller works in Chapter 5. Therefore, the state-space equations for this system are changed to

$$x(k+1) = A_d x(k) + B_d u(k) + B_d w(k), \text{ and} \tag{3.10}$$

$$y(k) = C_d x(k) + v(k). \tag{3.11}$$

The optimal discrete-time LQG control task is to determine the controller gain (K_c) and estimator gain (K_e) such that the performance index J,

$$J = \lim_{N \to \infty} \frac{1}{N} \times E\left(\sum_{K=0}^{N-1} x(k)^T Q\, x(k) + u(k)^T R\, u(k) \right) , \tag{3.12}$$

is minimal, where E denotes the expected value, R is a symmetric positive definite input weight matrix, and Q is a symmetric positive semi-definite state weight matrix ($\cdot^T$ is the matrix transpose).

The minimum of J is obtained for the feedback $u = -K_c x$, with the gain matrix,

$K_c = B_d^T S_c$, and S_c is the solution of the controller Riccati equation,

$$S_c - A_d^T S_c A_d + A_d^T S_c B\left(R + B_d^T S_c B_d\right)^{-1} B_d S_c^T A_d - Q = 0_{(J+1)\times(J+1)}. \qquad (3.13)$$

The optimal estimator gain is given by $K_e = S_f C_d^T$, where S_f is the solution of the estimator Riccati equation,

$$S_f - A_d S_f A_d^T + A_d S_f C_d^T\left(V + C_d S_f C_d^T\right)^{-1} C_d^T S_f A_d^T - W = 0_{J\times J}, \qquad (3.14)$$

where V is a symmetric positive definite, sensor noise covariance matrix,

$$E\{v^T v\} = V,$$

and W is a symmetric positive semi-definite, process noise covariance matrix.

$$E\{w^T w\} = W,$$

and $E\{\cdot\}$ is an expectation operator.

The design of the discrete-time LQG controller can be taken in four steps:

1. design estimate gain (K_c)

2. design controller gain (K_e)

3. form LQG controller

4. introduce the desired temperature trajectory.

3.7.2 Design of Estimator Gain

The controller law, which designs the optimal feedback gain, assumed that all the state variables are available for feedback, but in the wafer RTP, not all the wafer element temperatures are measured, because the cost of the required sensors may be prohibitive, or it

may be physically impossible to measure all the state variables. Therefore, it is necessary to reconstruct all the state variables from few sensor measurements. The method of estimation used in this design is a full-order estimate, which will estimate all the states.

Since the process noise and the measurement noise are both assumed to be zero-mean white Gaussian noise, we can have,

$$E\{w\} = E\{v\} = E\{wv^T\} = 0.$$

We design a discrete linear quadratic estimator (K_e) for the system, such that the discrete, stationary Kalman filter with observation update,

$$\hat{x}(k) = \bar{x}(k) + K_e(y(k) - C_d\bar{x}(k)), \tag{3.15}$$

and state update,

$$\bar{x}(k+1) = A_d\hat{x}(k) + B_d u(k), \tag{3.16}$$

produces LQG optimal estimate states $\bar{x}$.

From Equation (3.15), we can tell the size of the estimate gain is $(J+I)$-by-J. The estimate output vector $\bar{y}$ equals $\bar{x}$.

Matlab has the function, $dlqe(A_d, B_d, C_d, W, V)$ to return the estimate gain matrix K_e.

Based on the assumption made in Section 3.7.1, we can assume that there is no relationship between noises at two lamp zones. Therefore, the process noise covariance matrix W can be applied as a diagonal matrix. The element values of this matrix depend on the maximum lamp powers of the corresponding lamp zones. We usually set the biggest coefficient to the corresponding lamp zone which has the biggest maximum lamp power. The

matrix W should look like this,

$$W = diag(w_1, w_2, \ldots, w_J), \tag{3.17}$$

where w_1 is for inner lamp zone and w_J is for the edge lamp zone.

Suppose all the sensors (the total number is N) have the same measurement quality. The sensor noise covariance matrix V should look like this,

$$V = vI_N, \tag{3.18}$$

where the scalar v is selected based on the ratio between sensor measured temperature and lamp power settings.

3.7.3 Design of Controller Gain

The second step in the LQG controller design is to find the control law as feedback of the linear combination of the state variables, that is,

$$P^{FB} = -K_c\bar{x}. \tag{3.19}$$

From the above equation, we can tell that K_c is J-by-$(J+I)$ matrix. This design is also called discrete linear-quadratic regulator design. Matlab has function *dlqr*,

$$K_c = dlqr(A_d, B_d, Q, R),$$

which solves the controller Riccati equation (Equation (3.13)), and gives the optimal constant matrices K_c.

The selections of weighting matrices Q and R are important during the optimal feedback gain design; more discussion will be displayed in Section 3.7.6.

3.7.4 Formation of the LQG Controller

The control system has sensed temperatures $T^S(k)$ as inputs and has adjusted feedback lamp powers $u_c(k)$ as outputs. Based on the discrete-time linear model (Equation (3.3) and Equation (3.4)), the optimal estimate gain K_e and the optimal estimate control gain K_c, assume all the inputs of the system are control inputs and all the outputs of the system are sensor outputs, discrete-time LQG controller can be formed in state-space model as,

$$\bar{x}(k+1) = (A_d - A_d K_e C_d + B(K_c - K_c K_e C_d))\,x(k) + (A_d K_e + B_d K_c K_e)\,T^S(k), \quad (3.20)$$

$$u_c(k) = (K_c - K_c K_e C_d)\,x(k) + K_c K_e T^S(k). \quad (3.21)$$

The compensator structure of the discrete-time LQG controller is shown in Figure 3.12. Note that the inputs $T^S(k)$ are not the final inputs to the LQG controller we use, see Section 3.7.5 for more details.

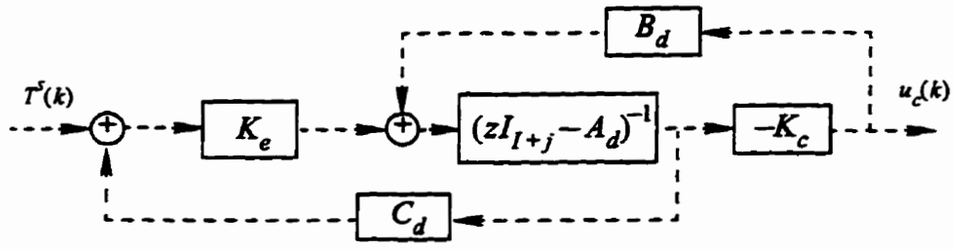


Figure 3.12 : The compensator structure of the discrete-time LQG controller

To simplify the presentation, we define A_K, B_K, C_K and D_K as the state-space data for the LQG controller. Therefore,

$$A_K = (A_d - A_d K_e C_d + B(K_c - K_c K_e C_d)),$$

$$B_K = A_d K_e + B_d K_c K_e,$$

$$C_K = K_c - K_c K_e C_d, \text{ and}$$

$$D = K_c K_e.$$

We can use Matlab function,

$$[A_K, B_K, C_K, D_K] = dreg(A_d, B_d, C_d, 0_{(J+I)\times J}, K_c, K_e)$$

to get the state-space data for the LQG controller.

Note that the LQG controller should be connected to the plant using negative feedback.

3.7.5 Desired Temperature Trajectory

The final step of LQG control system design is to introduce the desired temperature trajectory in such a way that the plant outputs will track external commands with acceptable rise-time, overshoot, and settling-time values. Therefore, the LQG control inputs have to be changed as

$$\tilde{e} = \left(\bar{y} + \begin{bmatrix} 0_{J\times 1} \\ T^{nom} \end{bmatrix} \right) - \begin{bmatrix} 0_{J\times 1} \\ 1_{I\times 1} \end{bmatrix} T^{des}.$$

Note that the estimate output vector $\bar{y}$ is a $(J+I)$-vector. The first J outputs are for the lagged lamp powers and the next I outputs are for the wafer element temperatures. Since we have to introduce the desired temperature to each of the wafer elements, we add a block, simply as a gain $1_{I\times 1}$, to transfer this scalar signal T^{des} to a vector signals. Then we combine a J dimensional zero vector signal with T^{nom} and $1_{I\times 1}$ to form the new vector signals

$\begin{bmatrix} 0_{J\times 1} \\ T^{nom} \end{bmatrix}$ and $\begin{bmatrix} 0_{J\times 1} \\ 1_{I\times 1} \end{bmatrix}$, which have the same dimensions as $\bar{y}$.

The block diagram of the LQG control system is shown as Figure 3.13, which is only used to design the LQG controller and is not the real-time system we implement in Chapter 4. The wafer RTP is represented as a linear model, linearized about a nominal temperature and nominal lamp powers.

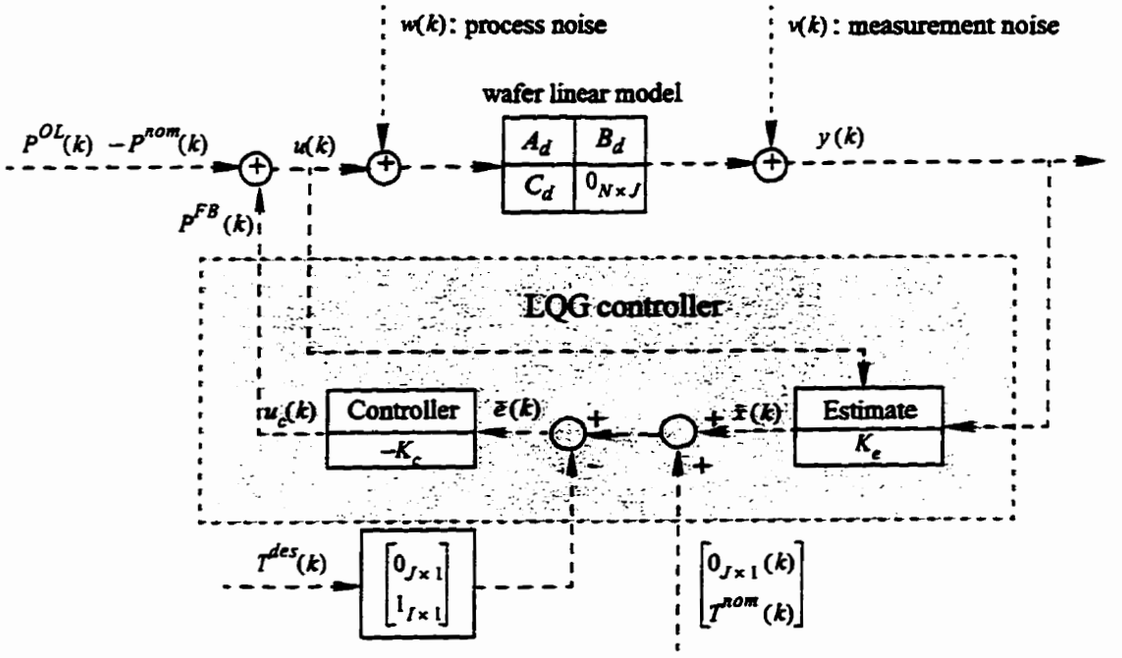


Figure 3.13 : Block diagram of LQG control system

After designing the LQG controller, we place it back in the real-time system (Figure 3.14). Note that the LQG controller is connected to the plant using negative feedback by introducing the negative sign of T^S and positive sign of T^{des}.

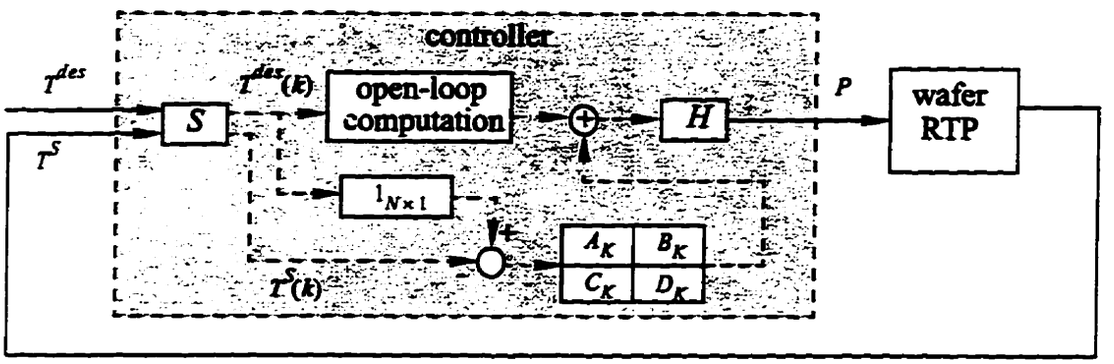


Figure 3.14 : Real-time computer control system (LQG controller)

3.7.6 Design of the Weighting Matrices for Optimal Feedback Gain

Different weighting matrices Q and R for optimal feedback gain K_c will cause huge differences in the LQG control system. We make Q small and R large if we do not want large inputs, but we lose the control precision. Otherwise, we make Q large and R small if we want fast response and do not mind large inputs. To reach the best control, we provide the calculations involved for selecting the weighting matrices below.

The input weighting matrix R is chosen as $diag(r_1, ..., r_J)$, where r_1 is for the inner lamp zone, and r_J is for the edge lamp zone. The element values of R depend on the maximum lamp powers of the corresponding lamp zones. As we mentioned in Chapter 1, the maximum lamp powers for lamp zones are all different. Usually, the inner lamp zone has the smallest maximum lamp power and the edge lamp zone has the largest maximum lamp power. We apply the biggest coefficient to the corresponding lamp zone which has the smallest maximum lamp power. Therefore, the value of r_1 should be the largest and the value of r_J is the smallest.

The calculation of the state weighting matrix Q is more complicated. In the following, we analyze the wafer temperature errors at one sampling period to generate the Q. To simplify the notation, we will remove the discrete-time notation (k) from all temperature notations and they still represent the discrete-time notations.

Let e_{ave} represent the average value for the temperature errors ($\bar{e}$ in Figure 3.13), where $e_{ave} = \frac{1}{I}\sum_{i=1}^{I} \bar{e}_i$. Then one measure of overall nonuniformity is:

$$\left[(\bar{e}_1 - e_{ave}), \ldots, (\bar{e}_I - e_{ave})\right] \begin{bmatrix} \bar{e}_1 - e_{ave} \\ \vdots \\ \bar{e}_I - e_{ave} \end{bmatrix}. \tag{3.22}$$

From the LQG controller design (one of the design goals as minimizing the sum of the squares, $e^T Q e$), we can chose a I-by-I matrix M as,

$$\bar{e}^T M \bar{e} = \left[(\bar{e}_1 - e_{ave}), \ldots, (\bar{e}_I - e_{ave})\right] \begin{bmatrix} \bar{e}_1 - \bar{e}_{ave} \\ \vdots \\ \bar{e}_I - e_{ave} \end{bmatrix}, \tag{3.23}$$

where M can be written as,

$$\begin{bmatrix} m_{11} & \cdots & m_{1I} \\ \vdots & \cdots & \vdots \\ m_{I1} & \cdots & m_{II} \end{bmatrix}.$$

The left side of Equation (3.23) is given as,

$$\begin{aligned} leftside = & \left(m_{11}\bar{e}_1^2 + m_{21}\bar{e}_1\bar{e}_2 + \ldots + m_{I1}\bar{e}_I\bar{e}_1 \right) + \ldots + \ldots \\ & \ldots + \left(m_{1I}\bar{e}_1\bar{e}_I + m_{2I}\bar{e}_2\bar{e}_I + \ldots + m_{II}\bar{e}_I^2 \right). \end{aligned} \tag{3.24}$$

For Equation (3.24), we have the descriptions as follows:

- items $\bar{e}_1^2, \bar{e}_2^2, \ldots \bar{e}_I^2$, have coefficient: m_{ii}
- item $\bar{e}_i \bar{e}_j$, has coefficient: m_{ji}.

The right side of the Equation (3.23) is given as,

$$rightside = \left(\bar{e}_1 - \frac{1}{I}(\bar{e}_1 + \ldots + \bar{e}_I)\right)^2 + \ldots + \left(\bar{e}_I - \frac{1}{I}(\bar{e}_1 + \ldots + \bar{e}_I)\right)^2$$

$$= \left(\left(\frac{I-1}{I}\right)^2 \bar{e}_1^2 + \frac{1}{I^2}\bar{e}_2^2 + \ldots + \frac{1}{I^2}\bar{e}_I^2 + \frac{2(I-1)}{I^2}\bar{e}_1\bar{e}_2 - \ldots\right.$$

$$\left. \ldots - \frac{2(I-1)}{I^2}\bar{e}_1\bar{e}_I + \frac{2}{I^2}\bar{e}_2\bar{e}_3 + \ldots \right) + \ldots$$

$$+ \left(\left(\frac{I-1}{I}\right)^2 \bar{e}_I^2 + \frac{1}{I^2}\bar{e}_1^2 + \ldots + \frac{1}{I^2}\bar{e}_{I-1}^2 - \frac{2(I-1)}{I^2}\bar{e}_I\bar{e}_1 - \ldots - \frac{2(I-1)}{I^2}\bar{e}_I\bar{e}_{I-1}\right.$$

$$\left. + \frac{2}{I^2}\bar{e}_1\bar{e}_2 + \ldots + \frac{2}{I^2}\bar{e}_{I-2}\bar{e}_{I-1} \right). \qquad (3.25)$$

For Equation (3.25), we have the descriptions as follows:

- items $\bar{e}_1^2, \bar{e}_2^2, \ldots \bar{e}_I^2$, have coefficient: $\frac{(I-1)^2}{I^2} + (I-1)\frac{1}{I^2} = \frac{I-1}{I}$
- since $\bar{e}_i \bar{e}_j = \bar{e}_j \bar{e}_i$, item $\bar{e}_i \bar{e}_j + \bar{e}_j \bar{e}_i$ has coefficient:

  $\frac{-2(I-1)}{I^2} \times 2 + (I-2)\frac{2}{I^2}$, which is $\frac{-2}{I}$.

If we compare the descriptions for both sides of Equation (3.23), we can obtain,

$$m_{ii} = \frac{I-1}{I}, \text{ and } \frac{-2}{I} = m_{ij} + m_{ji}. \qquad (3.26)$$

Since Q is a positive semi-definite weighting matrix, we can have $m_{ij} = m_{ji}$, and therefore,

$$m_{ij} = m_{ji} = \frac{-1}{I}. \tag{3.27}$$

From Equation (3.26) and Equation (3.27), the M matrix is given as,

$$M = \begin{bmatrix} \frac{I-1}{I} & \frac{-1}{I} & \cdots & \frac{-1}{I} \\ \frac{-1}{I} & \frac{I-1}{I} & \cdots & \frac{-1}{I} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-1}{I} & \cdots & \frac{-1}{I} & \frac{I-1}{I} \end{bmatrix}. \tag{3.28}$$

Note if the wafer temperature is perfectly uniform, Equation (3.23) will be equal to zero, even if the average wafer temperature is not close to setpoint. However, it is one of our design goals to let the average temperature be close to the process setpoint temperature. A measure of the deviation of the average is this sum of squares: $\begin{bmatrix} \bar{e}_1, \ldots, \bar{e}_I \end{bmatrix} \begin{bmatrix} \bar{e}_1 \\ \vdots \\ \bar{e}_I \end{bmatrix} = \bar{e}^T I_I \bar{e}$.

Finally, the Q matrix can be represented as Equation (3.29). Since we do not care about the states for the lag factor, the corresponding elements in the Q matrix are chosen as zeroes. The scalar c_2 can be adjusted to change the relative weighting of non-uniformity and setpoint error in the cost function. The scalar c_1 can be adjusted to change the relative weighting of temperature error and actuator use in the cost function.

$$Q = \begin{bmatrix} 0_{J\times J} & 0_{J\times I} \\ 0_{I\times J} & c_1(I_I + c_2 M) \end{bmatrix} \tag{3.29}$$

3.8 LQG Controller with Integrator

Like the P controller, the LQG controller generally has a non-zero steady-state error because the controller itself is only an optimal feedback gain. The LQG controller with integrator(s) (LQGI) can provide better control precision. In this section, we design a LQGI controller to take away the non-zero steady-state errors at one integrator output and make the wafer average temperature closer to the desired process temperature compared to the LQG controller.

3.8.1 Control System Structure and Weighting Matrices

In this controller design, we select one output from the linear wafer model given in Equation (3.3) as input parameter of the integrator. The output from the integrator is added as an additional input to the estimator. We number this input as Y ($Y = J+I+1$). The reformed inputs for the LQGI controller are then given as

$$\begin{bmatrix} y(k) \\ y_Y(k) \end{bmatrix} = \begin{bmatrix} C_d & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ x_Y(k) \end{bmatrix}, \tag{3.30}$$

where $x_Y(k) = x_i(k-1)$ and i is the corresponding index of the selected output.

With the new states in Equation (3.30), the linear system state equation can be regenerated as,

$$\begin{bmatrix} x(k+1) \\ x_Y(k+1) \end{bmatrix} = \begin{bmatrix} A_d & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ x_Y(k) \end{bmatrix} + \begin{bmatrix} B_d \\ 0 \end{bmatrix} u(k). \tag{3.31}$$

Use the LQG control methodology described in Section 3.7, we can design an opti-

mal estimate gain K_{eI} and an optimal feedback gain K_{cI} to combine a LQG controller for the linear system given as Equation (3.30) and Equation (3.31). We call this controller LQGI to distinguish from the regular LQG controller.

The block diagram of LQGI control system is similar to Figure 3.13, and the different part, LQGI controller, is shown in Figure 3.15. In this figure, y_i is the selected output from discrete-time linear model. (demux is used to separate a formed vector into several scaler or vector signals).

Since the inputs to the estimator add an additional dimension, the outputs from the estimator also add an additional dimension. We need to introduce one more nominal temperature signal T_i^{nom}, which represents the nominal temperature for the corresponding selected output. We also need to introduce the desired temperature signal to this estimate output.

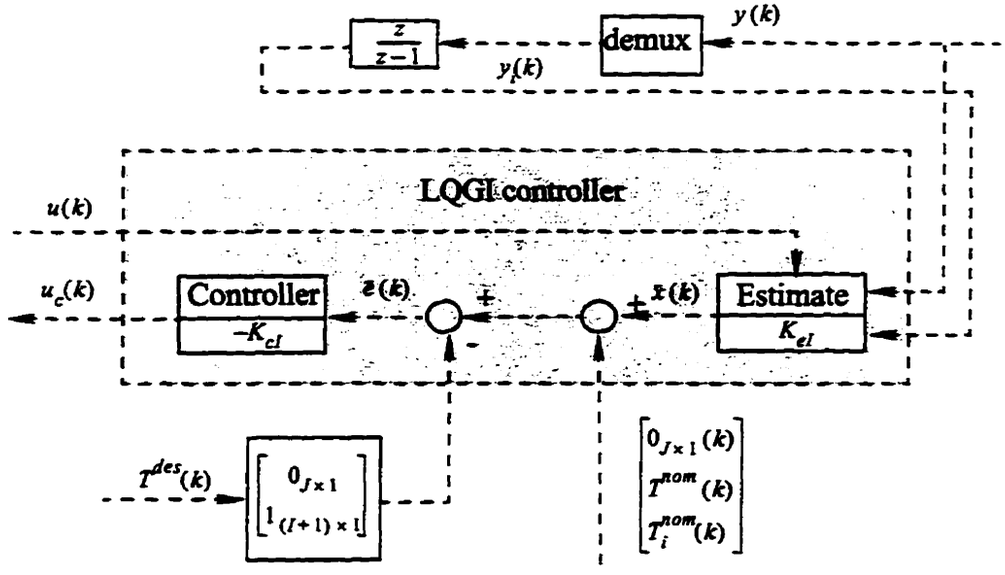


Figure 3.15 : Block diagram of LQGI controller

The sensor noise covariance matrix V_I should add one more dimension. If we treat

the integrator input as the same as the other inputs, we can have,

$$V_I = vI_{(N+1)} \cdot \tag{3.32}$$

The process noise covariance matrix W_I should be the same because the number of system lamp zones is still the same.

The state weighting matrix Q_I should also add one more dimension. If we treat the integrator state as the same as the others, we can have,

$$M_I = \begin{bmatrix} \frac{I}{I+1} & \frac{-1}{I+1} & \cdots & \frac{-1}{I+1} \\ \frac{-1}{I+1} & \frac{I}{I+1} & \cdots & \frac{-1}{I+1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-1}{I+1} & \cdots & \frac{-1}{I+1} & \frac{I}{I+1} \end{bmatrix},$$

therefore, Q_I is formed as,

$$Q_I = \begin{bmatrix} 0_{J\times J} & 0_{J\times (I+1)} \\ 0_{(I+1)\times J} & c_1(I_{(I+1)} + c_2 M_I) \end{bmatrix}. \tag{3.33}$$

The input weighting matrix R_I should be the same because the number of the system lamp zones is not changed.

3.8.2 Design Steps

Based on the LQG controller design discussed in Section 3.7, the LQGI controller can be designed using the following steps:

1. select an output, as the input parameter of the integrator
2. connect integrator in series to the wafer linear model and form a new linear plant model

3. add one more dimension on matrix V_I and matrix Q_I

4. reset the coefficient r_i for input weighting matrix R_I, c_1 and c_2 for state weighting matrix Q_I, w_i for process noise covariance matrix W_I, and v for sensor noise covariance matrix V_I.

5. design the estimate N-by-$(L+1)$ matrix K_{eI} with the integrator

6. design the feedback $(L+1)$-by-J matrix K_{cI} with the additional weighting of the integrator state

7. generate the state-space data A_{KI}, B_{KI}, C_{KI}, and D_{KI} for the LQGI controller

8. introduce the desired temperature trajectory

9. place the LQGI controller back in the real-time system as in Figure 3.16.

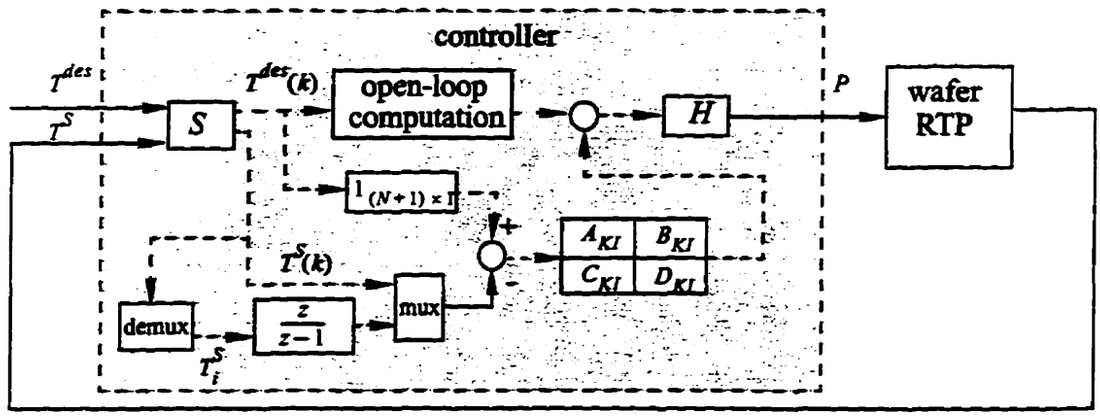


Figure 3.16 : Real-time computer control system (LQGI controller)

All the controllers designed in this chapter are tested to be stable and will be implemented in Matlab Simulink and Unix computing environment. The simulation results will be discussed in Chapter 5. The real-time software design in C language and the implementation for the wafer RTP and the control systems are discussed in the next chapter.

Chapter 4 Real-Time Software Design and Implementation

This chapter focuses on the real-time software design and the use of high-level computing languages to implement the wafer RTP system. Suggestions for graphic user interface (GUI) designs are presented near the end of the chapter. The wafer RTP mathematical model used in this chapter is given as Equation (2.46). The radiative heat transfers between wafer elements are neglected because they are really small, compared to the majority of the heat flow inside the chamber.

4.1 Introduction

Real-time computer systems must interact with the outside world on terms that are dictated by events taking place there. The computations that are done in response to those events must not only produce the correct results, but must also produce those results at the right time. The physical system usually contains several measuring devices, which the computer must interrogate to get information, as well as several actuators, which receive signals from the computer to control their actions (Figure 4.1).

At the heart of most real-time computation systems there are usually some key calculations. This could be a trend analysis of incoming data, generation of waveforms for system excitation, computation of the actuation signals on the basis of the measured signals.

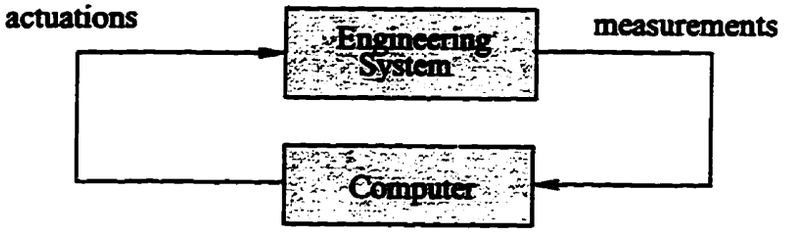


Figure 4.1: A real-time system

For the wafer RTP, the control of wafer temperatures is accomplished by adjusting the lamp powers to achieve the process temperature requirements, and to let temperatures across the wafer be close to uniformity. The real-time system for wafer RTP is shown as Figure 4.2, which has sensor temperatures as the incoming data, and lamp powers as the actuation signals. (A/D is analog-to-digital conversion, D/A is digital-to-analog conversion).

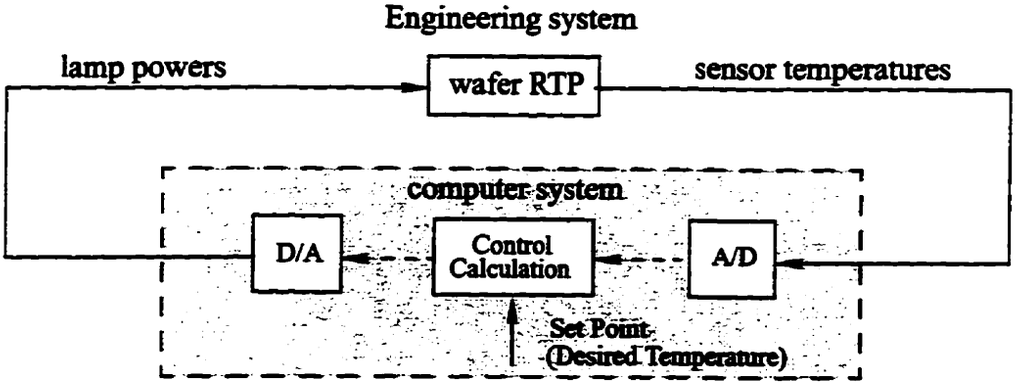


Figure 4.2: The real-time system for the wafer RTP

The above real-time system can be implemented as Figure 4.3. (PC stands for personal computer). In some research cases, the actual wafer RTP system will not exist. Therefore, we need to implement another computer system to emulate the actual system. We call this system a wafer RTP emulator. The implementation structure is shown as Figure 4.4.

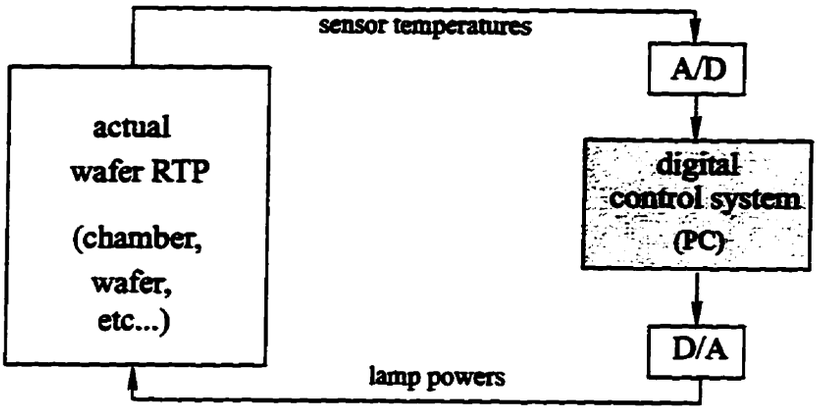


Figure 4.3: Implementation structure for the wafer RTP

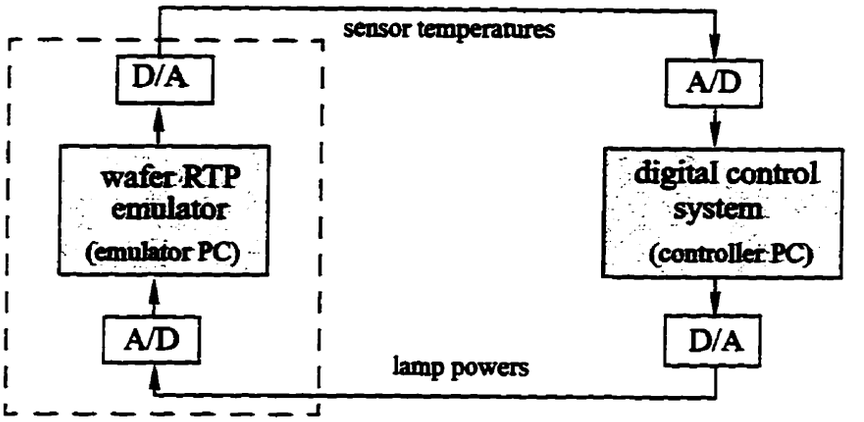


Figure 4.4: Implementation structure with the wafer RTP emulator

4.2 Application Design Overview

This section gives brief overview to the designs of the real-time software applications for the wafer RTP emulator and digital control system.

4.2.1 Design Goal

We need to design two separate software applications to install into two PCs. One is

for the digital control system, and the other is for the wafer RTP emulator. These two applications should only have limited data communications, which are sensor signals and lamp powers, through inputs and outputs (I/Os).

When the actual wafer RTP system (wafer, chamber, etc...) does not exist, we need to use both software applications. The emulator software should be able to compute the temperatures across the wafer in real-time. The second software for the control system should be able to control the lamp powers for the emulator.

When the actual wafer RTP system is attached, we only use the control system software application to control the powers of the physical lamps. This software application should have no changes whether it controls the actual system or the emulator.

We also need to design two separate graphic user interfaces (GUIs) for these two applications. These GUIs should be friendly and easy to operate. Each one will include its own user input and real-time information. The user input allows the user to manipulate the system in different situations within suitable ranges. The real-time information provides all necessary system data.

The design structure of these two software applications is shown as Figure 4.5.

4.2.2 Wafer RTP Emulator

This application is responsible for:

- initializing wafer RTP system model
- reading lamp power settings from controller
- computing temperatures across wafer at each wafer sampling time
- generating and sending sensor outputs

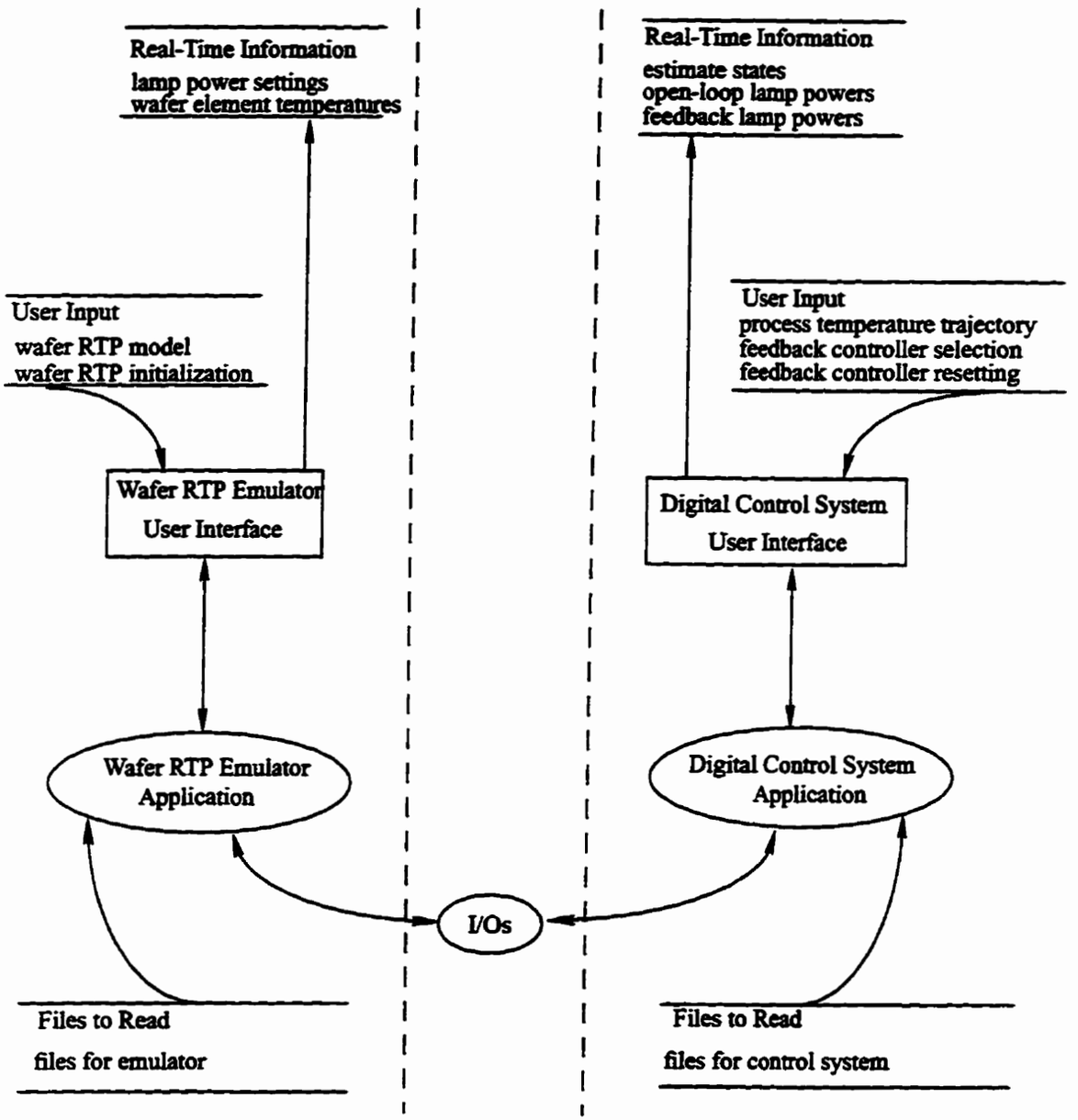


Figure 4.5: The design structure of the software applications

The user inputs are specified from the emulator PC (Figure 4.4), which may include:

- wafer RTP model (number of lamp zones, number of wafer elements, number of sensors, emissivity, thermal conductivity, specific heat, etc.)

- wafer RTP initialization (initial element temperatures, initial lamp powers, etc.)

The real-time information may have:

- lamp power settings
- real-time wafer temperature for the center element, etc.

4.2.3 Digital Control System

This application is responsible for:

- initializing the control system
- reading sensor temperatures
- updating new lamp powers at each controller sampling time
- generating and sending lamp power outputs

The user inputs are manipulated from the controller PC (Figure 4.4). It may include:

- process trajectory (preheat temperature, ramp up/down point, final temperature, process time, etc.)
- feedback controller selection (P, PI, LQG and LQGI)
- feedback controller resetting (feedback gain, rest time, weighting matrices, sampling period, etc.)

The real-time information may have:

- real-time information for the estimate states
- open-loop lamp power computation
- feedback lamp power calculation, etc.

4.3 Implementation in Unix Environment

In this thesis, we implement the software applications in a SUN station under Unix environment. This section focuses on real-time tasks. The installation of these applications into two separate PCs is introduced as suggestions in Section 4.4. The GUI designs will be included in Section 4.5, but not implemented.

4.3.1 Introduction

In this implementation, at both emulator and control system applications, several types of files are involved. We introduce them from both the user and programmer's points of views.

From the user's point of view, files include:

- editable text files (*filename.dat* or *filename*). These are edited by the user before running the simulation and the data inside of these files will be read by applications during the initialization processes. Editable text files are separated as type I (*filename.dat*) which only have numbers and as type II (*filename*) which could have parameter names and numbers.
    - For the emulator, files include *wafer.dat, ini_p.dat, ini_t, interface, io_quant.dat, model_w* and *sensor_p.dat.* (See Section 4.3.4.1).
    - For the control system, files are *set_procs, interface, io_quant.dat, ct_p.dat, ct_pi.dat ct_lqg.dat* and *ct_lqgi.dat.* (See Section 4.3.5.1).
- Matlab m files (*filename.m*). These are created by the user to generate text files. The user can program them into any type of language as long as the text files are

generated correctly. Matlab m files are an example of this implementation.

- For the emulator, the file *set_wafer.m* generates *wafer.dat*. (See Section 4.3.4.1).
- For the controller, the file *set_lqg.m* generates *ct_lqg.dat* and the file *set_lqgi.m* generates *ct_lqgi.dat*. (See Section 4.3.5.1).

- executable files (*sim_filename*). They are used for running simulations only within Unix environment. In this case, we have *sim_p, sim_pi, sim_lqg* and *sim_lqgi*. (See Section 4.3.6).
- output text files (*FILENAME.dat*). These are the log files generated when simulated experiments are running. In this application, we apply two log files, which are *WAFER_T.dat* for the emulator (Section 4.3.4.2) and *LAMP_P.dat* for the controller (Section 4.3.5.2).

From the programmer's perspective, files include:

- non-editable text files (*Filename.dat*): There are two files, *Pss.dat* and *Punif.dat* for the control system. The programmer has edited these two files to calculate the values of open-loop lamp powers. These two files should not be edited by the user since the data is generated according to the controller design discussed in Section 3.4. The data inside these two files will be read by the control system application during the initialization process. (See Section 4.3.5.1).
- non-editable Matlab binary file (*Filename.mat*). There are one group of this kind of file, *L_8.mat, L_9.mat, L_10.mat, L_12.mat*, etc. for the emulator application. They store values for fraction matrix L (Equation (2.46)) and these values

will be read by Matlab m file *setwafer.m* to generate text file *wafer.dat*. The user should not edit these files since they are generated according to the positions between system lamp zones and wafer elements. (See Section 4.3.4.1).

- C source files and header files (*filename.c* and *filename.h*). They are developed by the programmer in C language to do the implementation. Header files are used to define or declare C source files.
    - For the emulator, files are *wafer.h* and *wafer.c*. (See Section 4.3.4.2).
    - For the control system, files include *controller.h, set_tref.h, set_pref.h, set_tref.c, set_pref.c, ct_p.c, ct_pi.c, ct_lqg.c* and *ct_lqgi.c*. (See Section 4.3.5.2).
    - For the module of I/O emulations (Section 4.3.2), *files* include *wires.h, io_w.h, io_c.h, access_w.h, access_c.h, io_w.c* and *io_c.c*.

In the following subsections, we will describe the details of the implementation by files to be read, associated values defined in C header files, and C source files that make up the software applications. The three lamp zone system S shown in Figure 2.1 is used in our example.

In the further descriptions all file names are just examples for explanation and they can be renamed. If a parameter or a value in a table is marked by a asterisk (*), more explanation about this parameter or this value follows after that table.

4.3.2 Module of I/O Emulations

In our Unix simulation, the I/O boards and cables which communicate between the two applications (the emulator and the control system) introduced in Section 4.2 do not

physically exist. Therefore, besides these two applications, we need to design a module to emulate the behaviors of I/O boards and cables.

This module is the only one which provides the communication channels between emulator and controller. This is very important because in the real system, emulator and controller can not manipulate each other's system values and only communicate through I/O boards. With this module, we can separately install emulator and controller applications into two PCs without changing the code of emulator and controller applications, or install the controller application on a PC, and replace the emulator application with an actual wafer RTP system without changing the code of the controller application. We only have to add the physical hardware and replace with a new I/O software module which provides the same names of functions as this module.

Before the simulation starts, this module has to be initialized. It reads one value from a editable text file (*io_quant.dat*) as the number of quantization levels of the I/O board. For example, the most common 12-bit I/O board has the quantization level as $\left(2^{12} - 1 \right)$, which is 4,095. This value is defined according to the hardware specification. If the user wants to find out the functionality of the control system in the absence of significant quantization error, they can put a large number, like 16,777,215 (24-bits), which could not be provided by actual I/O boards. To edit *io_quant.dat*, simply open this file and change the value. Note that only one value is inside this file because we assume all the I/O boards provide the same number of bits. This file will be also read by the emulator and controller applications.

Besides the number of quantization levels, there are two other values which are quite important for this module. One is the number of communication channels and the other is

the valid range of wire voltages. In our case, these values are not editable and are defined in the C code. We have defined maximum channels as one hundred to provide more than enough communication channels and valid voltage range from minimum zero to maximum ten Volts.

According to Figure 4.4, this module should have four functions which are A/D for emulator, D/A for emulator, A/D for controller and D/A for controller. We program the first two functions in *io_w.c* and the last twos in *io_c.c*. The above four functions will call other four functions to provide wire accesses to the emulator and the controller. These four wire access functions are programmed in *wires.c*.

The header files involved in this model include:

- *wires.h*, define the number of communication channels and valid range of wire voltages
- *io_w.h*, define the I/O functions for the emulator
- *io_c.h*, define the I/O functions for the control system
- *access_w.h*, define wire access to the emulator
- *access_c.h*, define wire access to the control system.

The code of these header files are provided below.

```
/*****************************************************************
 * File: wires.h
 * Definitions for communication channels and valid voltage range
 *****************************************************************/
# define  MAX_CHANNELS 100  /* number of communication channels */
# define  MIN_V          0  /* minimum voltage                  */
# define  MAX_V         10  /* maximum voltage                  */
/*********************** end of wires.h ************************/


/*****************************************************************
 * File: io_w.h
```

```
 * Define the I/O functions for emulator
 ************************************************************************/
int emulatorAtoD(int lamp_zone);
/*
 * convert voltage read from actuator wire to quantization of
 * lamp power
 *
 * PRE:  "lamp_zone" must be positive integer
 * POST: if lamp_zone <= MAX_CHANNELS, analog voltage is read from
 *           actuator wire according to lamp zone, convert to digital
 *           quantization and this quantization is returned.
 *       if lamp_zone > MAX_CHANNELS, zero is returned
 */

void emulatorDtoA(int sensor_quant, int sensor_index);
/*
 * convert quantization of sensor temperature to voltage set to
 * sensor wire
 *
 * PRE:  "sensor_index" must be positive integer
 *       "sensor_quant" must be no less than 0 and no large than the
 *            number of quantization levels read from io_quant.dat
 * POST: if sensor_index <= MAX_CHANNELS, analog voltage is
 *           converted according to the sensor quantization and set
 *           to sensor wire according to sensor index
 *       otherwise, diagnostic is printed and program is aborted
 */
/************************ end of io_w.h ***************************/


/************************************************************************
 * File: io_c.h
 * Define the I/O functions for control system
 ************************************************************************/
int controllerAtoD(int input_index);
/*
 * convert voltage read from sensor wire to quantization of
 * sensor temperature
 *
 * PRE:  "input_index" must be positive integer
 * POST: if input_index <= MAX_CHANNELS, analog voltage is read from
 *           sensor wire according to input index and convert to
 *           digital quantization and this quantization is returned
 *       if input_index > MAX_CHANNELS, zero is returned
 */

void controllerDtoA(int actuator_quant, int output_index);
/*
 * convert quantization of lamp power to voltage set to
 * actuator wire
 *
 * PRE:  "output_index" must be positive integer
```

```
 *       "actuator_quant" must be no less than 0 and no large than the
 *           number of quantization levels read from io_quant.dat
 * POST: if output_index <= MAX_CHANNELS, analog voltage is converted
 *           according to actuator quantization and set to actuator
 *           wire according to output index
 *       otherwise, diagnostic is printed and program is aborted
 */
/************************ end of io_c.h ***************************/


/*****************************************************************
 * File: access_w.h
 * Definitions for wires access to emulator
 *****************************************************************/
double get_actuatorwirevolt(int lamp_zone);
/*
 * get voltage from actuator wire
 *
 * PRE:  "lamp_zone" must be positive integer and no larger
 *           than MAX_CHANNELS
 * POST: return actuator wire voltage according to the lamp_zone
 */

set_sensorwirevolt(double sensor_voltage, int sensor_index);
/*
 * send voltage to sensor wire
 *
 * PRE:  "sensor_index" must be positive integer and no larger
 *           than MAX_CHANNELS
 *       "sensor_voltage" must be no less than MIN_V and
 *           no larger than MAX_V
 * POST: voltage on sensor wire is set according to sensor index
 */
/************************ end of access_w.h ***********************/


/*****************************************************************
 * File: access_c.h
 * Definitions for wires access to control system
 *****************************************************************/
double get_sensorwirevolt(int input_index);
/*
 * get voltage from sensor wire
 *
 * PRE:  "input_index" must be positive integer and no larger
 *           than MAX_CHANNELS
 * POST: return sensor wire voltage according to input index
 */

void set_actuatorwirevolt(double actuator_voltage,int output_index);
/*
 * send voltage to actuator wire
```

```
 *
 * PRE:  "output_index" must be positive integer and no larger
 *           than MAX_CHANNELS
 *       "actuator_voltage" must be no less than MIN_V and no
 *            larger than MAX_V
 * POST: voltage on actuator wire is set according to output index
 */
/************************ end of access_c.h **********************/
```

To explain how this module works, we use the control system as an example. To get sensor quantizations, the controller application calls function *controllerAtoD* which calls wire access function *get_sensorwirevolt* to get voltages from I/O boards and converts voltages to corresponding quantizations for sensor temperatures. To send lamp power quantizations, the controller application calls function *controllerDtoA* to check on quantizations to make sure these quantizations are within valid ranges, converts quantizations to wire voltages, and calls wire access function *set_actuatorwirevolt* to send voltages to I/O boards.

4.3.3 Module for Matrix and Vector Calculations

The implementation involves a lot of matrix and vector calculations. It is useful to define data types which can be used to store vector and matrix values. It is also useful to design a module to handle all the matrix and vector actions.

Two data structures built in our example are

```
typedef struct
{
  int    row;  /* number of matrix row              */
  int    col;  /* number of matrix column           */
  double **el; /* pointer's point to element storage */
}MATRIX;

typedef struct
{
  int    nel; /* number of vector element   */
  double *el; /* pointer to element storage */
}VECTOR;
```

In our Unix simulation, one copy of this module is linked to the program for both applications, but it does not provide an extra communication channel between the emulator and controller. In the system where the emulator and controller run on separate PCs, each application has its own copy of this module.

The file we named *f_vec_mat.c* is programmed for this module and its functions are listed as follows:

- allocating memory for vector (*alloc_vector*)
- allocating memory for matrix (*alloc_matrix*)
- reading matrix element values from text files or MAT files(*read_matrix*)
- compute matrix-vector multiplication (*matrix_vector_multi*)
- compute vector addition(*vector_add*)
- de-allocating memories for vector (*dealloc_vector*)
- de-allocating memories for matrix (*dealloc_matrix*)

These functions are widely used by both emulator and controller applications.

4.3.4 Wafer RTP Emulator Application

This application can be divided into two portions as initialization and simulation. Details of these two process are provided below.

4.3.4.1 Initialization

Before the simulation starts, we need to initialize this application. It includes initialization of wafer model, lamp powers, wafer element temperatures, and emulator interfaces to I/O module. This portion only runs once.

The emulator application has to read values from five text files and allocate these values to corresponding parameters. The file *wafer.dat* stores the data for wafer RTP vector-matrix model and this file must be read first to get the numbers of wafer elements and sensors in the model. *ini_p.dat* has the values for initial lamp powers and *ini_t* has the values for initial temperatures across wafer. *io_quant.dat* stores the value for the number of quantization levels of I/O boards (this file is already described in Section 4.3.2 and it will not be discussed in this section). *interface* defines the values of minimum sensor temperature and lamp powers which correspond to zero quantity and this file also defines the values of maximum sensor temperature and lamp powers which correspond to the number of quantization levels of the I/O boards.

The first file discussed here is *wafer.dat*. The values inside of this file are organized according to the wafer vector-matrix model as Equation (2.46). They are listed as the scalar values for the number of lamp zones, the number of sensors and the number of wafer elements, the matrix values of A_{add}, B_{add}, $C_y C^{-1} K^{rad}$, $C_y C^{-1} K^{cond}$, $C_y C^{-1} K^{conv}$, $C_y C^{-1} L C_{add}$, T_{GAS} and a scalar value of wafer sampling period. The best way to edit this file, is not to change the element values one by one since there are too many of them, but to change several values of environment parameters and to generate *wafer.dat* by creating other files, like editable text files and Matlab m file. There are many ways to do it and we find that Matlab is one of the easiest ways. In our example, first we create two editable text files (*model_w* and *sensor_p.dat*) which include all environment parameters that are able to be edited by the user. Then we create a m file (*set_wafer.m)* to generate *wafer.dat* by reading values from *model_w, sensor_p.dat* and a Matlab M file (*L_I.mat)* which has element values

for the fraction matrix L (Equation (2.46)). This Matlab M file is not editable and is fixed by the programmer. More details are given as follows.

We program a type II editable text file, *model_w*, for the user to set the wafer model. This file is introduced in Table 4.1 by parameter names, definitions, low limits, high limits (the valid ranges are defined according to the process temperatures from 200K to 1700K) and example values. To edit this file, users simply open it and change the values (note that parameter names are not editable).

Table 4.1: User inputs for wafer RTP modeling (I)

Name	Definition	Low Limit	High Limit	Example Value
*LAMPNUM	the number of lamp zones	2	10	3
WAFERNUM	the number of wafer elements	3	30	10
SENSORNUM	the number of sensors	2	WAFERNUM	3
EMISSIVITY	wafer emissivity	0	1	0.6
THERMALCOND	thermal conductivity ($W/(mK)$)	22	266	31
SPECIFICHEAT	specific heat ($J/(KgK)$)	691	1017	916
TGAS	input gas temperature (K)	200	800	300
STEPW	wafer RTP emulator sampling period (second)	see ** below	see ** below	0.01

*total number of lamp zones is not implemented in our example since we use a three-lamp-zone system, but it could be made available in the future.

**the low limit is restricted by the hardware which we plan to implement in PC. Its high limit depends on the user requirements and this value should be at least several times

smaller than controller sampling period (which will be introduced in Section 4.3.5.1) in order for the emulator to get the updated lamp powers every time.

We have mentioned the number of sensors in Table 4.1, but it is not enough if we are only able to define the number of sensors and not their locations. Therefore we program a type I editable text file, *sensor_p.dat*, to put index numbers of wafer elements which stand for the sensor positions across wafer. The total number of this kind of parameter should be the number of sensors defined in *model_w*, and valid values are from one to the number of wafer elements defined in *model_w*. An example of how to set the sensor positions for the three-sensor-system when the wafer is broken into 10 elements is given in Table 4.2.

Table 4.2: An example for how to set sensor positions for the three-sensor-system

Sensor index	Wafer Element Number	Position
1	1	center element
2	8	third element from the edge
3	10	edge element

The M file (*L_I.mat*) used to generate *wafer.dat* stores element values for the fraction matrix L (Equation (2.46)). The user should not edit the values inside of this file because these values depend on the number of lamp zones, lamp zone positions, and number of wafer elements. They are generated according to complicated engineering calculations. Since the number of wafer elements in model is the only parameter which can be defined by the user from *model_w* in this implementation, we separately store the matrix values for L in M files according to numbers of wafer elements in the corresponding names as *L_I.mat*. For example, if we break the wafer into 8, 9, 10 or 12, the M files will be *L_8.mat*, *L_9.mat*, *L_10.mat*, or *L_12.mat*.

If an error occurs during processing, either because of the conflict of the values inside of *model_w*, *sensor_p.dat*, or other reasons, an error message will appear on the screen and the process will be terminated. As a result of this termination, the *wafer.dat* will not be generated.

The second file introduced here is *ini_p.dat*, which stores data for initial lamp powers. The data is simply organized as the inner lamp zone power to the edge lamp zone power. Note that the total number of values should be the number of lamp zones as defined in *model_w* (see Table 4.1) and all the values should be located within the valid ranges (Table 4.3). To edit this file, the user simply opens it and changes the values.

Table 4.3: An example for initial lamp power settings

Name	Definition	Low Limit	High Limit	Example Value
$Pini(1)$	initial lamp power for inner lamp zone (*W*)	0	2,000	200.37
$Pini(2)$	initial lamp power for middle lamp zone (*W*)	0	10,000	1078.08
$Pini(3)$	initial lamp power for edge lamp zone (*W*)	0	35,000	7290.52

The third file is *ini_t*, which has values for initial element temperatures. This file can be edited in two different ways. Users can either set initial temperatures as steady-state temperatures in vacuum situation under initial lamp powers, or set their own initial values. For steady-state settings, users only need to open *ini_t* and code keyword "STEADYSTATE" in this text file. For other settings, users need to open *ini_t* and code the key-word "SET" followed by the values for the inner wafer element temperature to the edge wafer element temperature. Note the total number of values should be the same as the number of wafer

elements defined in *model_w*, and all values should be inside of the valid range (which is from 200K to 1700K in this implementation example).

The file *interface* is discussed last in this sub-section. This file stores the parameter names and corresponding values which are used to establish the interfaces to the I/O module. See Table 4.4 for details. (This file will also be read by the controller application). Note that the number of values found under "Example Value" for MAXLAMPPOW should be the total number of lamp zones defined in *model_w*. (The first value is for the inner lamp zone and the last value is for the edge lamp zone).

Table 4.4: Parameters specified in the file *interface*

Name	Example Value	Explanation for example value
MINSENSORT	200	200 Kelvin degree of sensor temperature is 0 quantity for sensor signal
MAXSENSORT	1700	1700 Kelvin degree of sensor temperature is a maximum quantity for sensor signal
MINLAMPPOW	0	0 Walt of lamp power is 0 quantity for actuator signal
MAXLAMPPOW	2000 10000 30000	for three-lamp-zone system, 2,000 (10,000 or 30,000) *W*atts of the inner (middle or edge) lamp power is a maximum quantity for actuator signal

If any error appears in *ini_p.dat*, *ini_t* or *interface*, error messages will appear on the screen during the emulator initialization process and this process will be terminated. Users have to correct the editable text files they created in order to run the wafer RTP simulation.

4.3.4.2 Wafer RTP Simulation

Wafer RTP simulation retrieves the actuator quantizations from I/O boards, updates

wafer temperatures and sends the sensor quantizations to I/O boards. This simulation is running inside of a loop to emulate the real-time wafer RTP to update the simulation states at each sampling instant. Each loop can be described as follows:

- *wafer_update* calls *emulatorAtoD* to get correct actuator quantizations from I/O boards and it converts these quantizations to correct lamp powers
- *wafer_update* calls *wafer_ode* to update element temperatures by implementing the wafer RTP vector-matrix model as ordinary differential equations, using fourth-order Runge-kutta method, and it converts these temperatures to correct sensor quantizations
- *wafer_update* calls *emulatorDtoA* to send updated sensor quantizations to I/O boards.
- *write_t* writes updated element temperatures to file *WAFER_T.dat* which is a log file to store a big matrix value of the temperatures for all wafer elements at each sampling period

To view the information inside of *WAFER_T.dat*, the user can either open this text file directly to see the values or go through a software package like Matlab to load the data and generate figures.

4.3.5 Digital Control System Application

This application can be divided into two parts: initialization and controller updating. Details of each process are provided below.

4.3.5.1 Initialization

The controller initialization process only runs once before the controller goes into its update loop. This process includes setting the process temperature trajectory, calculating the open-loop lamp powers as a table of temperature setpoints, getting controller algorithms and initialization of controller interfaces to the I/O module.

Regardless of the types of the feedback controller selected by the user, this application needs to read values from four editable text files (*set_procs*, *interface*, *io_quant.dat*, and *ct_lqg.dat (ct_p.dat, ct-pi.dat* or *ct_lqgi.dat)*) and two non-editable text files (*Pss.dat* and *Punif.dat*). *set_procs* stores data for process temperature setpoints. *ct_lqg.dat (ct_p.dat, ct_pi.dat* or *ct_lqgi.dat)* has the data for the selected controller algorithms. *interface* has the data to establish the interfaces to the I/O module (this file has been described in Section 4.3.4.1 and it will not be introduced in this section). *io_quant.dat* stores the value for the I/O board maximum quantization (this file has been introduced in Section 4.3.2 and we will not repeat the details). *Pss.dat* and *Pdyn.dat* are used for open-loop lamp powers calculation.

The file *set_procs* is first introduced. We code some keywords followed by one or two scalar values to store information inside this file and thus set process temperature trajectory. These keywords are START followed by the initial process temperature, RAMP followed by the temperature ramp up (or cool down) and how many seconds this action takes from the previous setpoint to the present one, HOLD followed by how many seconds the temperature holds from the previous setpoint, FINAL followed by the final temperature and how many seconds to research this final setpoint from the previous one. One example is given as follows to show how to edit this file for the trajectory set as Figure 1.2.

```
START    1070
HOLD     2
RAMP     1120     4
HOLD     4
RAMP     1020     16
FINAL    1020     4
```

Note that the controller application only reads the values from line START to line FINAL. So we should start editing this file with keyword START and finish editing this file with the two number following the keyword FINAL. All numbers referring to temperatures should be within valid range (which is from 200K to 1700K in our example), and all numbers referring to time should be positive values. During the initialization process, we have coded *set_tref.c* to read the above data and automatically calculate the process reference temperatures at controller sampling points. We can get the total simulation duration from adding all the numbers which refer to time in file *set_procs*. If any error appears in *set_procs*, an error message will be shown during the initialization, and the process will be terminated.

Secondly, we introduce four files (*ct_p.dat, ct_pi.dat, ct_lqg.dat* and *ct_lqgi.dat*), which store the data for four feedback controller algorithms (Chapter 3). Details of these files are provided below.

ct_p.dat has the data for P controller. It is listed as the scalar values for the numbers of controller inputs and outputs, a matrix value of feedback gain and a scalar value of controller sampling period. *ct_pi.dat* has the data for PI controller. It is listed as the scalar values for the numbers of controller inputs and outputs, a matrix value of feedback gain, and the scalar values of reset time and controller sampling period. To edit these two files, users only have to open them and change the values. Note that the dimensions of feedback gains must

match the numbers of controller inputs and outputs.

The values inside of *ct_lqg.dat* and *ct_lqgi.dat* are organized according to state-space models of LQG and LQGI controllers. The data is listed as the scalar values for the number of controller inputs, the number of controller outputs and the number of the wafer elements set by the controller, the matrix values of A, B, C and D (controller state matrices), and a scalar value of controller sampling period. To edit these two files, the user should not open them and change the element values one by one since these controllers are designed under the optimal control theory. (See Section 3.7 and Section 3.8 for details). In our example, we create two Matlab m files (*set_lqg.m* and *set_lqgi.m*) to generate *ct_lqg.dat* and *ct_lqgi.dat* according to values of controller parameters read from a type II editable text file (*model_c*), and a wafer model built inside of these two Matlab m files. This wafer model is used as a plant model for the design of LQG and LQGI feedback controllers.

The editable parameters specified in *model_c* are mainly used to set weighting matrices for LQG and LQGI feedback controllers and they are listed in Table 4.5. To edit *model_c*, users can open it and change the values for corresponding parameters. The parameter names and structures are fixed as in Table 4.5 and are not editable. For correctly editing *model_c*, we have explanations below.

The controller sampling period needs to be set at least several times larger than the emulator sampling period in order for emulator to get updated lamp power settings from the control system every time.

Table 4.5: Editable controller parameters for LQG and LQGI

Name	Definition	Example Value
INPUTS	the number of controller inputs (scalar)	3
OUTPUTS	the number of controller outputs (scalar)	3
STATES	the number of wafer elements set by the controller (scalar)	10
R	input weight matrix (OUTPUTS-by-OUTPUTS symmetric positive definite matrix)	*diag* (10.8, 1.5, 0.01)
c1	coefficient to build state weight matrix Q (scalar)	10
c2	coefficient to build state weight matrix Q (scalar)	10000
v	coefficient to build sensor noise covariance matrix V (scalar)	0.0001
W	process noise covariance matrix (OUTPUTS-by-OUTPUTS symmetric positive semi-definite matrix)	*diag* (120, 2200, 80000)
STEPC	controller sampling period in seconds (scalar)	0.1

According to the LQG and LQGI design in Section 3.7 and Section 3.8, The dimensions of input weighting matrix R and the dimensions of process noise covariance matrix W should match the number of lamp zones to be controlled (three in our example since we use a three-lamp-zone system).

According to Equation (3.29), for the LQG, the dimension of the state weighting matrix Q is the sum of the number of lamp zones and the number of wafer elements (in our example, since we use the three-lamp-zone system and our controller set wafer elements as 10, *set_lqg.m* will generate it as a 13-by-13 matrix). According to Equation (3.33), for the LQGI, the dimension of Q is the addition of the number of lamp zones, the number of wafer elements and one (*set_lqgi.m* will generate it as a 14-by-14 matrix).

According to Equation (3.18), for LQG, the dimension of the sensor noise covariance matrix V is the number of sensors. (In our example, we use three sensors, therefore, *set_lqg.m* will generate it as a 3-by-3). According to Equation (3.32), for LQGI, the dimension of V is the number of sensors plus one (*set_lqgi.m* will generate it as a 4-by-4 matrix).

Note, if the user has set a wafer model which are different from our example values (LAMPNUM, SENSORNUM, WAFERNUM, positions of sensors for emulator wafer RTP, etc. see Table 4.1 and Table 4.2), the user must edit *set_lqg.m* or *set_lqgi.m* to set the number of controller inputs as SENSORNUM and the number of controller outputs as LAMPNUM. The user also must edit *set_lqg.m* or *set_lqgi.m* to set correct dimensions of weighting matrices and to build a new plant model according to their emulator model in order to design a LQG or a LQGI feedback controller based on the correct wafer RTP model. The details about how to edit *set_lqg.m* or *set_lqgi.m* are beyond our topics here. (Look at Section 2.2, Section 2.4, Section 2.6, Section 3.7 and Section 3.8 for more information).

If any error occurs during the generation of *ct_lqg.dat* and *ct_lqgi.dat*, either because of the conflict of the parameter values inside of *model_c* or other reasons, error messages will appear on the screen and the process will be terminated. As a result, *ct_lqg.dat* or *ct_lqgi.dat* will not be generated.

The third file is *Pss.dat*. It stores lamp powers used to balance the loss of the heat from the wafer by radiation and convection. The values are listed as a table of temperature setpoints from possible lowest process temperature (200K) to possible highest one (1700 K). The working range is 50K. Each row has the powers of all lamp zones at one setpoint. The

last file introduced in this section is *Punif.dat*, which stores a vector value for P^{unif} in Equation (3.8) to calculate additional heat (P^{dyn}) to obtain a rate of temperature increase of close to desired process temperature. In our example, we programmed a C file *set_pref.c* to generate the open-loop lamp powers according to the values read from *Pss.dat* and *Punif.dat*. These two files are not editable.

4.3.5.2 Control Update

Control update process retrieves the quantizations of sensor temperatures from the I/O module, updates controller states and sends the quantizations of lamp powers to the I/O module. This is running inside of a loop to update the lamp powers at each controller sampling instant. The loop can be described as follows:

- *controller_update* calls *controllerAtoD* to get correct sensor quantizations and it converts these quantizations to correct sensor temperatures
- *controller_update* updates controller states and lamp power settings using matrix-vector multiplication and vector addition, then it converts these lamp powers to correct actuator quantizations
- *controller_update* calls *controllerDtoA* to send updated actuator quantizations to I/O boards
- *write_p* writes updated lamp powers to the file *LAMP_P.dat* which is a log file to store the adjusted lamp power settings for all lamp zones at each controller sampling point

We can use the same way as described in Section 4.3.4.2 to view the information

inside of *LAMP_P.dat*.

4.3.6 Main File and Executable Files

Since our implementation is within Unix environment, we need to design a main file (*sim.c*) to run the emulator and controller applications. (This main file will not be installed if the applications are installed into two PCs). In our case, we simply apply simulations run in loops. The pseudo code for this main file is represented as follows.

```
main()
{
  /* initialize applications and modules */
  wafer RTP emulator initialization;
  controller initialization;
  emulator I/O initialization;
  controller I/O initialization;
  wire initialization;

  /* get total simulation counts */
  totalsimulation = (int)(totalsimulationduration
                          / controllersampleperiod);
  /* get wafer simulation counts inside of once controller update */
  wafersimulation = (int)(controllersampleperiod
                          / emulatorsampleperiod);

  /* start simulations */
  for(loop_counter_1 = 0;
      loop_counter_1 < totalsimulation; loop_counter_1++)
  {
    controller update;
    for(loop_counter_2 = 0;
        loop_counter_2 < wafersimulation; loop_counter_2++)
    {
      emulator update;
    }
  }/* end of loop_1 */

  de-allocate vector and matrix memories;
  /* end of main file */
}
```

To start the simulation, the user can run the executable file named by the selected controller. In our case, we have four executable files available and listed in Table 4.6. For

example, if users want to select the LQG feedback controller, they need to run the file named *sim_lqg*.

Table 4.6: List of available executable files

Execute File Name	Feedback Controller Type
sim_p	P
sim_pi	PI
sim_lqg	LQG
sim_lqgi	LQGI

4.4 Real-Time Implementation Ideas

In this section, we give brief suggestions about how to install the software applications into two separate PCs. We will focus on two topics which are not included in Section 4.3. These are the requirements of hardware and software, and the clock module which provides reference to the real-time.

4.4.1 Hardware and Software

The hardware in this implementation includes two personal computers, I/O boards and cables. A simplified diagram is shown in Figure 1.3.

The minimum requirements for the I/O boards are listed below:

- at least 100 Hz sampling rate
- 12-bit (or higher) A/Ds with at least 8 identical analog inputs
- 12-bit (or higher) D/As with at least 8 identical channels, for each of which has voltage output

- at least 8 lines of digital I/O

We suggest using National Instrument software tool called LabWindows to implement the emulator and control system applications. This software tool can provide a development environment that simplifies programming data acquisition and instrument control systems. The GUI Tools can easily create custom GUI for data presentation. It has a variety of controls-graphs, strip charts, switches, and more to display data and accept operator input. These characteristics help not only data measurement but also data presentation.

Some other software packages may also be needed. They are not listed here (such as dos/windows drive software, etc...).

4.4.2 Clock Module

The software used for the implementation study within Unix environment made no reference to time since the simulation for emulating wafer RTP and the updating of digital control system run in loops. This is very convenient for programming, but usually, the real-time RTP wafer application requires some synchronization with "real" time.

The real-time tasks in our applications only need to know relative time rather than absolute time. That means these tasks only need to know when to run, when to stop and when to run again, therefore a clock model can be used. The nature of this module is such that at the time it is run, the interval to the next time it must run again is already known. The interval timer is then set for that time, and when it runs out, the task is run again.

We use the controller application as an example. It is important that time is being kept by a process that is completely independent of the control program, and running parallel with it. The parallel process implies that nothing in the control program will interfere

with the timekeeping function, and no constraints on execution time have to be applied to the control program other than the requirement that it be able to complete its work by the time of the next sample.

If the clock model is implemented with a hardware timer, the parallel operation is achieved naturally. Suitable circuitry must be provided so that the clock can be set and interrogated. When the counting part of the clock is implemented in software, both the clock software and control software must be run at the same computer. It means that the real parallel operation cannot be achieved. However, if the computer is fast enough, both tasks can be carried out with an appearance of concurrent operation even though they run in sequence. If this is to be done in a manner that is "invisible" to the control program, a facility must be available that can suspend the execution of the control program whenever a pulse is detected, run the clock counting function, and then resume execution of the suspended program. It is called an interrupt mechanism.

The interrupt mechanism is a hardware device which sends a signal to the process requesting an interrupt. If the processor's operating model is such that the signal can be recognized, it initiates the interrupt by suspending the execution of its current program and saving whatever internal processor information is necessary to restart that task when the interrupt has been completed. The processor then starts the execution of the foreground task, the clock module. When the foreground task has completed its work, the process is reversed. A signal is sent to the device that caused the interrupt indicating that interrupt processing is complete. The saved information is used to restore the process state to where it was when the interrupt first occurred. The high priority background process (control program) is

resumed. Then the low priority background process (the graphic display) is also resumed. The sequence of events is illustrated in the following list:

1. hardware device requests an interrupt
2. processor recognizes request
3. execution of existing task is continued till the end of the current machine instruction
4. internal processor status information is saved
5. foreground task (interrupt routine) is started
6. foreground task finished
7. background task's status information is restored
8. high priority background resumes
9. low priority background resumes

From the above sequence, the high priority background task (item 8) in our controller application is the "Controller Update" as discussed in Section 4.3.5.2. In our emulator application, this task is "Wafer RTP Simulation" as discussed in Section 4.3.4.2. Low priority background task (item 9) is the user graphic display for each application (which will be discussed in Section 4.5.1 and Section 4.5.2). Background tasks are the only tasks we have implemented among all real-time tasks (initialization tasks discussed in Section 4.3.4.1 and Section 4.3.5.1 are not real-time tasks). If the displayed update task (low priority background task) is not finished when the interrupt comes, it will be resumed at the next time after emulator or controller states have been updated (high priority background task) and before the next interrupt comes. If this happens, the displays will not be updated until the display task is resumed and finished. If this situation happens all the time, the programmer

has to think about using faster computers or reduce the user displays.

4.5 User Interface Designs

The suggestions about how to design user interfaces are given in this section. The programmer can build GUIs using the LabWindows introduced in Section 4.4.1 or other software packages to present the same information.

For implementing the applications on two PCs, each computer will have its own GUI. The design details will be introduced below.

4.5.1 Wafer RTP Emulator

For the wafer RTP emulator application, we suggest designing one main window and three sub-windows. Figure 4.6 shows the basic structure of the design. According to Figure 4.4, these windows will be implemented in the emulator PC.

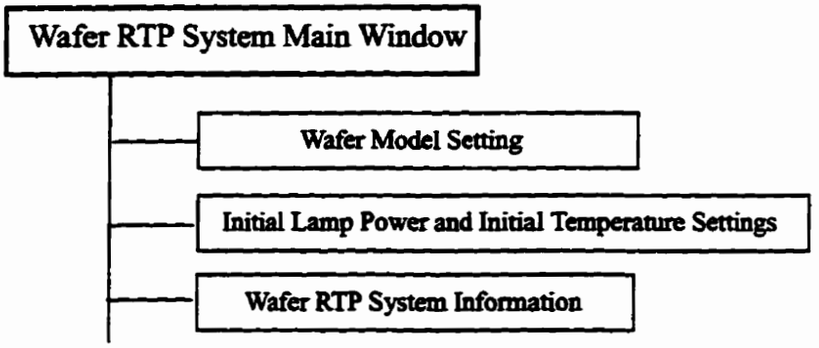


Figure 4.6: Window organization overview for the emulator

The main window will appear first when we go into the user interface of this application. From the main window, we are able to open three sub-windows; two are for application

initialization and one is for wafer RTP simulation. These windows are implemented as multi-display which means multiple windows can be displayed simultaneously. More details for the design of each window are described from Section 4.5.1.1 to Section 4.5.1.4.

4.5.1.1 Wafer RTP System Main Window

The main window is the first window we see when we use this GUI. One of our suggestion example is shown as Figure 4.7. We can do two things on this main window. These two things are introduced as follows.

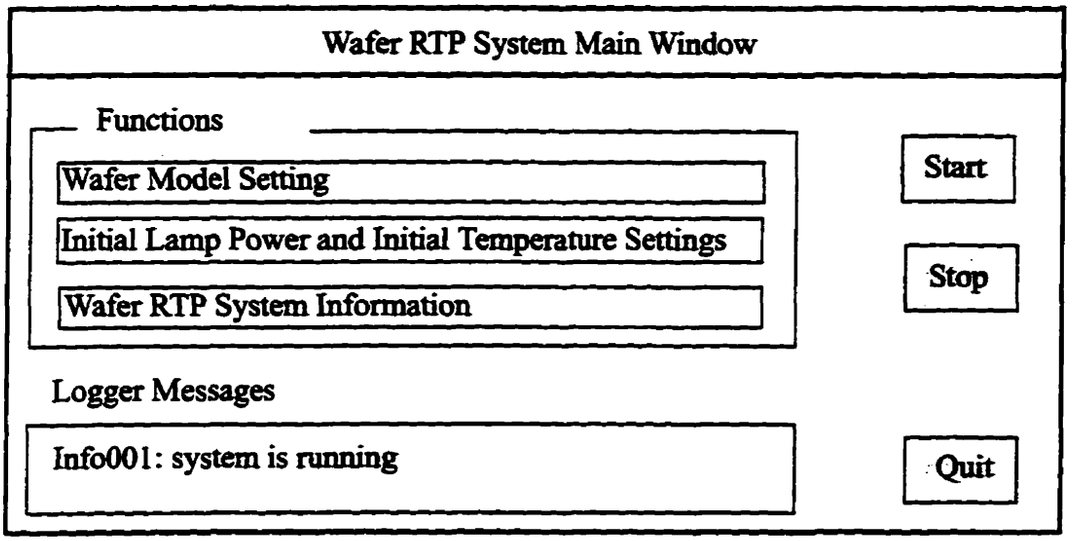


Figure 4.7: Wafer RTP system main window

First, we can open the sub-windows to initialize the system or to view the simulation information under some conditions. When the simulation is not running, we are able to open the sub-windows "Wafer Model Setting" and "Initial Lamp Power and Initial Temperature Settings" to set the parameter values. But if the simulation is running, we are only able to open them to view the parameter values. The sub-window "Wafer RTP System Information" is only able to be opened after we run the simulation once. A sub-window can be closed

using the "Quit" button within the sub-window.

Second, we can start or stop the simulation using the "Start" or the "Stop" button. The first time we run the simulation, if initialization values are not set, the emulator will use default values. During the simulation, if we click the "Stop" button, the simulation will stop running. At the same time, the warning message will appear in the controller application and the controller application will stop running as well. If we click the "Start" button again, the simulation will start from the beginning.

Messages are provided in message logger to explain what the system is doing. These messages can be divided as error messages (like Err008: initial temperature setting errors, etc...), warning messages (like Warn001: no signals from control system, etc...) or information messages (like Info001: system is running, etc...)

If we click the "Quit" button, the emulator application will be terminated and the system will be off-line.

4.5.1.2 Wafer Model Setting

This sub-window is the interface to set or view the parameter values for the wafer model. We suggest to design it as in Figure 4.8. The first time we open it, all the parameter values will be filled by default values. (See Table 4.1 under "Example Value").

The configuration table in this sub-window only has one record which includes all the editable parameters for the wafer model during a real-time on-line situation. To configure the values to this table, the user simply selects the value for each parameter and then uses the "Save" button. Choosing the "Save" button will save the information and lock the configured values. If an error occurs, as a result of editing the parameters, an error message

window will appear when the user attempts to save the changes. Once the user has successfully edited the values in this window, these values will be read by the emulator application during the emulator initialization process.

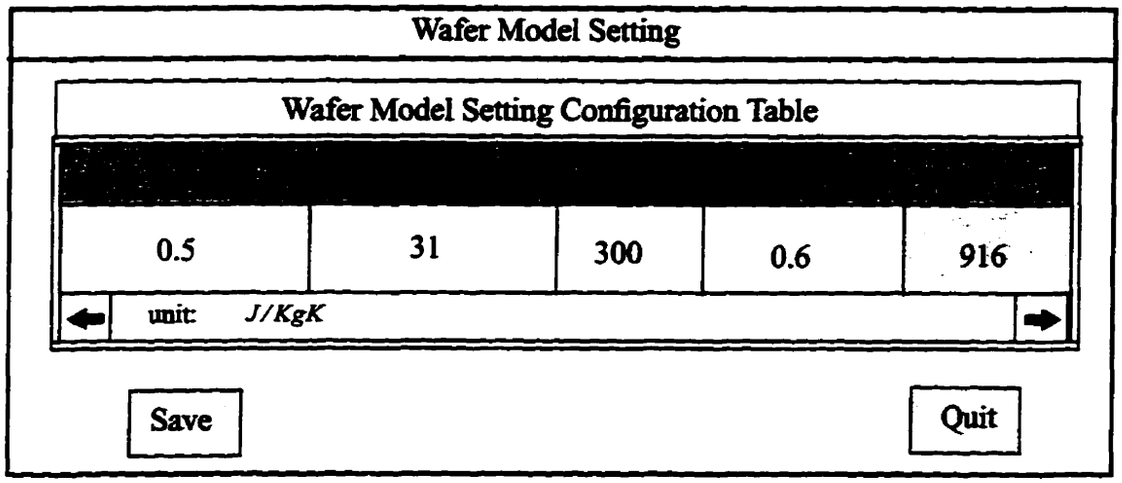


Figure 4.8: An example for the sub-window "Wafer Model Setting"

Three parameters listed in Table 4.1 (the numbers of lamp zones, wafer elements and sensors) and all parameters in Table 4.2 (sensor positions) are not editable from this configuration table. These parameters only can be edited during an off-line situation. This is done to avoid drastic changes to both the emulator and controller applications.

When the simulation is running, we are only able to view the information in this window. This means that all parameter values are not editable and are locked in the configuration table.

4.5.1.3 Initial Lamp Power and Initial Temperature Settings

From this sub-window, the user can set the initial lamp powers and initial wafer element temperatures. An example design is given in Figure 4.9, where two configuration

tables are available. One is for setting initial lamp powers and the other is for setting initial temperatures. These two configuration tables include all the editable parameters listed in Table 4.3. The first time we open this sub-window, all the parameter values will be filled by default values. For each configuration table, detail explanations are provided below.

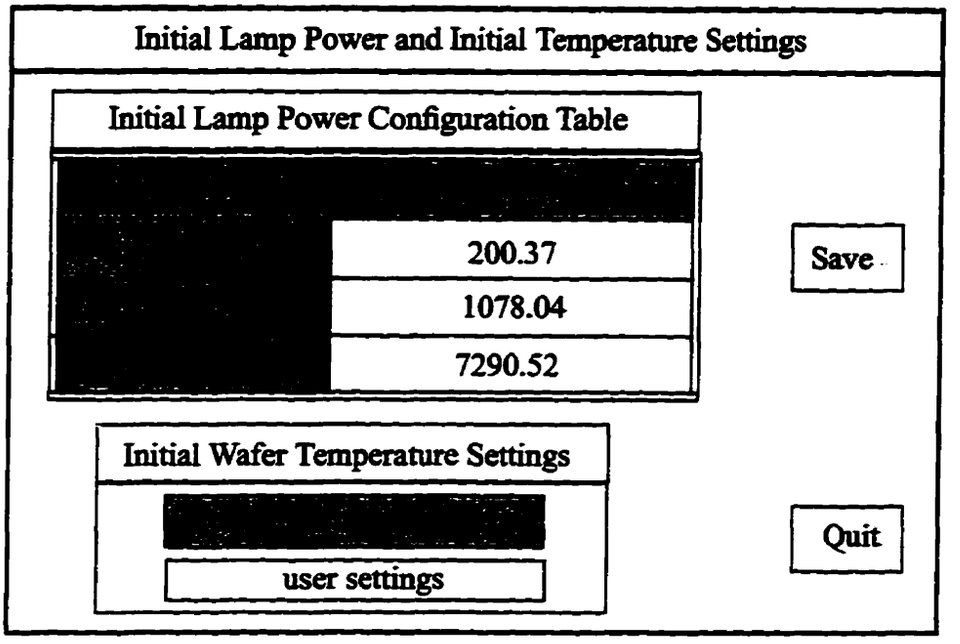


Figure 4.9: Sub-window for initial lamp power and initial temperature settings

For the first table, the first record is for the center lamp zone and the last record is for the edge lamp zone. The total number of records is fixed according to the number of lamp zones configured while off-line. All values for initial lamp powers should be inside the valid range.

For the second table, the user has two selections, either "steady-state temperatures" which refers to the initial conditions for wafer elements will be set as steady-state temperatures under the initial lamp powers configured in the first table, or "user settings" which will bring a sub-table shown in Figure 4.10 for the user to configure. In this sub-table, the first

record is for the center wafer element and the last record is for the edge wafer element. The total number of records configured should match the value configured for the number of elements on off-line situation. All values should be within valid range (which is from 200K to 1700K in this example). By clicking the grey box in top left side, the user can close this sub-table and confirm the configuration.

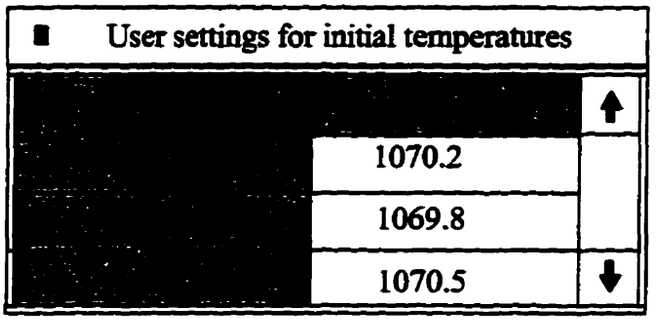


Figure 4.10: One example of the initial temperature settings

Once this sub-window (Figure 4.9) has been configured, the user can click the "Save" button to complete the initial settings. (If an error occurs, an error message window will appear).

When the simulation is running, we are only able to view information in this window. This means that all values are not editable and are locked in their configuration tables. If the user has selected "user settings" for "Initial Wafer Temperature Settings", the sub-table (Figure 4.10) will also be shown.

4.5.1.4 Wafer RTP System Information

This sub-window displays all the real-time running information for the wafer RTP system. The running information will be updated at suitable durations according to what kinds of data are shown. An example of this window is given as Figure 4.11.

This example window has selections to display various information, such as

- real-time temperature of the center element
- real-time element temperatures compared with the one of the center element
- real-time lamp power settings

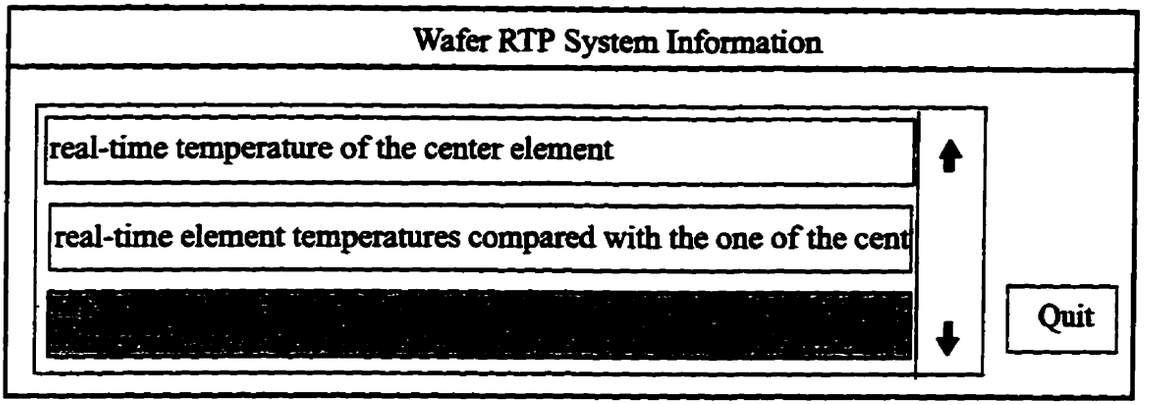


Figure 4.11: An example window for the emulator information

The designer can remove above information or can add more information. When the user selects a category to view, each of the above items of information will be displayed in its unique window. These windows are independent. They can be displayed simultaneously and can be closed by clicking the little grey box at the top left of each window.

We give examples to show how to design these windows to carry real-time information as follows.

Figure 4.12 displays the real-time temperature trajectory of the center element. The x-axis is the simulation time in seconds and the y-axis is the temperature in Kelvin degrees. After each wafer sampling instant, the sampled temperature of the center element will be added to in the figure. From this sub-window, users also can know the radius of the wafer and the "center radius" (Section 2.2) of the center element.

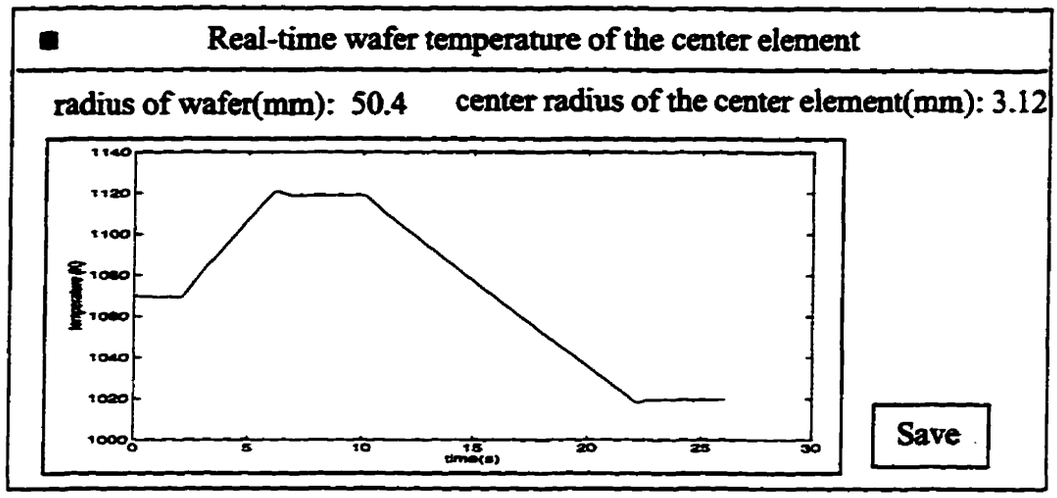


Figure 4.12: Real-time wafer temperature of the center element

Figure 4.13 displays the subtraction of the temperature at the center element from temperatures of all elements. The x-axis is the radius in millimeters and its scale is fixed as the radius of the wafer. The y-axis is the temperature in Kelvin degrees and its scale is fixed as ±5 K. These relative wafer temperatures are represented as asterisks (*) at their "center radius" positions. The refresh rate of this display depends on the hardware. The programmer can update it once every four times of updating the emulator simulation, which is about 25 times per second, if the emulator simulation is not slowed down. Otherwise, the programer should reduce the display rate. From this window, users also can know the number of wafer elements.

Figure 4.14 displays the real-time lamp power settings for all lamp zones. The x-axis is the simulation time in seconds and the y-axis uses the logarithmic (base 10) scale. For each lamp zone, the data displayed is generated according to the Equation (4.1) because the lamp powers can be in wide range (which could be from 50W to 150KW for different lamp zones) and the lamp powers compared with their maximum lamp powers also can be in wide

range (which could be from 0.0001 to 0.1 for the same lamp zone). After each sampling instant, this figure will add new data. From this window, users also can know the number of lamp zones and the maximum lamp powers of each lamp zones.

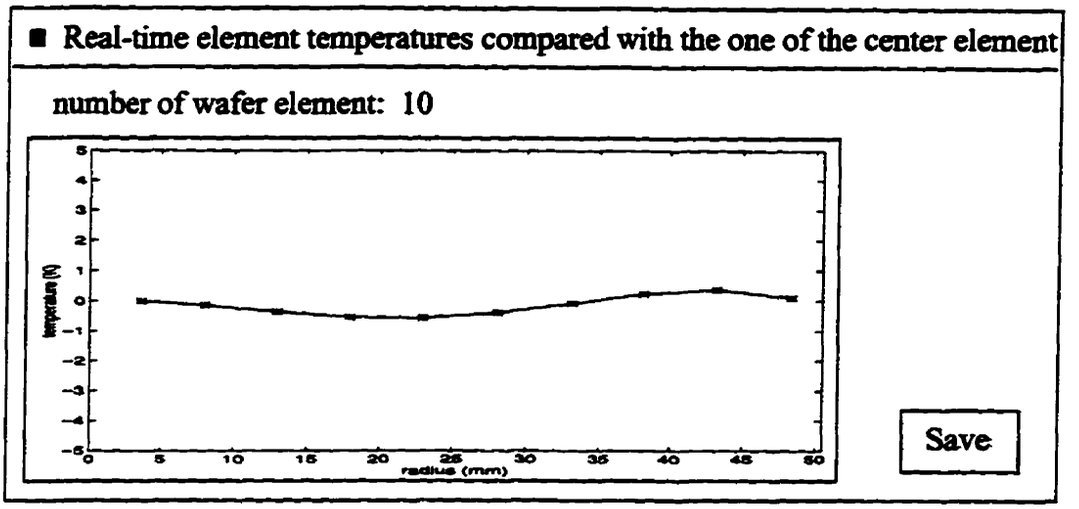


Figure 4.13: Real-time wafer temperatures compared with the center element

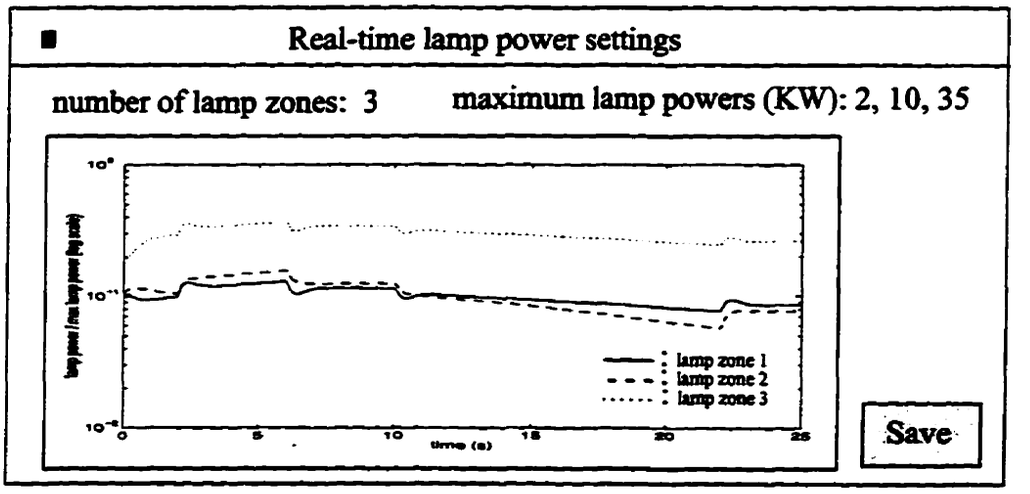


Figure 4.14: Real-time lamp power settings

$$data = \log\left(\frac{RealTimeAbsoluteLampPower}{MaximumLamPower}\right) \quad (4.1)$$

When the simulation is running, all data is stored in buffers, and they will be replaced when the next simulation starts. However this data can be saved to files by clicking the "Save" button at each window after simulation stops. A window (Figure 4.15) will appear and users only have to fill in the file name and click the "Confirm" button to save or click the "Cancel" button to cancel. If users want to view this data, they can either open the files to see the number directly, or use some software packages to load numbers from files to generate figures. These can only be done when simulation is not running.

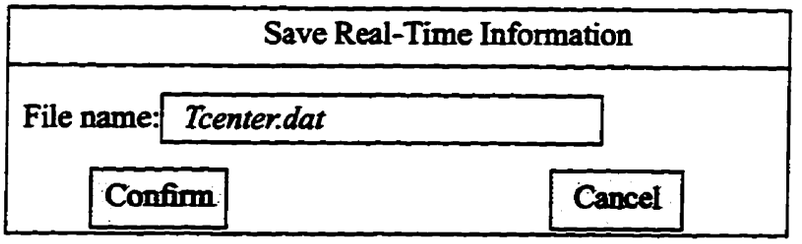


Figure 4.15: An example window to save real-time running information

4.5.2 Digital Control System

The design of the user interface for the control system is similar to the one for the emulator. We also suggest to implement one main window and three sub-windows. Figure 4.16 shows the organization of example windows which will be implemented in the controller PC (Figure 4.4).

Main window will show up first when we go into the user interface of this application. From the main window, we are able to open three sub-windows. These sub-windows are implemented as multi-display. More details for the design of each window are described from Section 4.5.2.1 to Section 4.5.2.4.

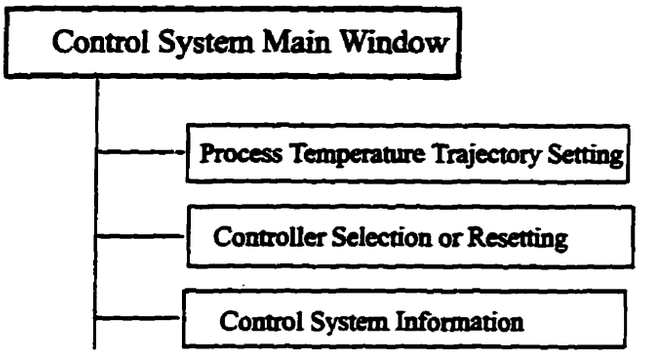


Figure 4.16: Window organization overview for the control system

4.5.2.1 Control System Main Window

This screen has similar functions as "Wafer RTP System Main Window". An example is shown in Figure 4.17, which allows users to open sub-windows, to start and to stop the control system.

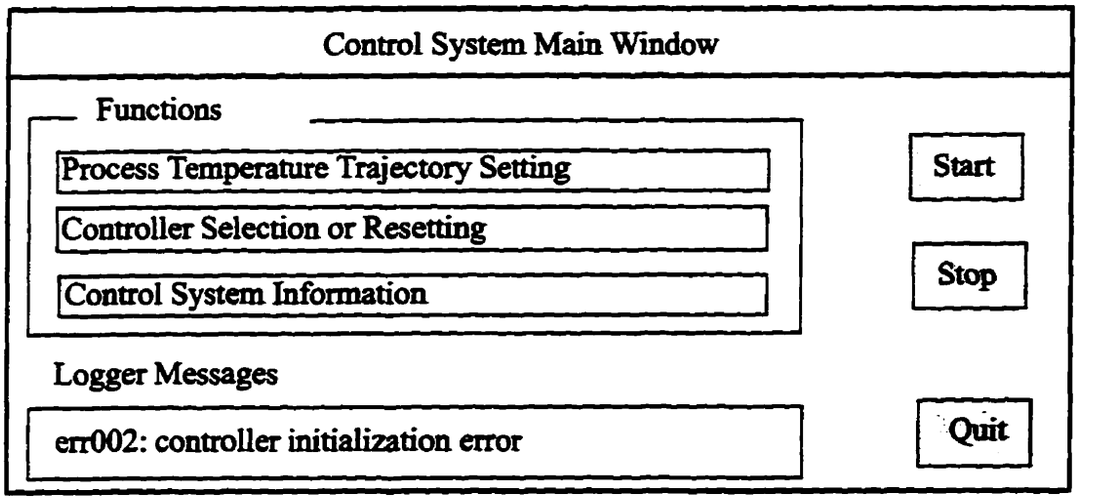


Figure 4.17: Control system main window

When the control system is not running, we are able to open "Process Temperature Trajectory Setting" to set the process temperature, and "Controller Selection or Resetting" to select a controller or to reset a controller. If the system is running, we are only able to open

them to view the information. The sub-window "Controller System Information" is only able to be opened after we run the control system once. A sub-window can be closed from "Quit" button within the sub-window.

We can start or stop the simulation by clicking "Start" or "Stop" buttons. The first time we run the control system, we must set the process temperature trajectory and select/ reset a controller. If we click the "Stop" button, the controller application will stop running. At the same time, an warning message will appear in the emulator application and the emulator application can still run at the open-loop situation. To restart the control application, the user needs to stop running the emulator application and start both applications from the beginning.

The message logger will provide messages to explain what the control system is doing. These messages are divided in three types (error, warning, and information).

4.5.2.2 Process Temperature Trajectory Setting

This sub-window is the interface to set the process temperature trajectory. One of the suggestion designs is shown as Figure 4.18.

When the user starts setting the process temperature trajectory, first, set "Initial Temperature" (Kelvin degree), which stands for the first setpoint value and this value only needs to be set once. Second, set the value of "Temperature" (Kelvin degree) which stands for the present setpoint value, then set the value of "Time" (second) which refers to how many seconds the action is taken from previous setpoint to present setpoint. "Temperature" and "Time" work as a pair and they can be set as many times as desired.

After setting the initial temperature or a setpoint, the user can click "OK" to confirm

the selection. Then in the bottom left window, this point will be added in the figure as asterisk (*) and be linked with previous setpoint. If there is no setpoint, the figure will only show the x-axis and y-axis.

The "Cancel" button will clear all the points included in the initial temperature. Therefore, the user needs to start from setting the "Initial Temperature" to get the new process trajectory.

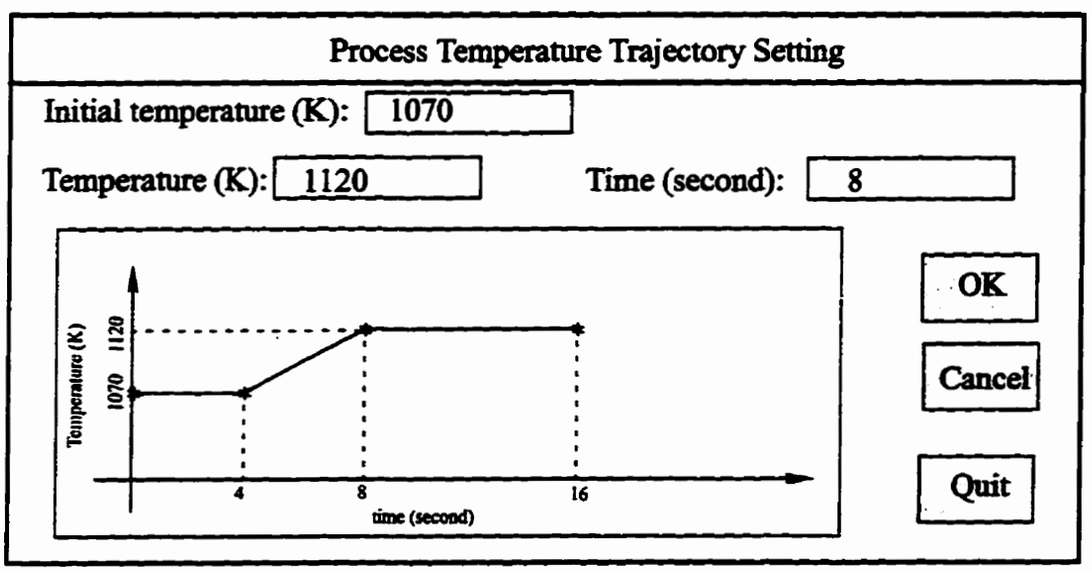


Figure 4.18: An example sub-window for "Process Temperature Trajectory Setting"

4.5.2.3 Controller Selection and Resetting

This sub-window (Figure 4.19) has the functions to select or reset a controller. All controller states are automatically set according to the initial wafer temperatures set by users. The controller inputs and outputs are fixed according to the number of sensors and the number of lamp zones configured while off-line. From this sub-window, we only can view this information.

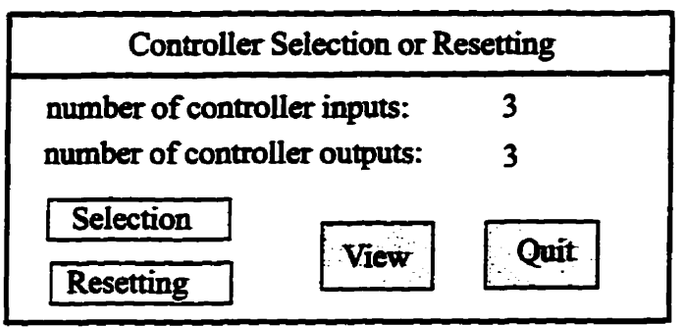


Figure 4.19: An example window for controller selection and resetting

To select a controller, the user just needs to click the "Selection" button. A window shown in Figure 4.20 will appear and it will list all available controllers. As can be seen, the user can click a button to select a controller. Therefore this window will automatically close to confirm the selection and load the data to buffers. If no controller matches, an error message window will appear.

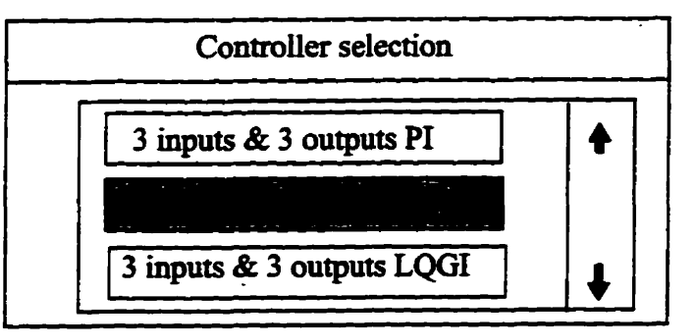


Figure 4.20: Window for selecting a controller

To reset a controller, the user just needs to click the "Resetting" button (Figure 4.19), a window (Figure 4.21) will appear. It always has four different types of feedback controllers (P, PI, LQG, and LQGI) to be set. After the user clicks a button to select the type of feedback controller, one of the windows shown in Figure 4.22 will appear to allow the user to set parameter values of the feedback controller. (See Section 4.3.5.1 in page 84 for the

details about how to set these values). After the user closes that window by clicking the little grey box at top left, the data will be loaded to the buffers. If any error appears in the setting, an error message window will appear.

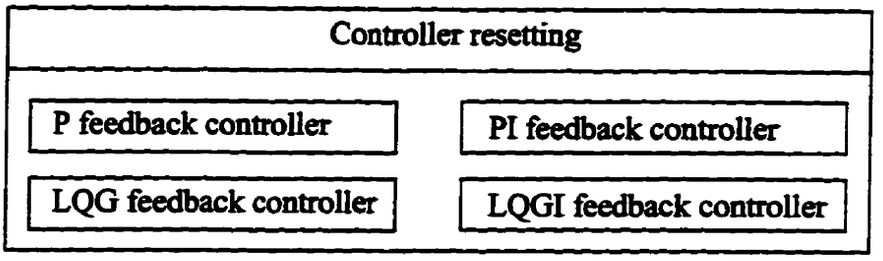


Figure 4.21: Window for resetting a controller

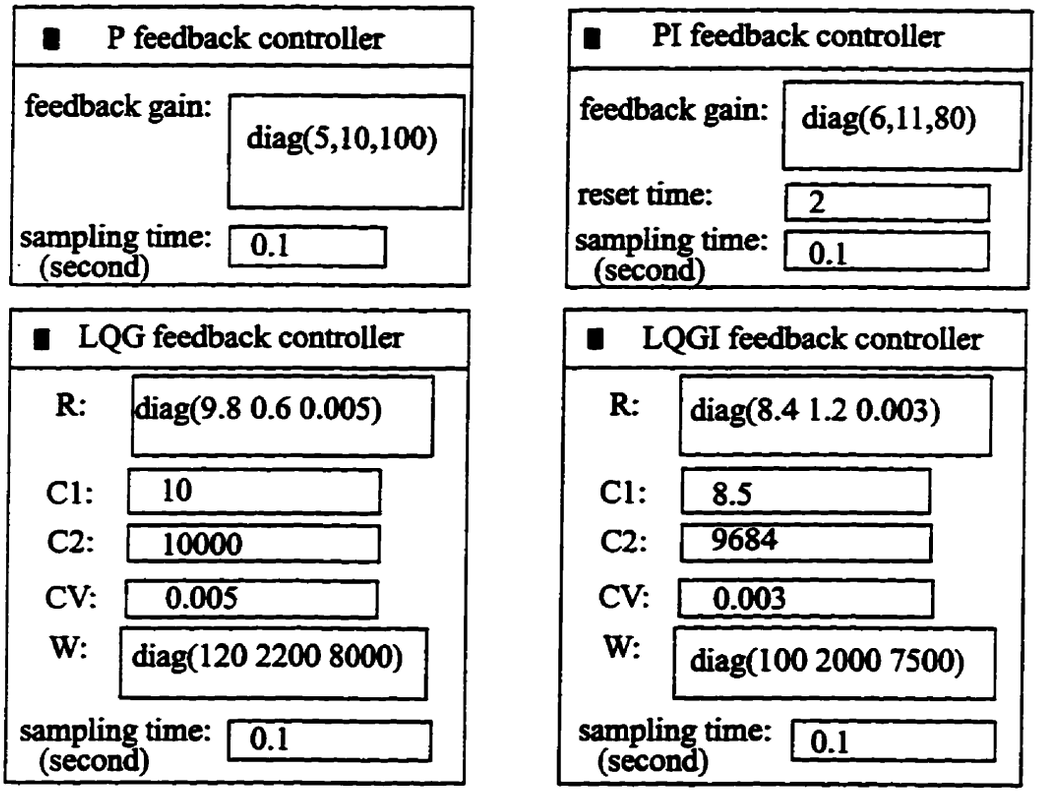


Figure 4.22: Windows to set parameters for feedback controllers

Through "View" (Figure 4.19), we can view the information of the selected controller. One corresponding window shown as Figure 4.22 will appear. It has information like

gain, reset time, and sampling time for users to view, but not edit.

4.5.2.4 Control System Information

This sub-window presents the real-time running information for the control system. It has the similar functions as "Wafer RTP System Information" introduced in Section 4.5.1.4. One of the example windows is shown as Figure 4.23.

This example window has the selections to view various information as:

- real-time estimate state of the center element (only available for LQG and LQGI feedback controllers)
- real-time estimate states compared with the one of the center element (only available for LQG and LQGI feedback controllers)
- real-time open-loop lamp powers and total lamp powers

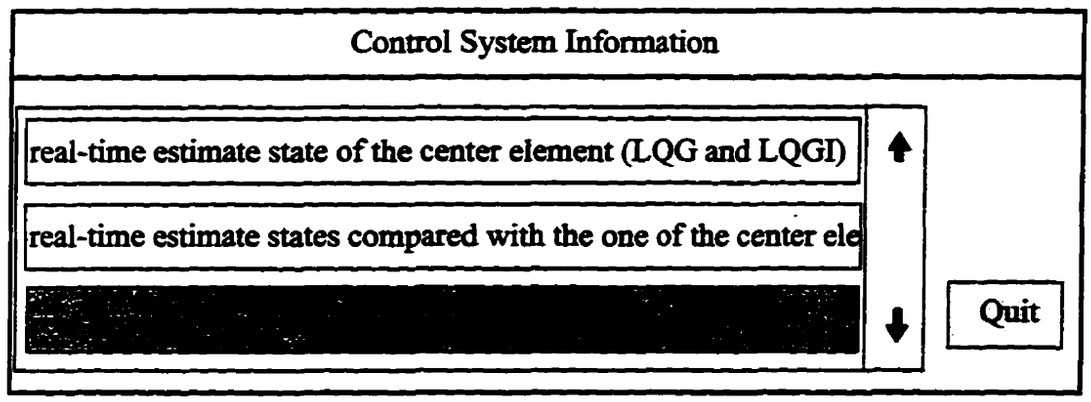


Figure 4.23: An example sub-window for "Control System Information"

The programmer can remove the above information or add more information. Each item of information will be shown in its unique window when users select to view it. All these windows can be displayed simultaneously and can be closed by clicking the little grey box at top left of that window. We will give examples to show how to present the above

information as follows.

Figure 4.24 displays real-time estimate state (temperature) of the center element. This display window is only available for LQG and LQGI feedback controllers. The x-axis is the simulation time in seconds and the y-axis is the temperature in Kelvin degrees. After each time the controller outputs update, the updated estimate temperature will be added to the figure. From this window, users also can know the "center radius" (see Section 2.2) of this center element and the number of estimate states.

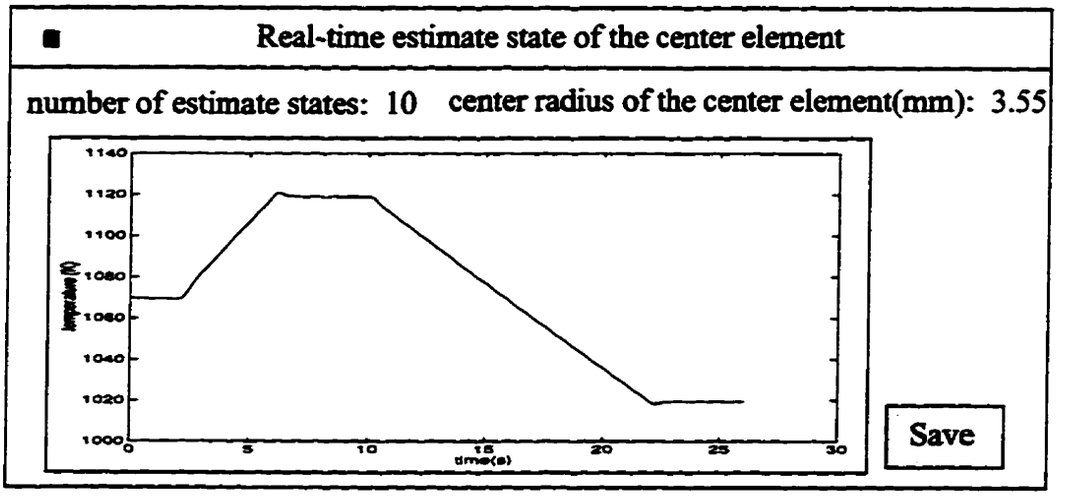


Figure 4.24: An example window for real-time estimate state of the center element

Figure 4.25 displays the real-time relative temperatures of subtracting the estimate temperature of the center element from ones of all elements. This sub-window is also only available for LQG and LQGI feedback controllers. The x-axis is the radius in millimeters and its scale is fixed as the radius of the wafer. The y-axis is the temperature in Kelvin degrees and its scale is fixed as ±5 K. These relative temperatures of estimate states are represented as asterisks (*) at their "center radius" positions. The refresh rate of this display depends on the hardware. The programmer can update it every time the controller outputs

(lamp powers) are updated, which is about 5 times per second, if the update of controller outputs is not slowed down. Otherwise, the programmer should reduce the display rate. From this window, users also can know the numbers of controller inputs and outputs.

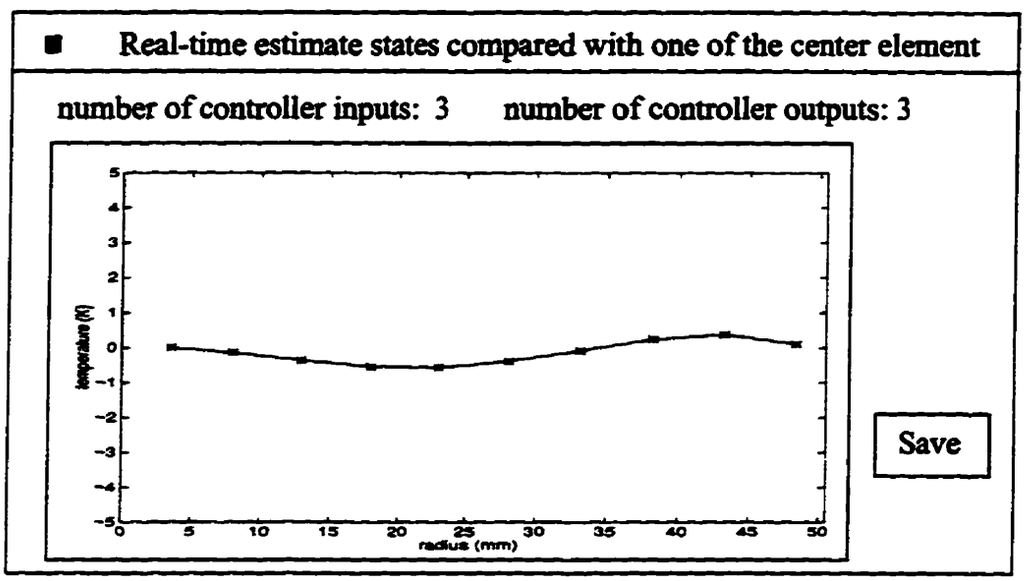


Figure 4.25: Real-time estimate states compared with estimate state of the center element

For P and PI feedback controllers, we can suggest to represent the real-time error temperatures which are the temperatures of subtracting the process setpoints from sensor temperatures.

Figure 4.26 shows the real-time open-loop lamp powers and total lamp powers (which are the additions of the open-loop lamp powers and the adjusted feedback lamp powers calculated by the feedback controller). The x-axis is simulation time in seconds and the y-axis is the lamp power in Watts. After each time the controller outputs are updated, these updated lamp powers will be added in the figure. From this window, users also can see the number of controller outputs.

When the simulation is running, all data is stored in buffers and will be replaced

when the next simulation starts. However, this data can be saved to files by clicking the "Save" button at each window after simulation stops. These have been introduced in Section 4.5.1.4.

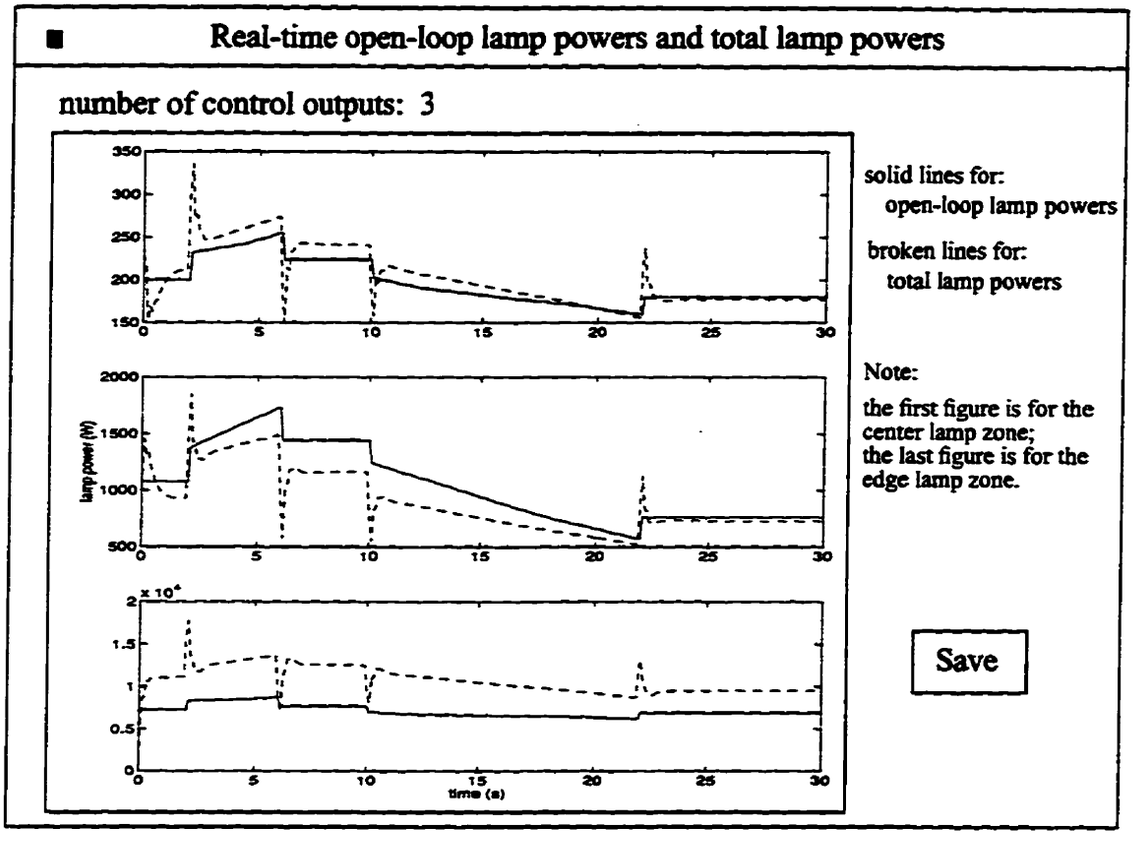


Figure 4.26: An example sub-window for real-time open-loop lamp powers and total lamp powers

Chapter 5 Test and Simulation

The goal of this chapter is to verify the functions and characteristics of the wafer RTP system modelled in Chapter 2, and the control systems designed in Chapter 3. The test mainly includes model test, controller functionality test and controller robustness test.

5.1 Sample System and Test Cases

In this section, we briefly describe a wafer RTP system with multivariable control of a circularly symmetric three lamp zones, which is adapted from [Nor92] and is shown in Figure 2.1. This system will be our sample system and its detail is given as follows.

The wafer is assumed to be perfectly cylindrical and its surface is assumed to be gray and diffuse. The wafer diameter (2R) is 100.8mm, its thickness is 0.675mm and its density is 2.34g/cm^3. The effect of thermal expansion on the wafer dimensions is neglected. The wafer is modelled as collection of integer I concentric annular elements (Figure 2.2). For an example, if $I = 10$, the outer radii of elements - r_i^{out} for i equal to 1 through 10 are: 10.08mm, 20.16mm, ..., 90.72mm, 100.8mm. The interior of the chamber is assumed to be axisymmetric and coaxial with the wafer. The inner diameter of the chamber is 260mm, the distance from the chamber floor to the chamber ceiling is 150mm, and the distance from the wafer bottom to the chamber floor is 100mm. The maximum power is 2kW for the inner lamp zone, 10kW for the center lamp zone and 35kW for the outer lamp zone. The inner radius of the center lamp zone is the same as the wafer radius.

The test cases are conducted with the following conditions:

- implement open-loop systems as Figure 2.4 and feedback systems as Figure 3.1
- implement the wafer RTP nonlinear model as Equation (2.46) and the linear model as Equation (2.41)
- implement control systems shown in Figure 3.5. The feedback controllers include P, PI, LQG and LQGI
- use Equation (2.23) to calculate steady-state temperatures when convection exists, and use the same equation without K^{conv} at vacuum situation

A typical process will be like this:

1. let the wafer reach its steady-state temperatures at vacuum or convection situation
2. reduce or increase the lamp powers and start the simulation

Simulation tests have been carried out using the software designed in Chapter 4 within Unix environment and a Simulink tool box. (Simulink is a computer simulation environment which is part of the Matlab software package supplied by MathWorks Inc.).

5.2 Modeling Test

In this section, the test will be done with an open-loop situation, thus the feedback controller is not linked. We will test:

- steady-state behaviors when the numbers of wafer elements are different
- steady-state behaviors when the initial element temperatures are different
- steady-state behaviors with and without convection
- model linearization

- estimate states

5.2.1 Steady-State Behaviors when the Numbers of Wafer Elements are Different

No matter by how many elements we divide the wafer, there should be almost no effect to the steady-state temperatures under the same lamp powers. One example is shown below.

An example of such a wafer model is shown in Table 5.1. We have named this sample System A.

Table 5.1: Parameters for wafer modeling

Parameter	Unit	System A
τ : time delay coefficient for lag factor	second	0.5
ε_i : average total emissivity over exposed surface of element i	none	0.6
$k_{i,i-1}$ and $k_{i,i+1}$: thermal conductivity at the inner and outer boundaries of element i	$W/(cmK)$	0.3
C_p : specific heat	$J/(gK)$	0.92
h_i : convective heat loss coefficient for element i	$W/\left(m^2K\right)$	$7.1 + 4.3\left(\frac{r_i^{cent}}{R}\right)^4$
$*L$: fraction matrix	none	matrix L

$*L$ is a 10-by-3 fraction matrix (Equation (2.46)) in System A. Its element values are the coefficients for radiative heat transfer from lamps to wafer elements. These values depend on the number of lamp zones, lamp zone positions, and number of wafer elements. We use L to represent the matrix value of L. The range of its element values is from 0 to 1.

We then divide the wafer into 8, 10, 12 elements. The center radius for each element

is shown in Table 5.2.

Table 5.2: Center radiuses for wafer elements

Element Number	Wafer Broken into I elements The center radius for each elements (mm)		
	I=8	I=10	I=12
1	4.4901	3.5921	3.2655
2	10.0402	8.0322	7.3020
3	16.1894	12.9515	11.7741
4	22.4506	17.9605	16.3277
5	28.7508	23.0007	20.9097
6	35.0690	28.0552	25.5047
7	41.3969	33.1175	30.1069
8	47.7307	38.1846	34.7132
9		43.2546	39.3224
10		48.3268	43.9335
11			47.3504
12			49.6589

We apply the lamp power settings at 200.3710W for the inner lamp zone, 1078.080W for the middle lamp zone and 7290.520W for the edge lamp zone.

We will show the steady-state temperature trajectories as Figure 5.1 and the steady-state temperature at the "center radius" of each wafer element as Table 5.3.

Table 5.4 is an example of how the heat flows inside the wafer elements. In this example, the wafer is broken into eight elements.

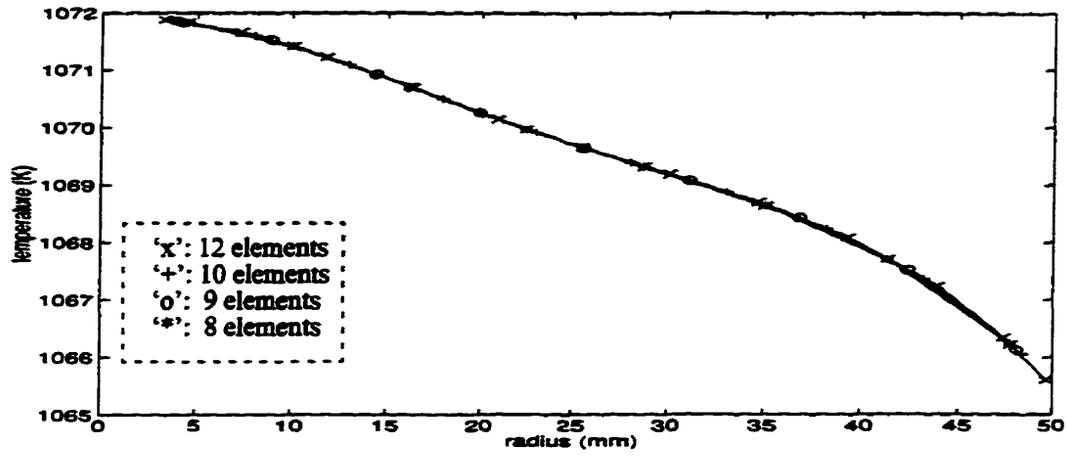


Figure 5.1 : Steady-state temperatures across wafer

Table 5.3: Steady-state temperatures for wafer elements

Element Number	Temperature at different elements (K)		
	I=8	I=10	I=12
1	1071.828	1071.873	1071.888
2	1071.416	1071.602	1071.662
3	1070.709	1071.094	1071.227
4	1069.970	1070.492	1070.690
5	1069.314	1069.911	1070.147
6	1068.641	1069.387	1069.650
7	1067.709	1068.869	1069.190
8	1066.227	1068.239	1068.701
9		1067.354	1068.086
10		1066.052	1067.237
11			1066.342
12			1065.605

Table 5.4: Heat flows inside of each element

Element Number	Radiation (W)		Conduction (W)		Total heat flow q (W)
	Absorbed by wafer	Emitted by wafer	from $i-1$ to i	from $i-1$ to i	
1	11.4353	-11.3753	0	-0.0600	0
2	34.1993	-34.0734	0.0600	-0.1858	0.0001
3	56.7396	-56.6393	0.1858	-0.2861	0
4	79.1268	-79.0762	0.2861	-0.3367	0
5	101.5137	-101.4202	0.3367	-0.4302	0
6	123.9301	-123.6463	0.4302	-0.7141	-0.0001
7	146.2272	-145.6183	0.7141	-1.3230	0
8	175.2402	-176.5632	1.3230	0	0

From the above table, it is easy to see that the total heat flow is approximately equal to zero for each element, when the final temperature is reached. As a result, these temperatures are the steady-state temperatures.

For the following test examples, without extra explanation, we will set the wafer model as System A and divide the wafer into 10 elements.

5.2.2 Steady-State Behaviors when Initial Temperatures are Different

Under the same lamp powers and the same RTP environment, regardless of the initial temperatures, within several minutes, the wafer always reaches its steady-state temperatures and these temperatures should always be the same. Figure 5.2 is one of the examples which shows the temperature trajectories of the center and the edge elements when the lamp

powers are set to reach the steady-state temperature of 1070K at convection situation.

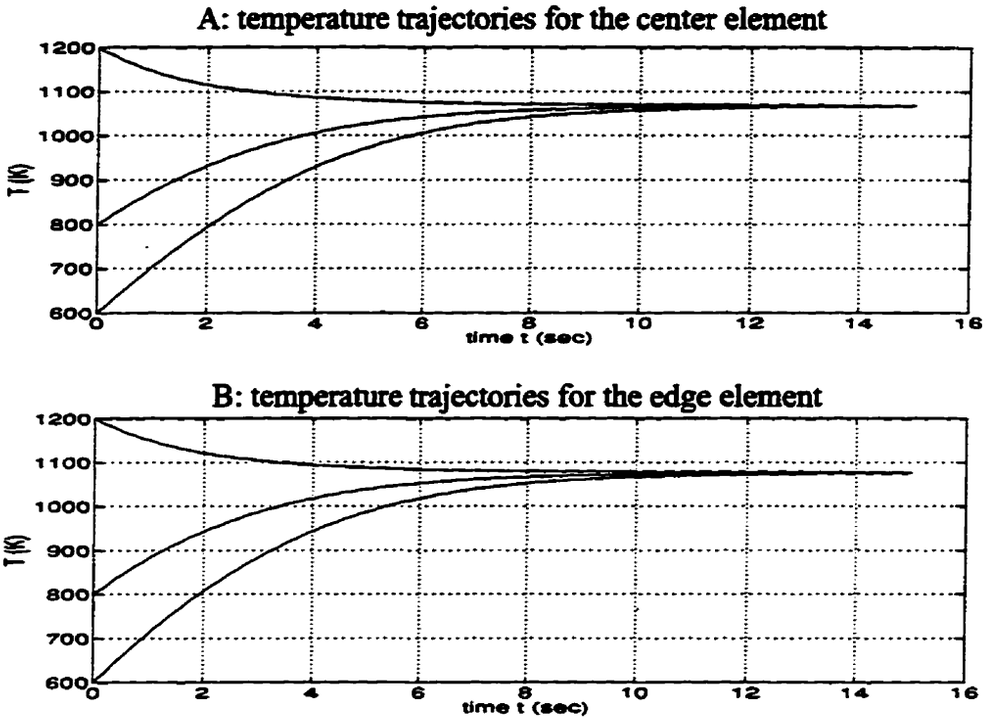


Figure 5.2 : Wafer temperature trajectories with different initial temperatures

5.2.3 Steady-State Behaviors with and without Convection

Under the same lamp powers, with and without convection, the wafer steady-state temperatures have large differences. The test process is described as follows.

First, at vacuum situation, we apply lamp powers to let the wafer reach its steady-state temperature profile near 1070K. Remember this lamp powers as a vector value P_{ini}^{v}. Second, at convection situation, we apply lamp powers to let the wafer reach its steady-state temperature profile, also near 1070K. (Assume the gas temperature is close to room temperature, 300K). Finally, we change the lamp powers to P_{ini}^{v} to start the simulation.

The process temperature trajectories for wafer elements are shown at Figure 5.3; the initial and final steady-state temperatures at "center radius" for each element are shown as asterisks (*) in Figure 5.4.

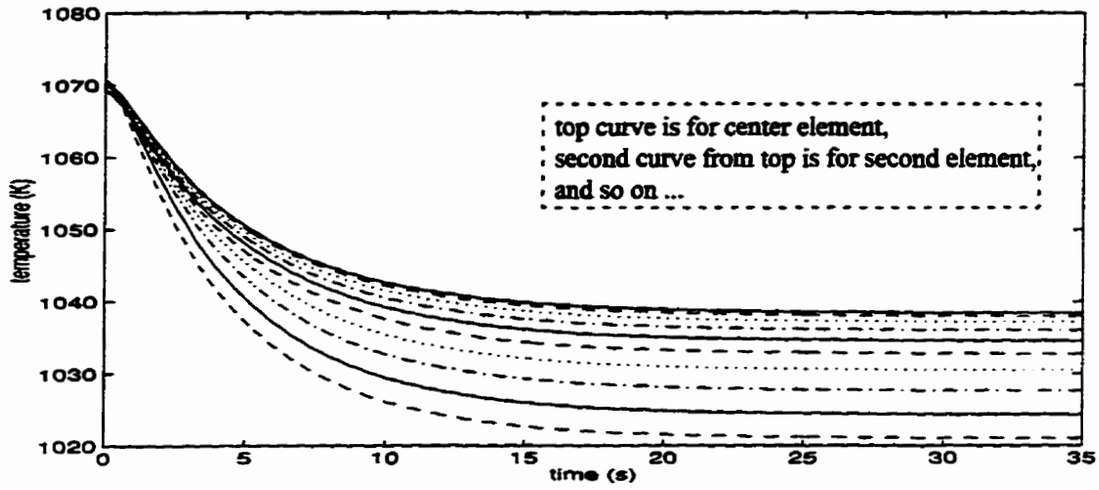


Figure 5.3 : Wafer temperature trajectories for the nonlinear model (I)

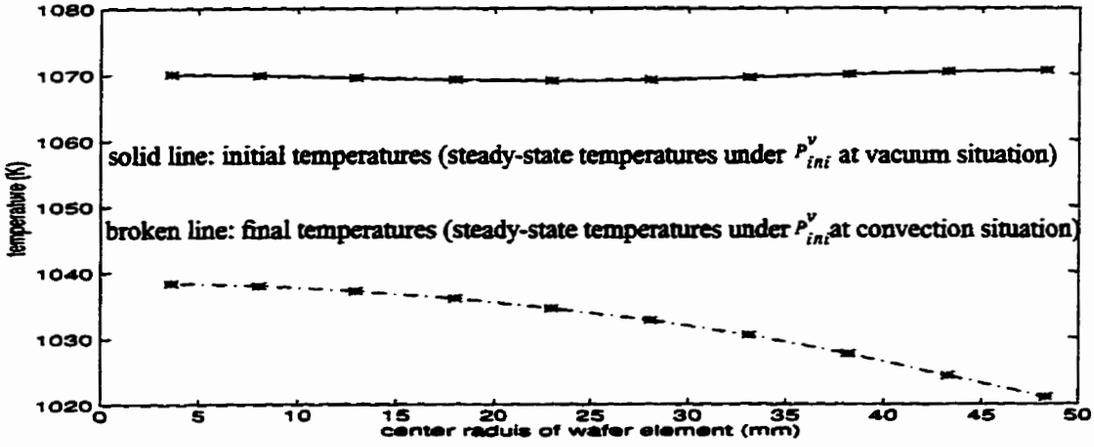


Figure 5.4 : Initial and final steady-state temperatures at center radiuses of wafer elements

From Figure 5.3, we find that although the initial temperatures are near uniformity, the non-uniformity is obvious when the steady-state temperatures are reached. From Figure 5.4, we notice that when we apply the lamp powers which cause the wafer to reach its

steady-state temperature profile near 1070K during vacuum situation, to the wafer during convection situation, the temperatures across the wafer drop significantly, especially the edge element. The temperature difference between initial and final situations is about 32K for the center element and is about 49K for the edge element.

This test shows that convection affects the edge of the wafer much more than the center of the wafer. Therefore, the edge heat loss is one main reason for the non-uniformity across the wafer during the process. It is also one of the reasons why we have to use the feedback controller.

5.2.4 Model Linearization Test

In this subsection, we test the linear model behaviors to see if it is close to the non-linear model.

For the linear model, we apply the nominal lamp powers to let the nominal temperature reach close to 1070K by Equation (2.23). These nominal lamp powers are 223.3W for the inner lamp zone, 857.6W for the middle lamp zone and 11560.7W for the edge lamp zone. Note that for all the test cases which will use the linear model, only this one nominal temperature and these corresponding lamp powers will be used. This is because we want to show that the controllers will be able to accommodate a wide range of process temperatures and only need one set of nominal signals.

We apply the same initial conditions and the same process to the linear wafer model as to the nonlinear wafer model in Section 5.2.3. The wafer temperature differences between these two models are represented in Figure 5.5. (The wafer temperature trajectories for the nonlinear model is shown in Figure 5.3.)

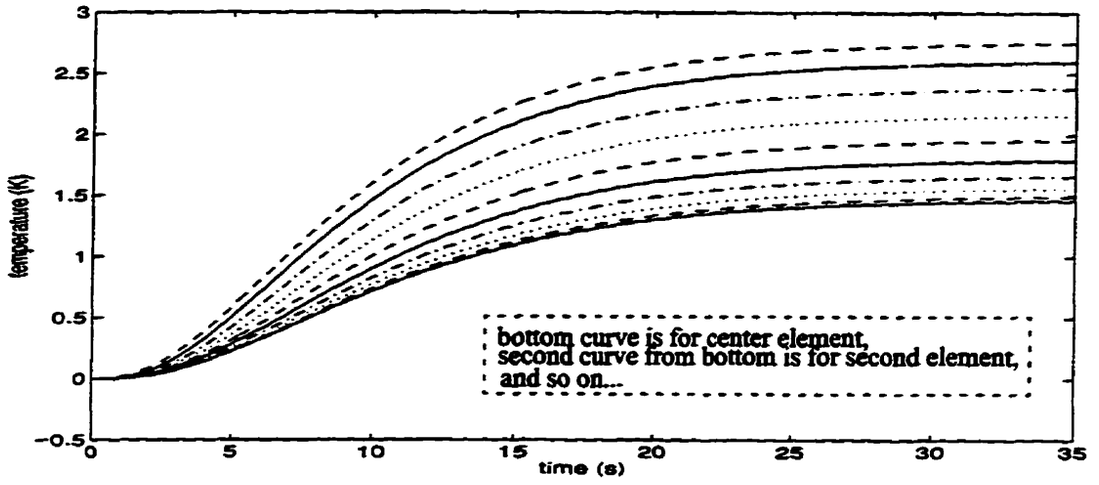


Figure 5.5 : Subtraction of element temperatures of the nonlinear model from element temperatures of the linear model (I)

From Figure 5.5, we could see that at the beginning of simulation, the worst case temperature difference between the linear model and the nonlinear model is within 1K. At the end of the simulation, the worst case has increased to 3K. This behavior is reasonable because the linearization is done at the nominal temperature point (1070K).

If we apply lamp power signals, which keep wafer temperatures around nominal temperature (1070K), the differences between these two models will be much less than the results we had in Figure 5.5. Suppose lamp power is set as 230W for the inner lamp zone, 870W for the middle lamp zone, and 11600W for the edge lamp zone, we apply the same initial conditions in Section 5.2.3 to the nonlinear and linear wafer models. The simulation results during the convection situation (gas temperature is 300K) for the nonlinear model is shown in Figure 5.6 and the wafer temperature differences between these two models are represented in Figure 5.7.

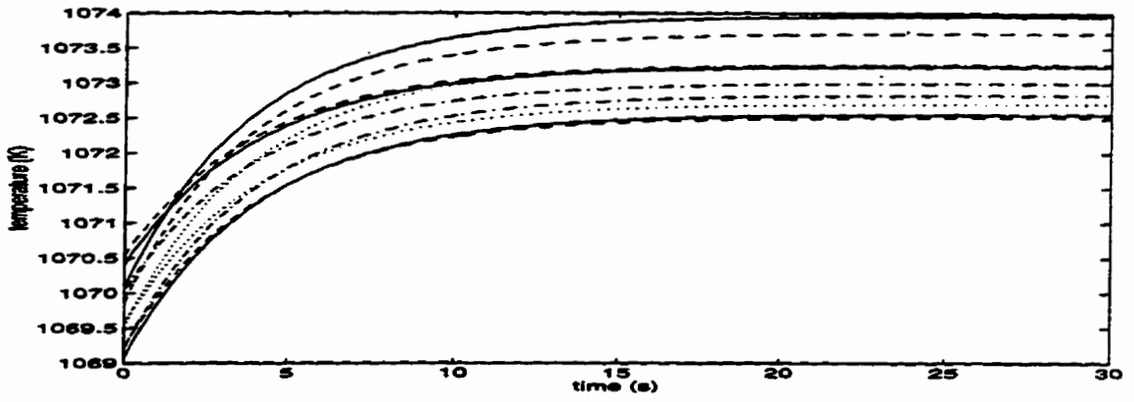


Figure 5.6 : Wafer temperature trajectories for the nonlinear model (II)

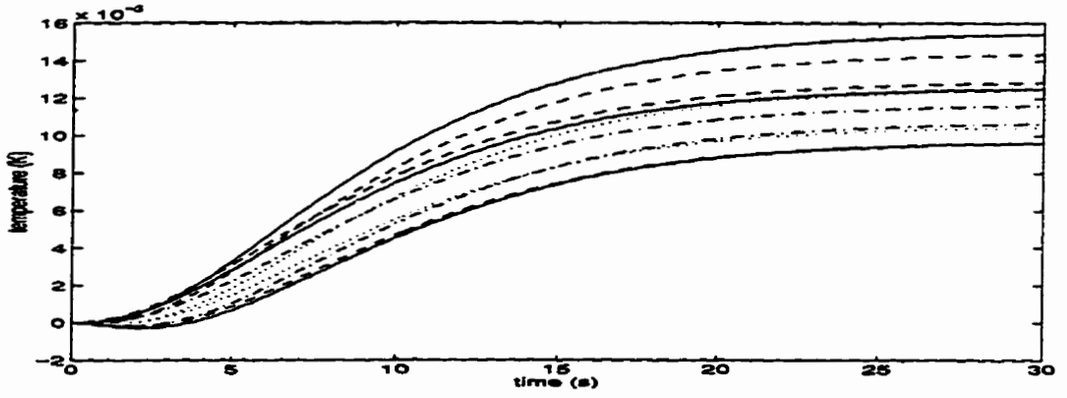


Figure 5.7 : Subtraction of element temperatures of the nonlinear model from element temperatures of the linear model (II)

From these two simulations, we can see that it is reasonable to design the feedback controllers by using the linear model.

5.2.5 Estimate States

In this section, we would like to see the information about the estimate states generated by LQG and LQGI feedback controllers. Since we only care about the wafer element temperatures, the estimate states of lagged lamp powers will be neglected.

To generate our sample figure as shown in Figure 5.8. First, we designed an estimate gain matrix according to the linear model used as the plant. This linear model has the same number of wafer elements as the nonlinear model used as the emulator in order that we can compare the estimate states with the element temperatures one by one. We use the T in Equation (2.46) as the element temperatures and the last I outputs of $\bar{x}$ in Figure 3.13 plus nominal temperature (1070K) as the estimate states. This estimator gain matrix has been selected as the best scenario to perform the best result. Second, we apply the same initial temperatures and the same process as described in Section 5.2.3.

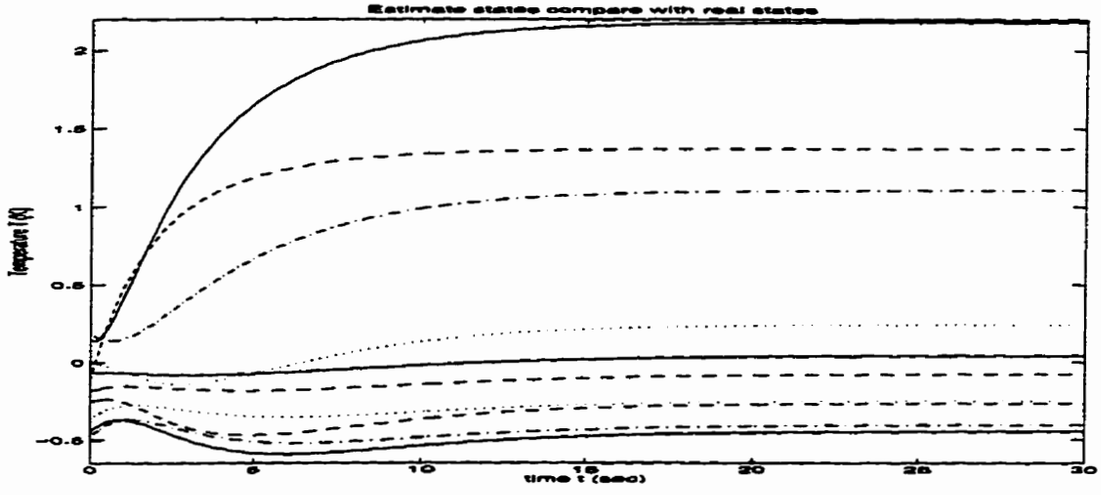


Figure 5.8 : Subtraction of element temperatures from estimate states

From Figure 5.8, we can tell that the largest error between the element temperature and the estimate state is around 2.2K. The error is quite large because the non-uniformity of temperatures at this situation is also very large. (See Figure 5.3). These errors will be reduced in a feedback loop when the non-uniformity of temperatures is reduced. This result shows that the real-time wafer temperatures estimated by the controller are reasonably close to the wafer temperatures of the emulator.

5. set the powers of the lamp zones to an ideal condition corresponding to a steady-state temperature of 1120K.

Process C

1. hold at the preheat temperature 1070K for time 2 seconds.

2. ramp at a rate of 12.5K/s to the process temperature 1120K.

3. hold at the process temperature for 4 seconds.

4. ramp down at a rate of 8.33K/s and terminate upon reaching temperature 1020K.

5. set the powers of the lamp zones to an ideal condition corresponding to a steady-state temperature of 1020K.

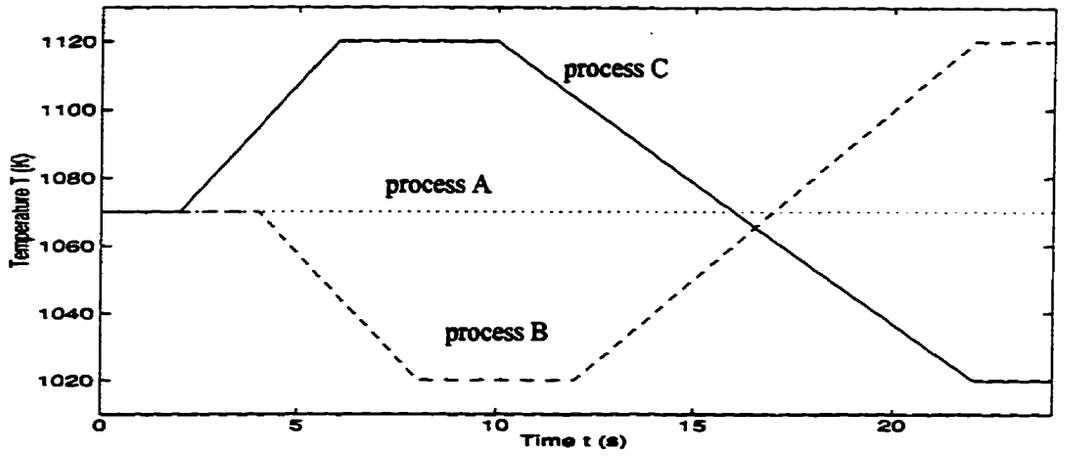


Figure 5.9 : Process temperature trajectories

5.3.2 Open-Loop Lamp Powers

The details on how to calculate the open-loop lamp powers are provided in Section 3.4. Equation (3.6) shows that open-loop lamp powers consist of two parts: P^{ss} which balances the loss of the heat from the wafer by radiation and convection, and P^{dyn} which adds enough additional heat to obtain a rate of temperature increase close to the derivative of the

process temperatures at all points on the wafer.

In order to test how the feedback controllers work, for all our test cases, we provide the computations of P^{ss} under the condition without convection, which means P^{ss} only balances the loss of the heat from the wafer by radiation. The pre-computed table of values of P^{ss} at a grid of temperature space of 50K from 800K to 1500K is achieved by Equation (2.23) without K^{conv}. This is shown in Figure 5.10, where the lamp settings are displayed as fractions of maximum power and plotted using a log scale (base 10). This is the only table we used to calculate P^{ss} for all three processes.

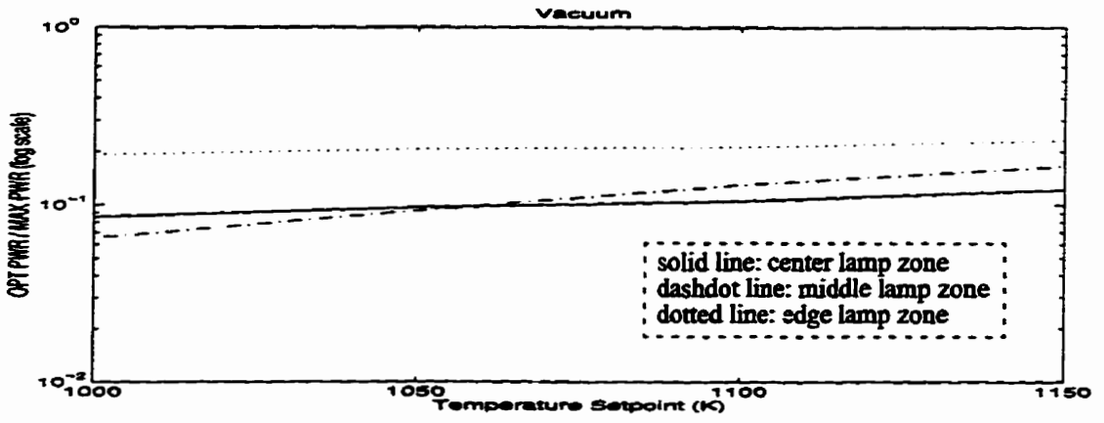


Figure 5.10 : Lamp powers balance the loss of the heat from the wafer by radiation

P^{dyn} is calculated by Equation (3.8), where $T^{des}(k)$ is the process temperature at kth sampling instant. In our test cases in Section 5.3.3, Section 5.3.4 and Section 5.3.5, P^{dyn} will be the same for control systems with different feedback controllers if the process temperature trajectories are the same. However they will be different if the process temperature trajectories are different.

5.3.3 P and PI Controllers

This section shows the simulation results when the P and PI feedback controllers are used in the control system. For the system which has P controller (Figure 3.7), the feedback gain is set as $K_P = diag(11, 40, 800)$.

First, during convection situation, we apply lamp powers to let the wafer reach its steady-state temperature profile near 1070K (suppose the gas temperature is 300K), then we change the lamp powers to corresponding P^{ss} (which causes the wafer to reach its steady-state temperature profile near 1070K during vacuum situation) to start the simulation. The simulation results are shown in Figure 5.11 and Figure 5.12.

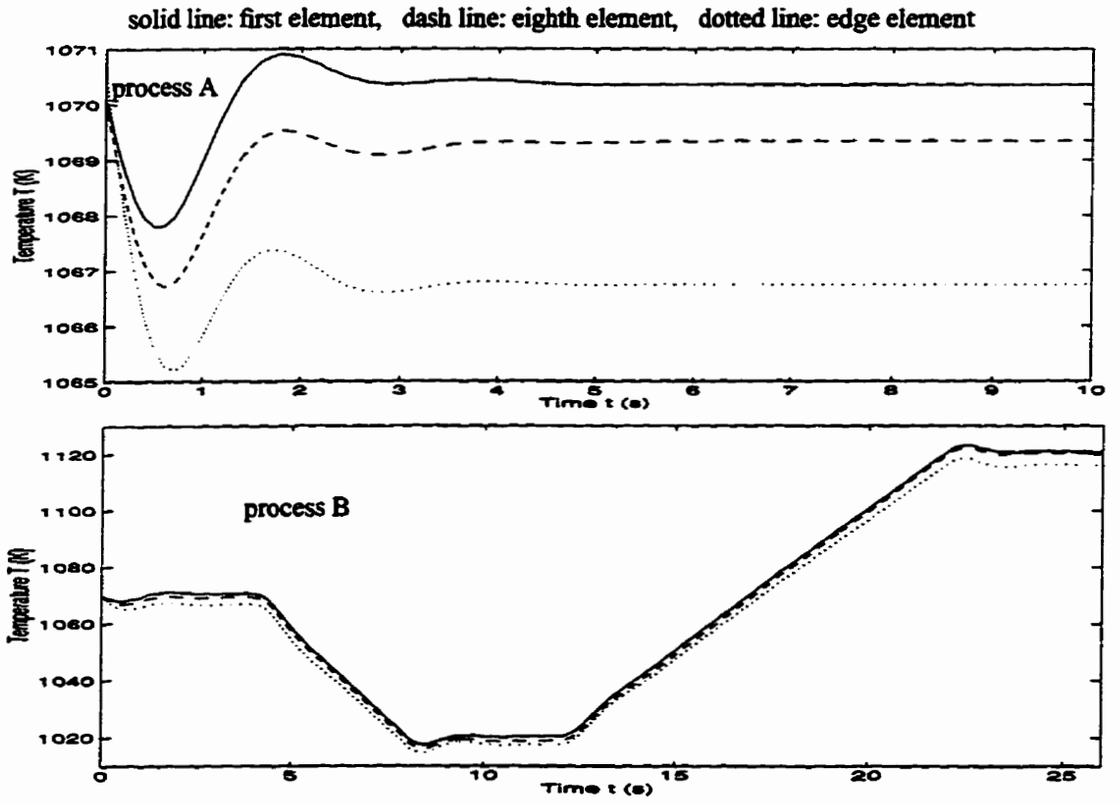


Figure 5.11 : Wafer temperature trajectories generated from Process A and B (P controller)

From Figure 5.11, we can tell that in process A, the non-uniformity of the wafer temperatures is about 3.6K and the overshoots is about 0.9K for the center element. (Note that the overshoot here is not the classical definition of overshoot. The overshoot is defined as the largest excess values during the flat part of a specific process trajectory. The overshoot mentioned in later sections is all similarly defined). The wafer temperatures could not reach the close uniformity around 1070K, especially the edge element.

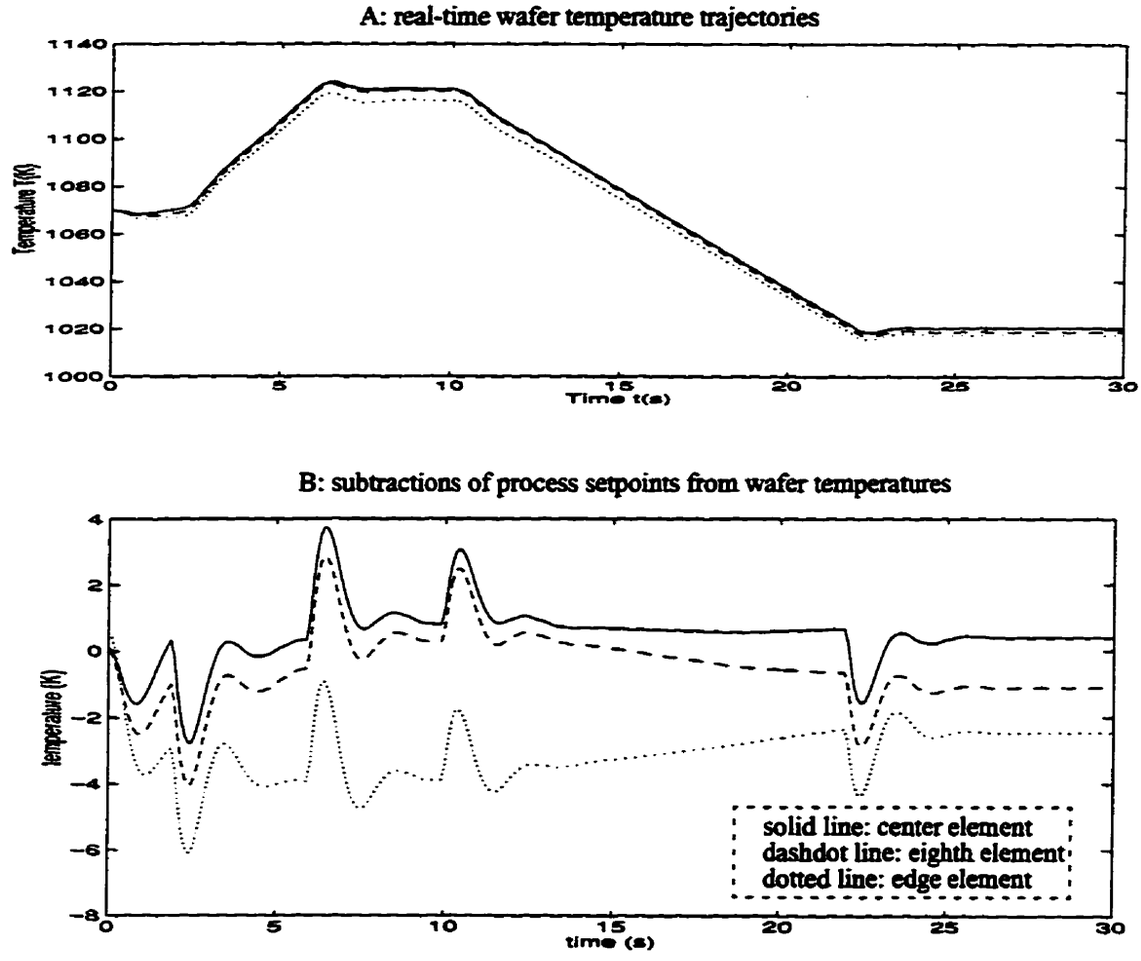
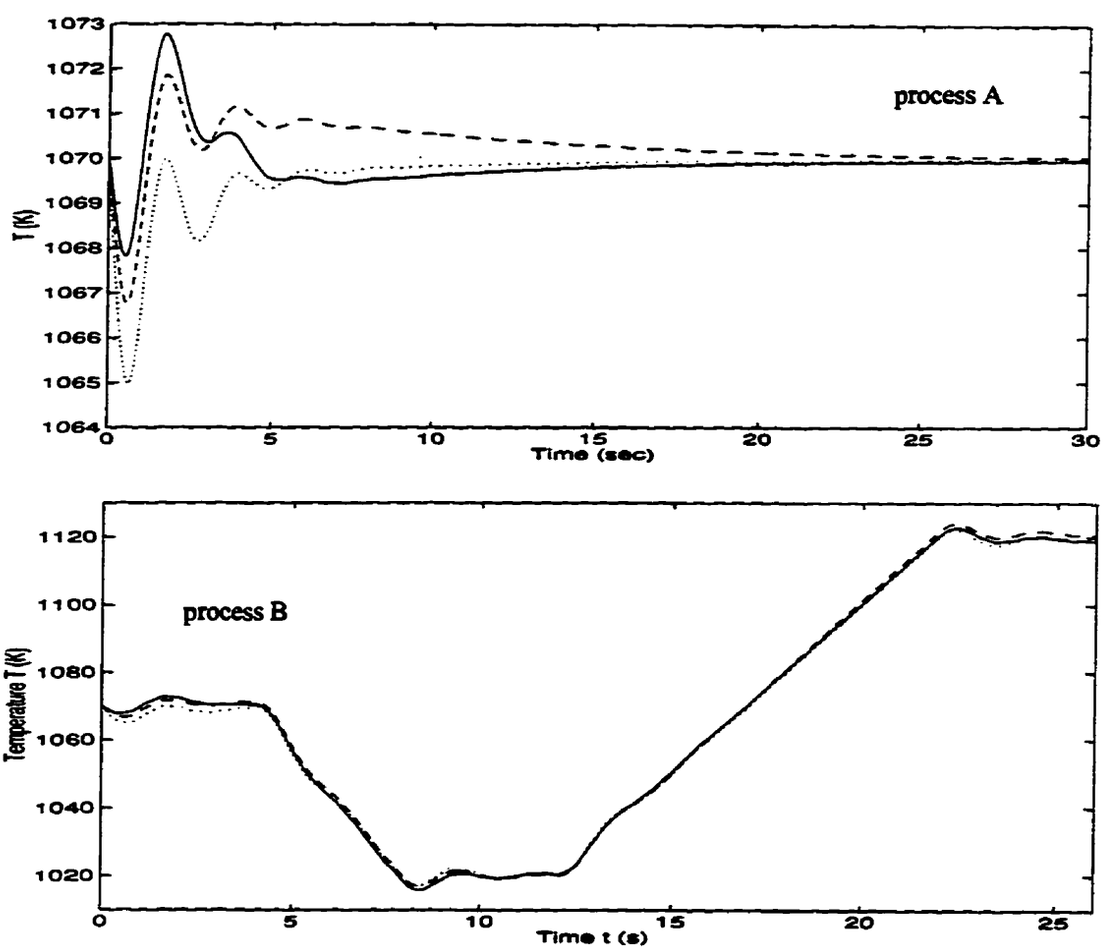


Figure 5.12 : Simulation results for process C (P controller)

From Figure 5.12, we can tell that in process C, the non-uniformity of the wafer temperatures is about 5.1K at ramping up period, 4.8K at process temperature in steady-state of

1120K, 4.1K during ramping down to terminate temperature, and it is about 2K at the end of simulation.

For the system which has the PI controller (Figure 3.11), the feedback gain is the same as the P controller. The reset time is selected as $\tau_{PI} = 2$. The simulation results are shown in Figure 5.13 and Figure 5.14.



solid line: first element, dash line: eighth element, dotted line: edge element

Figure 5.13 : Wafer temperature trajectories generated from Process A and B (PI controller)

From Figure 5.13, we notice that in Process A, the biggest overshoot in the three elements is about 3K. Although after 30 seconds, the non-uniformity of the wafer temperatures

is less than 0.1K, the wafer temperatures have large vibrations at the beginning of simulation.

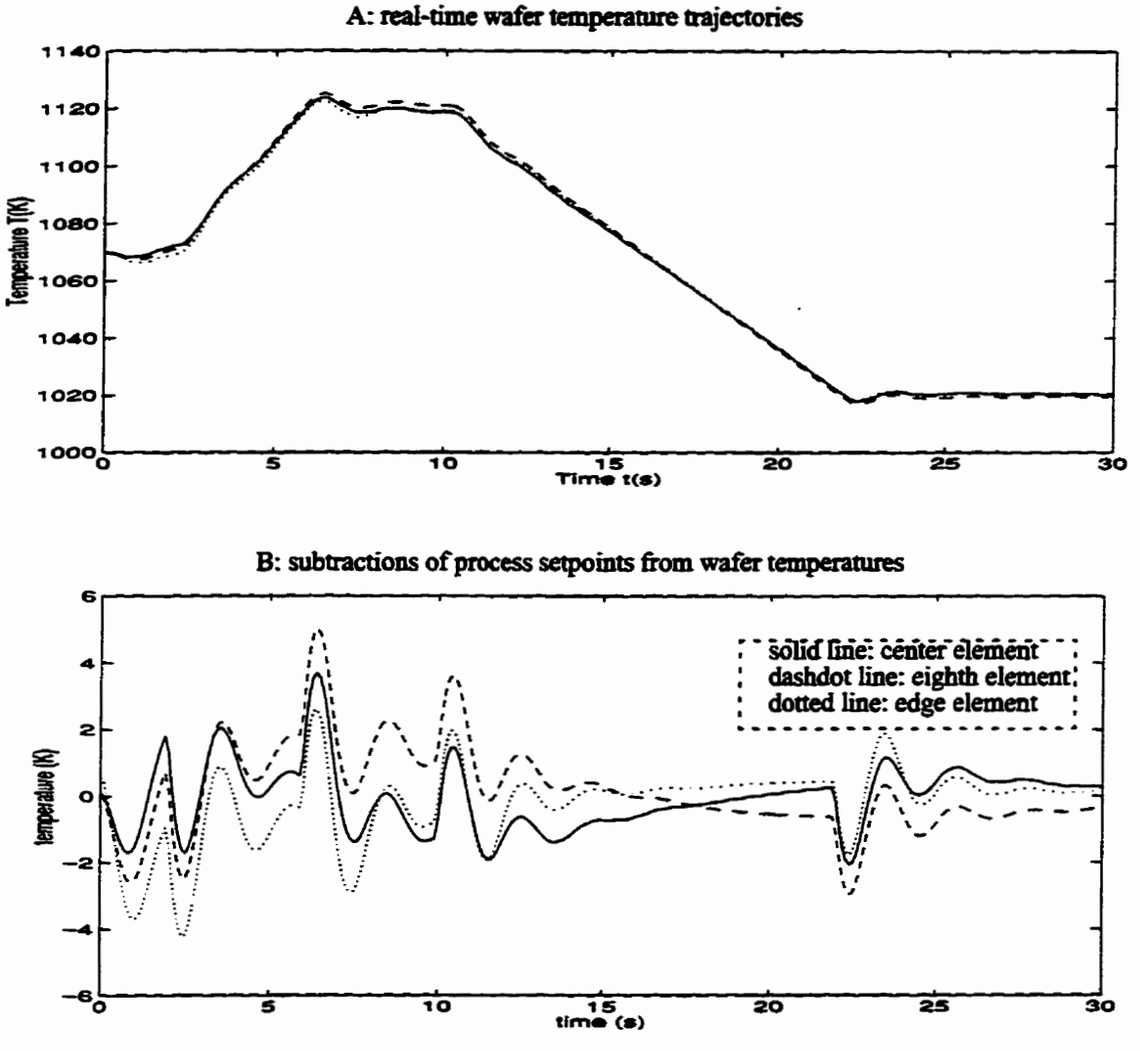


Figure 5.14 : Simulation results for process C (PI controller)

From Figure 5.14, we find that in Process C, the temperature vibrations are much larger than in Process A.

Overall, for our design cases, these two control systems were not functioning well. This is because the close uniformity (within 1K) was not achieved and the wafer temperatures had large vibrations during the processes.

The simulation results may be better, if we design a proportional-integral-derivative feedback controller to provide simultaneous improvement in transient and steady-state responses. However, these are beyond the area studied in this thesis.

5.3.4 LQG Controller

With LQG feedback controller (Figure 3.14), we repeat three processes with the same initial conditions in Section 5.3.3. The weighting matrices of LQG are shown as follows.

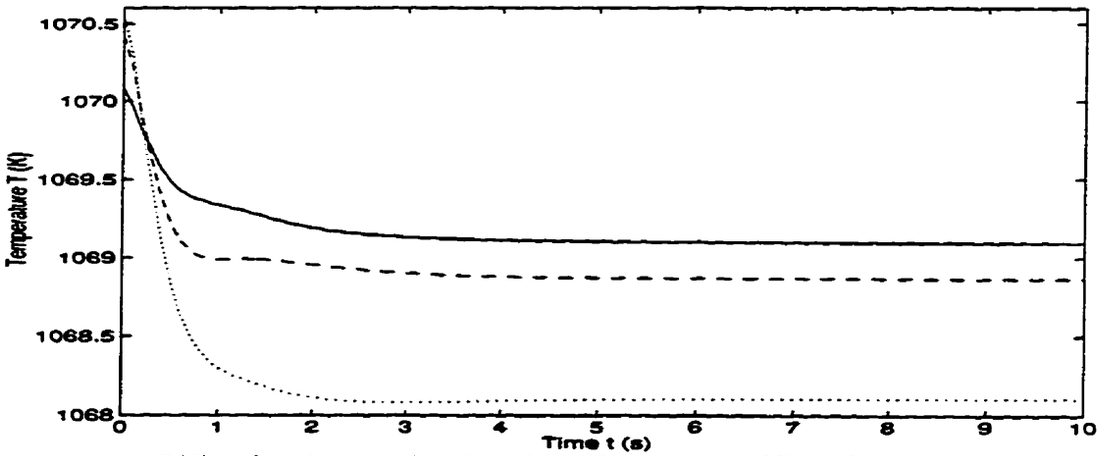
$$Q = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 10} \\ 0_{10\times 3} & 10\,(I_{10} + 10000M) \end{bmatrix},$$

where M can be calculated according to Equation (3.29),

$$R = diag\,(10.8,\ 1.5,\ 0.01),$$

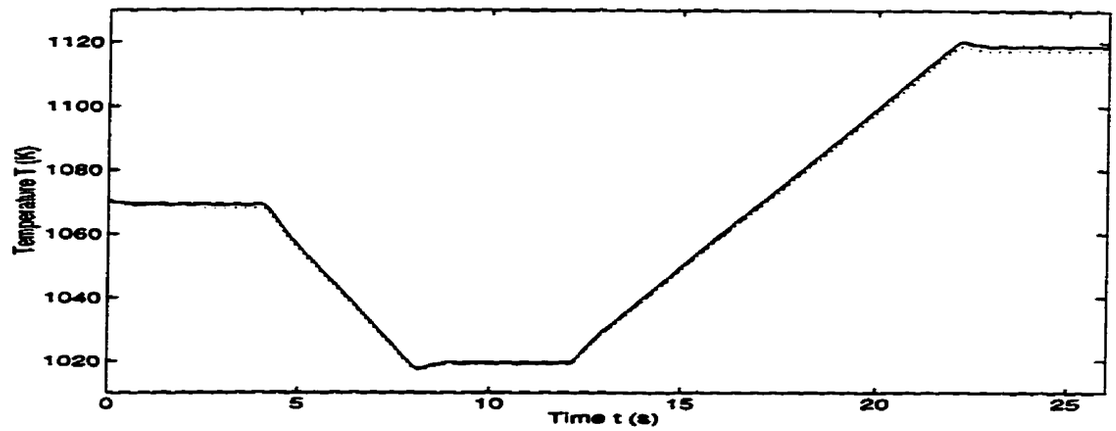
$$W = diag\,(120,\ 2200,\ 80000), \text{ and } V = 0.0001 I_3 .$$

The simulation results are shown as Figure 5.15, Figure 5.16 and Figure 5.17.



solid line: first element, dash line: eighth element, dotted line: edge element

Figure 5.15 : Wafer temperature trajectories for process A (LQG controller)



solid line: first element, dash line: eighth element, dotted line: edge element

Figure 5.16 : Wafer temperature trajectories for process B (LQG controller)

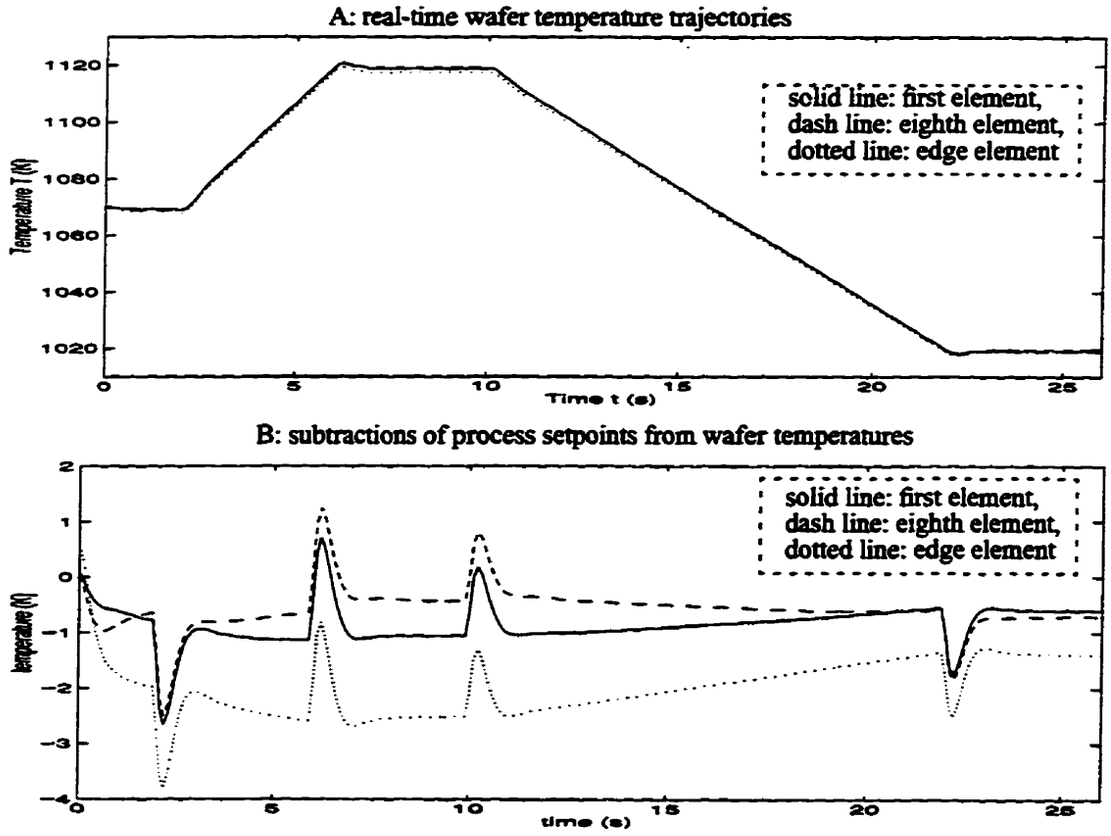


Figure 5.17 : Simulation results for process C (LQG controller)

From these figures, we see that the temperature trajectories are smooth and the temperature vibrations are small. However, we notice that the non-uniformity of the element temperatures is still obvious during the processes. The non-uniformity of the temperatures in process A is about 1K and in process C it is about 2K.

5.3.5 LQGI controller

Finally, for the same processes and same initial conditions as we set in Section 5.3.3, we use the LQGI as the feedback controller (Figure 3.16). In the wafer RTP, the deviation between the desired temperature and the temperature in the edge element is usually the largest compared to other elements because of the edge heat loss. In this case, we measure the edge element and set it as the last output from the plant. We select this edge element (10th element) as the integral output, and the weighting matrices are chosen as follows.

$$Q_I = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 11} \\ 0_{11\times 3} & 10\,(I_{11} + 10000 M_I) \end{bmatrix},$$

where M_I can be calculated according to Equation (3.33).

$$R_I = diag([9.8, 0.6, 0.005]),$$

$$W_I = diag([120, 2200, 8000]), \text{ and } V_I = 0.00001 I_4.$$

The simulation results are displayed in Figure 5.18, Figure 5.19 and Figure 5.20. From these figures, we can tell that the LQGI controller is functioning well. For process A, in Figure 5.18, we can tell that the non-uniformity of the wafer temperatures is about 0.6K. For process C, in Figure 5.20, we notice that the non-uniformity during the whole process is quite small (within 1K). At several points, the differences between setpoints and wafer tem-

peratures are large (around 1.8K). This situation is reasonable because the process temperature trajectory has very sharp corners at these points, and we have used lag factor to slow down the changes of lamp powers.

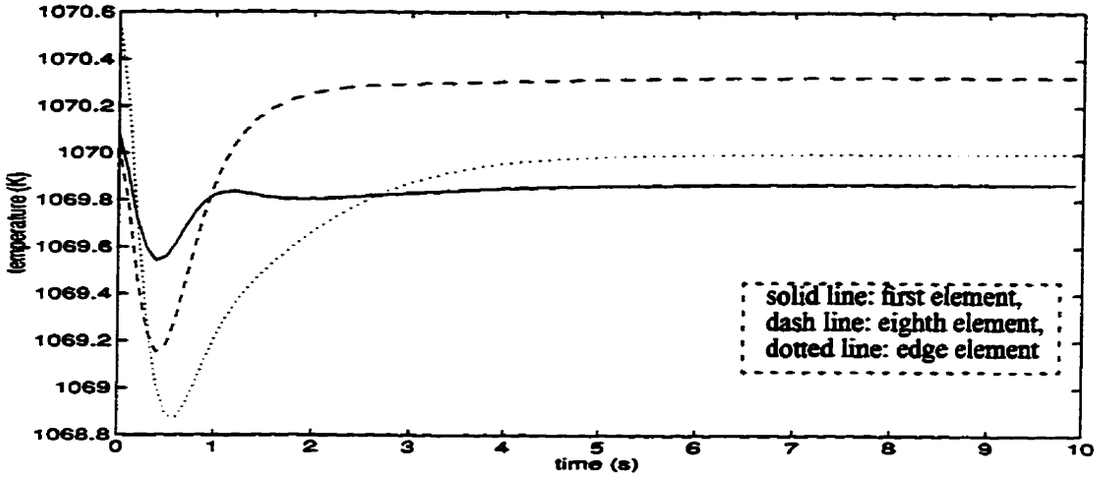


Figure 5.18 : Wafer temperature trajectories for process A (LQGI controller)

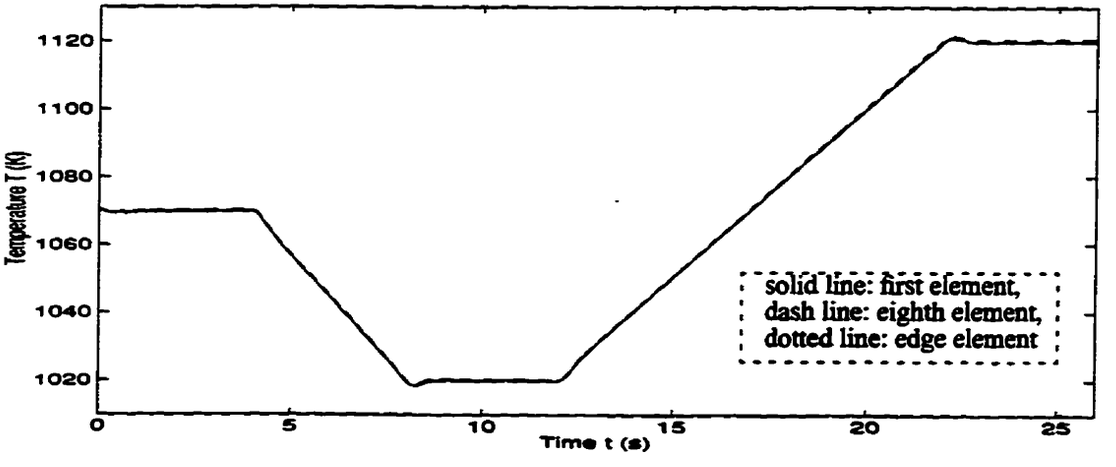


Figure 5.19 : Wafer temperature trajectories for process B (LQGI controller)

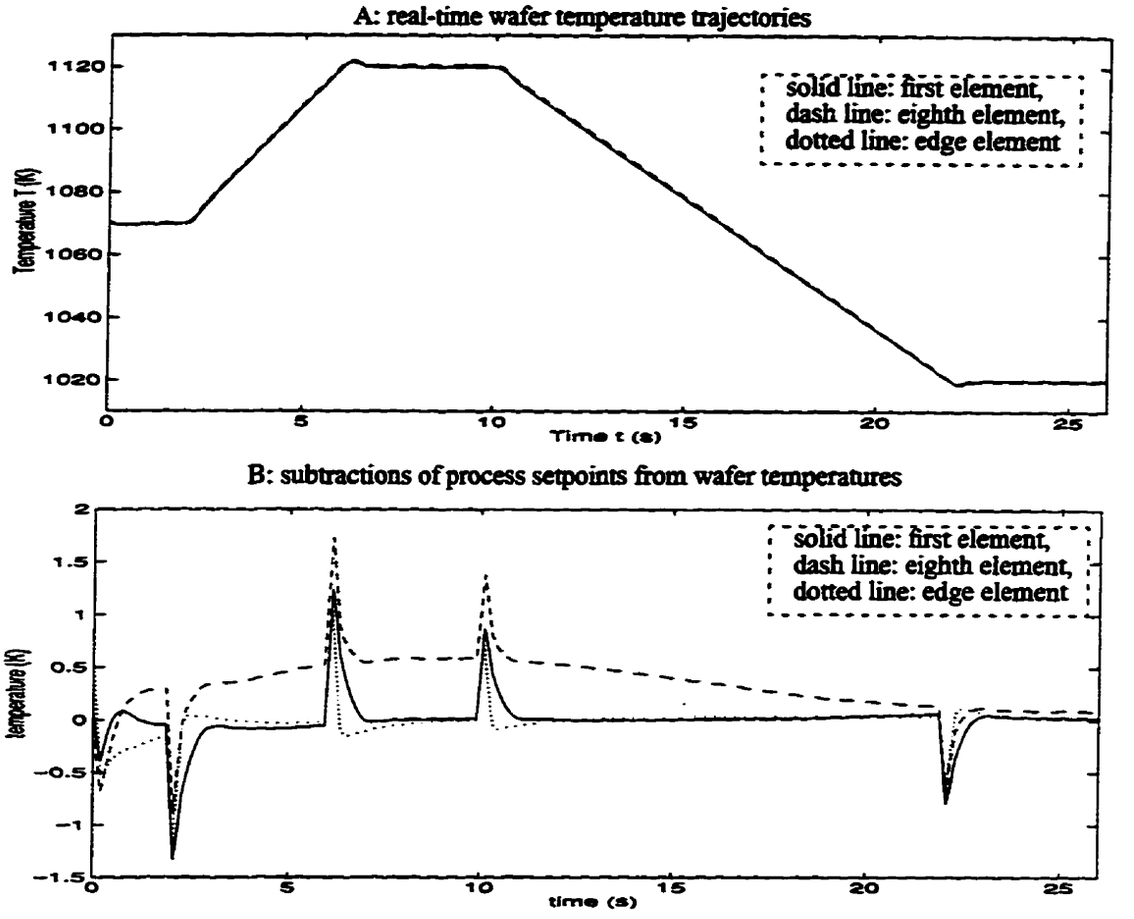


Figure 5.20 : Simulation results for process C (LQGI controller)

5.3.6 Results Compared

In this section, we give details about comparing the simulation results generated in Section 5.3.3, Section 5.3.4 and Section 5.3.5.

Table 5.5 shows the non-uniformity and overshooting values. The numbers in this table are approximate values (Kelvin degrees) of the worst cases among three sensor signals generated from process A and C.

Table 5.5: ...

tempe... non-un...	stead...	
	stead...	
	ra...	
	...	
tempe... overs...	ste...	
	...ea...	

From the above ... reached wh... the contr... shoot and ...ation ... controller, th... temperature ... quite large ... obvious ... troller has ...hieved the ... example pr...

5.4 Robustness ...

If ... system ... temperatur... may reach ... model has ... errors ... be the sam... all times. In... cally adjus... the lamp pow...

Table 5.5: List of wafer temperature non-uniformity and overshooting

		LQGI	LQG	PI	P
temperature non-uniformity	steady (Process A)	0.6	0.9	3	3.6
	steady (Process C)	0.6	2.0	2.1	4.8
	ramp up (Process C)	0.8	2.1	2.3	5.1
	ramp down (Process C)	0.8	2.0	2.2	4.1
temperature overshoot	steady (Process A)	0.3	none	2.9	0.9
	steady (Process C)	1.7	1.2	5.1	3.9

From the above table, we can tell that the wafer temperature uniformity is not reached when the control system has the P feedback controller. The wafer temperature overshoot and vibration are large when the control system has the PI controller. For the LQG controller, the temperature overshoot and vibration is the smallest, but non-uniformity is still quite large. It is obvious that in our test cases, the control system with LQGI feedback controller has achieved the best result. This control system functions well during these three example processes.

5.4 Robustness Test

If the system environment of the open-loop wafer RTP has changed, the element temperatures may reach different trajectories, but it is quite possible that our wafer RTP model has slight errors for some processes, since the conditions inside the chamber may not be the same at all times. In this section, we will show that our control system will automatically adjust the lamp powers to let the wafer reach its temperature near uniformity and fol-

low the process trajectory although the wafer model has slight errors.

Model mismatch is simulated by re-selecting the values of three parameters defined in Table 5.1. These values describe the wafer RTP environment and they are usually set by the user's experience, which could be different from the actual system. Therefore, in Table 5.6, we list several possibilities to show the changes of the RTP environment.

Table 5.6: Robustness test cases

Parameter	Unit	System A	Test Cases
τ	second	0.5	0.2
			1
h_i	$W/\left(m^2K\right)$	$7.1 + 4.3\left(\frac{r_i^{cent}}{R}\right)^4$ (H)	H *0.9
			H *1.1
			vacuum ($h_i = 0$)
L	none	10-by-3 matrix L	L* $diag$(0.95, 1, 1)
			L* $diag$(1, 0.95, 1)
			L* $diag$(1, 1, 0.95)

For each test case:

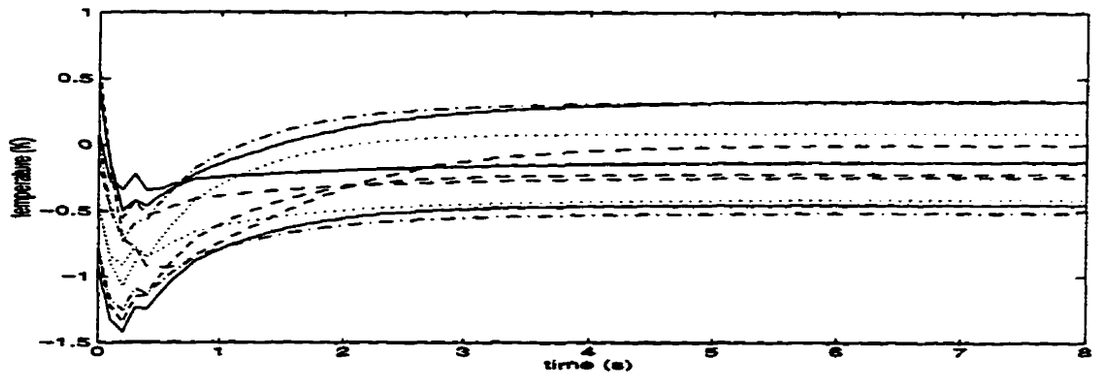
- only one parameter value will be changed as the result of model mismatch
- the LQGI feedback controller described in Section 5.3.5 will be used because we have shown it is the best design in our cases
- Process A will be applied (see Figure 5.9 for the trajectory)
- the initial lamp powers will be set as described in Section 5.3.3.

5.4.1 Different Values of Time Delay Coefficient

The time delay coefficient (τ) describes the situations of the lag factor. (See Section 2.5 for details). Two test cases will be done in this subsection and the values of τ for changed wafer models are shown in Table 5.6 under the "Test Cases" column.

The simulation results are shown in Figure 5.21. From this figure, we can find that the initial steady-state temperatures for these two systems are the same. This is because the time delay coefficient only effects the lag factor. We also notice that at the end of these simulations, the temperature differences between process setpoints and these two systems are quite small (about ±0.8 K).

A: subtraction of process setpoints from temperatures of the system with τ as 0.2



B: subtraction of process setpoints from temperatures of the system with τ as 1

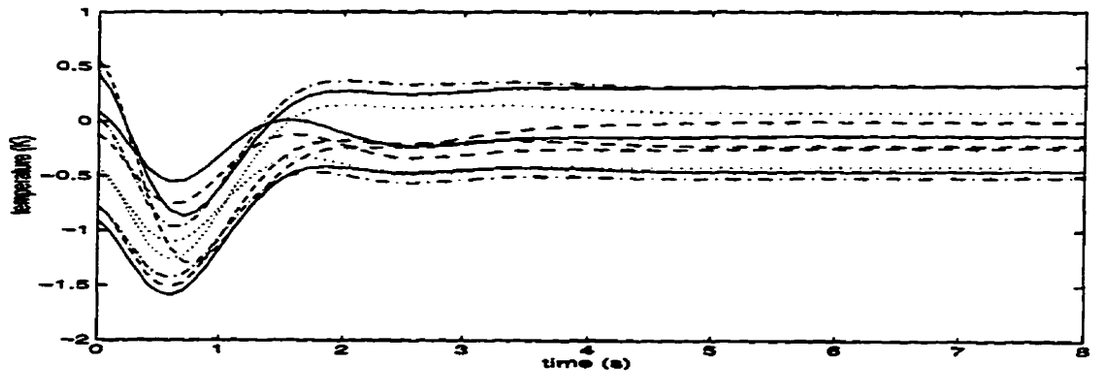


Figure 5.21 : Comparing temperature trajectories with different values of τ

5.4.2 Different Values of Convective Heat Loss Coefficient

The heat loss coefficient (h_i) describes the convective heat transfer between wafer elements and surrounding gas. Three test cases will be done in this subsection and the values of h_i for changed wafer models are shown in Table 5.6 under the "Test Cases" column.

The simulation results are shown in Figure 5.22. From this figure, we can see that the initial steady-state temperatures for these systems are all different. Especially for the third system (vacuum situation), the initial temperatures are much higher than the setpoints. However, we can notice that at the end of the simulations, the temperature differences between process setpoints and these three systems are reduced to about ±0.8 K.

5.4.3 Different Values of Fraction Matrix L

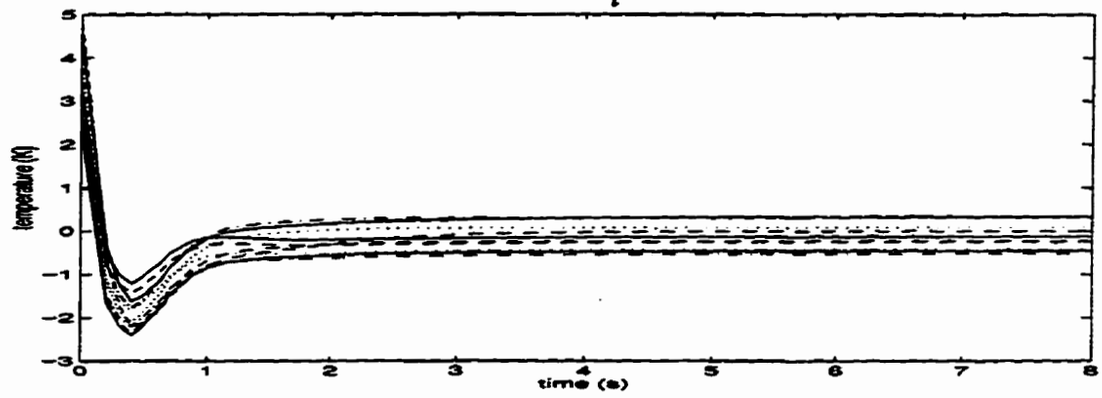
Fraction matrix (L) describes the radiative heat transfer from lamps to wafer elements. Three test cases will be done in this subsection and the values of matrix L of changed wafer models are shown in Table 5.6 under "Test Cases" column.

The simulation results are shown in Figure 5.23. From this figure, we can find that at the end of the simulations, the temperature differences between process setpoints and these three systems are within ±0.8 K although the errors are quit large (from –8.5 K to 2K) at the beginning.

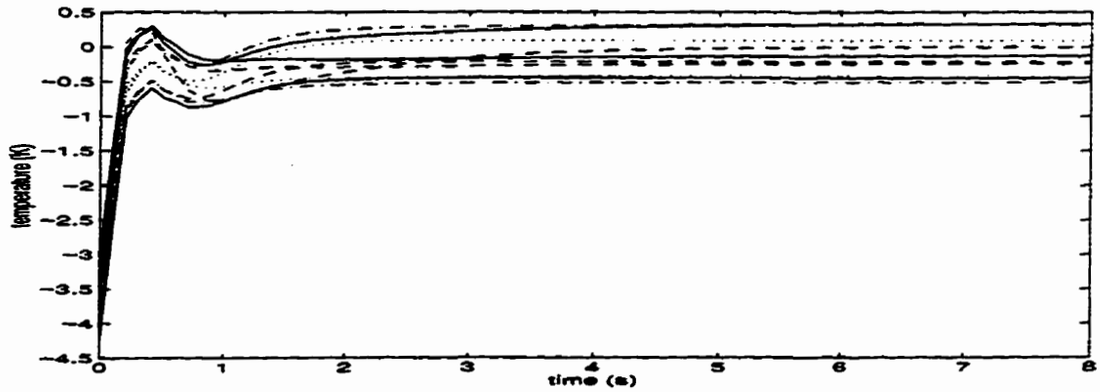
From the test cases done in Section 5.4.1, Section 5.4.2 and Section 5.4.3, we have shown that the LQGI feedback controller can efficiently handle the wafer model mismatch. That means this controller will automatically adjust the lamp powers to reach the process requirements when the actual wafer RTP system has slight differences from the model or the

actual system has a slight disturbance.

A: subtraction of process setpoints from temperatures
of the system with h_i as $0.9 \times H$



B: subtraction of process setpoints from temperatures
of the system with h_i as $1.1 \times H$



C: subtraction of process setpoints from the temperatures
with the system without convection

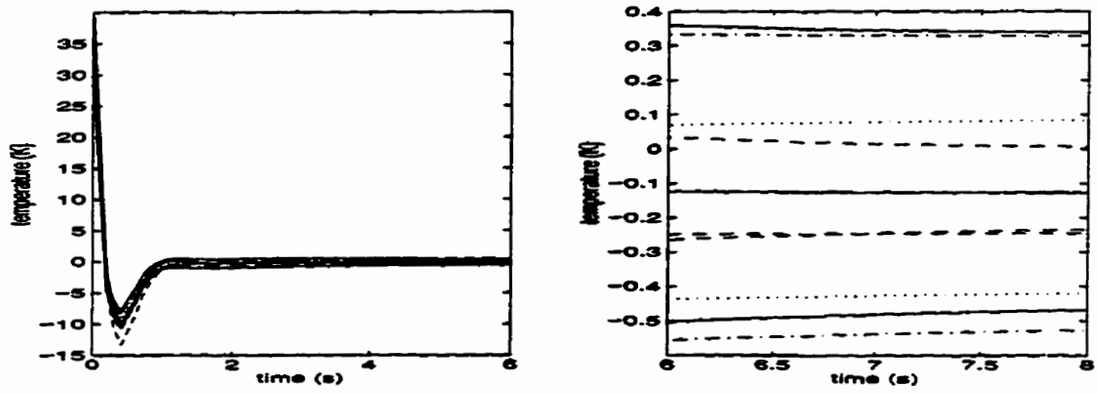


Figure 5.22 : Comparing temperature trajectories with different values of h_i

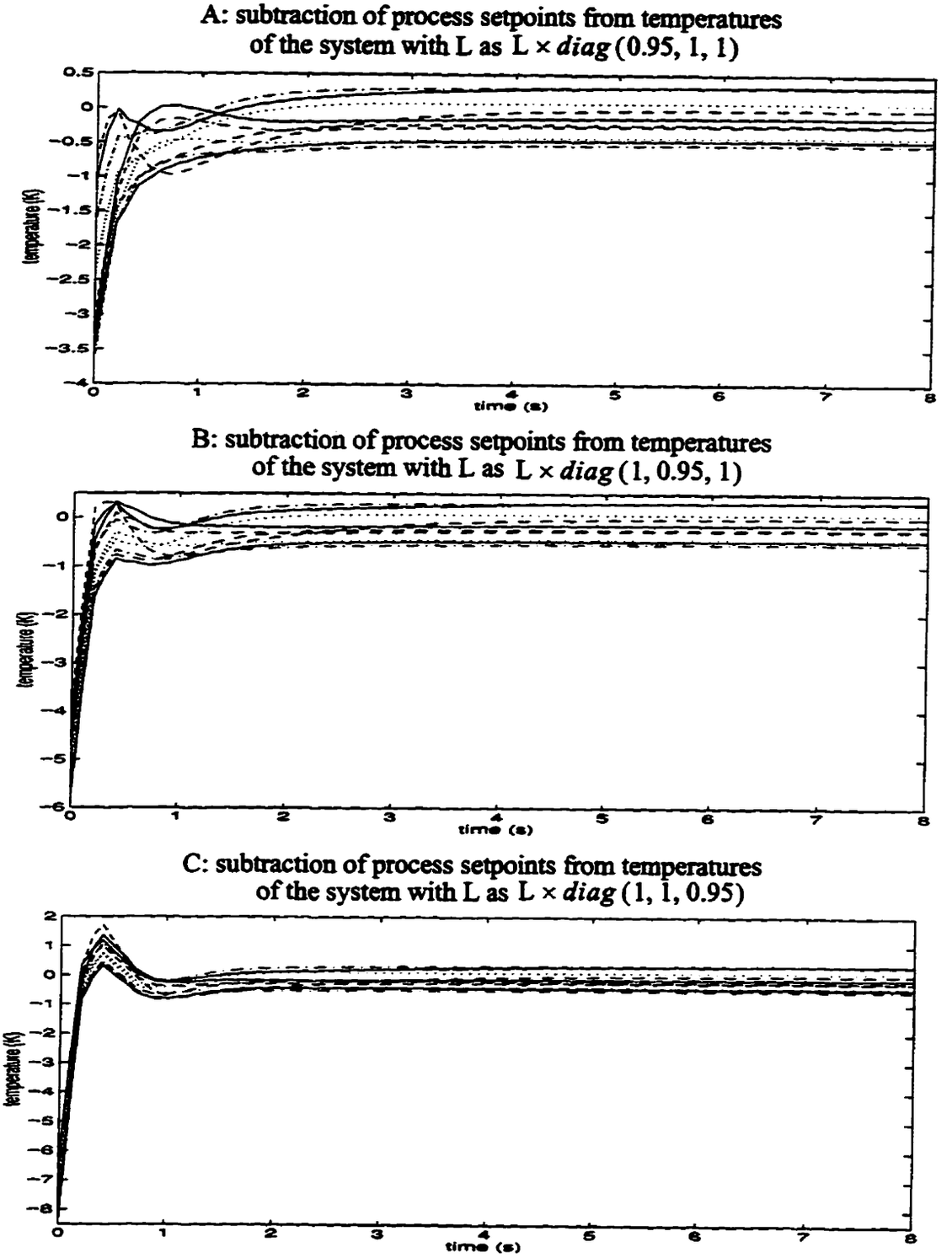


Figure 5.23 : Comparing temperature trajectories with different values of L

Chapter 6 Conclusions

In this chapter, a list of thesis contributions is given and topics for further research are suggested

6.1 List of Thesis Contributions

The paragraphs of this section form a list of the contributions made in this thesis.

Combination wafer models. The thermal models used in this thesis (nonlinear and linear) are the combination models of ideal thermal models and the lag factor. The ideal model is adopted from [Nor92]. The lag factor makes the wafer RTP models close to the actual system due to gradual lamp power changes.

Design of feedback controllers. We have designed a control system which includes an open-loop controller and a feedback controller. The open-loop controller is adopted from [Nor92]. For the feedback controller, we have designed four different types of digital controllers (P, PI, LQG and LQGI). For the LQG and LQGI controller, we have presented basic ideas for how to select the weighting matrices, which are very important for the designs of these two controllers.

Design of real-time software applications. We have designed two real-time software applications. The first is the wafer RTP emulator and the second is the digital control system. These two applications only communicate through I/O boards. The emulator application can be replaced by the actual wafer RTP system without changing the code of the

controller application.

Implementation of real-time software applications within Unix environment. We have implemented both the emulator and controller applications within an Unix environment by using the C programming language. We have also designed a module of I/O emulations. This module is the only one which provides communication channels for these two applications. These two applications allow the user to run the simulation in many different environments including wafer RTP system model, initial lamp powers, initial wafer temperatures, process temperature trajectories and feedback controllers. The details of how to edit initialization files and how to generate initialization files for setting the environment are also provided.

Suggestions of the installation of applications to two PCs. We have given some brief suggestions on how to implement both the emulator and controller applications to two separate personal computers. These suggestions are the selections of hardware and software, and the design of the clock model which provides the reference to the real-time.

Design of graphic user interface. We have presented the details on how to design two user graphic interfaces. The first is for the emulator application and the second is for the controller application. Each application includes one main window and three sub-windows which provide initialization settings for process environment and running time information. For the emulator application, this information includes real-time wafer temperatures for the center element, real-time element temperatures compared with the one of the center element, and real-time lamp power settings. For the controller application, this information includes the real-time estimate state for the center element (only available for LQG and LQGI feed-

back controllers), real-time estimate states compared with the estimate state for the center element (only available for LQG and LQGI feedback controllers), real-time open-loop lamp powers and total lamp powers.

Presentation of the simulation results. The simulation results of model test and control system functionality test have been provided. Through these simulations, we have shown the general behaviors of wafer nonlinear and linear models. We have also shown that the control system with the LQGI feedback controller designed among our cases can control the wafer temperatures to near uniformity, while also following fast temperature trajectories in a wide range. In addition, it is shown that LQGI feedback controller can handle the slight model mismatch well.

6.2 Further Research

This section contains a brief list of research projects which complement the work done for this thesis.

Implementation. To verify that the software applications can be run in real-time, we need to implement them in two separate personal computers. We also need to design a clock module to track the time and send signals to the process requesting an interrupt. Graphic use interfaces for these applications need to be implemented by using a software package such as Labwindows.

Test and development. All implementations mentioned above need to be tested. The design of real-time software applications need to be developed to save memory and handle more complicated cases.

Enhanced mathematical wafer RTP model. The wafer model used in this thesis is only a simple thermal model. This model simplifies the heat transfers inside of the chamber, as it does not consider the radiation from chamber wall to wafer element, the view factor, etc. (see [Nor92]). An enhanced mathematical wafer RTP model needs to be generated to be closer to the actual system.

Appendix A Real-Time Software User's Guide

A.1 Introduction

The purpose of this appendix is to guide the user on how to use the real-time software applications, which are designed in Section 4.3 and are used for implementation study under Unix. This guide requires some knowledge of wafer RTP system, Matlab scripts and the C programming language. It is organized into the following sections:

Section A.2 gives a brief introduction about the real-time software applications and what the user has to set up and select before the simulation start.

Section A.3 provides details on how to set wafer RTP model, initial wafer temperatures and lamp powers for the emulator application.

Section A.4 considers details of selecting and resetting a feedback controller for the control system application. How to set a process temperature trajectory is not provided since it is already clearly introduced in Section 4.3.5.1.

Section A.5 provides information about how to run the program and how to view the results.

A.2 Overview

The software applications designed in Section 4.3 are used for implementation study within an Unix environment for wafer temperature control in RTP. These applications include the emulator application and the control system application. The emulator applica-

tion emulates the actual real-time Wafer RTP system and the control system application acts as a digital controller to control the lamp powers. These two applications only communicate through a module of I/O emulations. See Section 4.3 for more details.

The directory structure of this software is listed as Figure A.1. All the files introduced in Section 4.3.1 will be stored under these four directories according to their functions. The details about these files are introduced in Section 4.3.

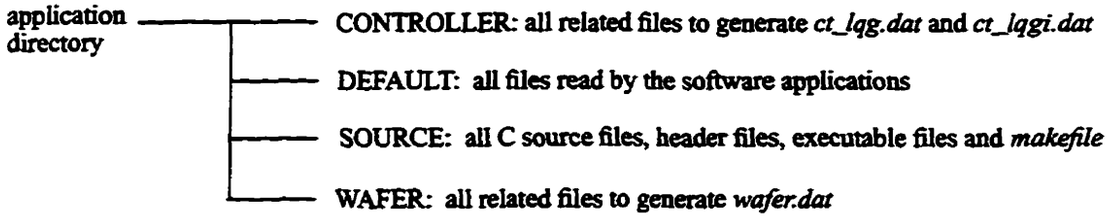


Figure A.1 : Directory structure overview

Before running the program, the user has to create a new directory. Under this new directory, the user must have following files:

- *io_quant.dat* and *interface*
- *wafer.dat*, *ini_p.dat* and *ini_t* for wafer initialization
- *set_procs* and a corresponding text file to store the controller data (*ct_p.dat*, *ct_pi.dat*, *ct_lqg.dat* or *ct_lqgi.dat*) for controller initialization.

If the user would like to use our example systems, all above files can be copied from the DEFAULT directory. Otherwise, users can copy files from a related directory to this new directory to generate or edit the above files. As a result of generating or editing these files, users can:

- set the number of quantization levels of the emulated I/O board (Section 4.3.2)

- set interface for the emulator and the controller application (Section 4.3.4.1)
- get wafer RTP model, set initial temperatures and initial lamp powers for the emulator application
- set process temperature trajectory, select or set a feedback controller and the controller parameters for the controller application.

Details of above settings and selections are provided in Section A.3 and Section A.4. Note all changes should be done only under the directory created by the user.

A.3 User Settings for the Emulator Application

The wafer RTP model implemented in the emulator application is presented as a vector-matrix form as Equation (2.46). The wafer is heated by lamps in concentric ring lamp zones. The effect of thermal expansion is neglected. See Chapter 2 for more details.

A.3.1 Wafer RTP Model

The parameters that the user needs to set for the wafer RTP model are represented in Table 4.1 and Table 4.2. To set the wafer model, simply

- copy *model_w* and *set_wafer.m* from WAFER
- copy *sensor_p.dat* from DEFAULT
- edit the text file *model_w*
- change the **values** following the parameter names which the user wants to reset, **but do not change the parameter names.**
- save the file *model_w*

- edit the text file *sensor_p.dat*
- change the **values**, which represent the sensor positions, ensure that the total number of values is the same as the number of sensors as defined in file *model_w* and the range of values are from one to the number of elements as defined in *model_w*
- save the file *sensor_p.dat*
- be within Matlab programming environment
- type **set_wafer** and press <<return>> key
- type **quit** to get back to Unix Shell.

The Matlab m file, *set_wafer.m,* will produce the text file, *wafer.dat*, which stores the data for the wafer RTP vector-matrix model.

The file *model_w* looks like this:

```
LAMPNUM          3
WAFERNUM         10
EMISSIVITY       0.6
THERMALCOND      30      <-- unit: W/mK
SPECIFIHEAT      691     <-- unit: J/KgK
TGAS             300     <-- unit: K
STEPW            0.01    <-- unit: second
SENSORNUM         3
```

For example, if users want to change the wafer element to 12 and the gas temperature to 250K, The new *model_w* looks like this (the bold numbers have been changed):

```
LAMPNUM          3
WAFERNUM         12
SENSORNUM        3
EMISSIVITY       0.6
THERMALCOND      30
SPECIFIHEAT      691
TGAS             250
STEPW            0.01
```

The *sensor_p.dat* looks like this

```
1
8
10
```

For example, if users want to change the sensor position for the last sensor to the 12th element, the new *sensor_p.dat* looks like this (the bold number has been changed):

```
1
8
12
```

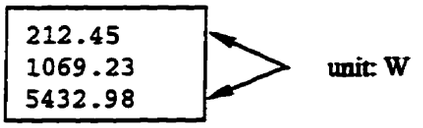
If users make a mistake in editing *model_w* or *sensor_p.dat*, an error message will be shown on the screen when *set_wafer.m is* creating *wafer.dat*. As a result, *wafer.dat* will not be generated and the simulation can not be started.

A.3.2 Initial Lamp Powers

To set the initial lamp powers, users simply have to

- copy the text file *ini_p.dat* from DEFAULT directory
- edit *ini_p.dat*
- change the **values** which correspond to the lamp zones users want to reset (the first value presents the center lamp zone; the last one represents the edge lamp zone). Ensure that the total number of values are the same as the number of lamp zones as defined in file *model_w*. The value for each lamp zone must not exceed the "Low Limit" and "High Limit" (Table 4.3) for that lamp zone.
- save the file *ini_p.dat*.

init_p.dat looks like this:



For example, if users want to reset the lamp powers for the edge lamp zone as 8764.35W, the new init_p.dat looks like this (the bold number has been changed)

```
212.45
1069.23
8764.35
```

If users make a mistakes in editing *ini_p.dat*, an error message will be shown during the emulator process initialization, and the simulation will be terminated.

A.3.3 Initial Wafer Temperatures

The values for the initial temperatures across the wafer can be edited in two different ways.

To set the initial temperatures as steady-state ones under lamp powers defined by *ini_p.dat*, users have to:

- copy the text file *ini_t* from DEFAULT directory
- edit the text file *ini_t*
- type **STEADYSTATE**
- save the file *init_t*.

init_t looks like this:

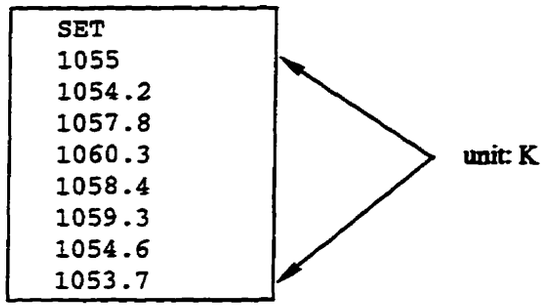
```
STEADYSTATE
```

To set the initial temperatures according to user definition, users need to:

- edit the text file *ini_t*

- type **SET**
- type **values** following after keyword SET, (ensure the total number of values is the same as the number of elements as defined in *model_w* and the range of values must be from 200K to 1700K)
- save the file *init_t*.

For example, if the number of wafer elements is 8, *init_t* looks like this:



If users make a mistake in editing *ini_t*, an error messages will be shown during the emulator process initialization and the simulation will be terminated.

A.4 User Settings for the Control System Application

The control system includes an open-loop controller and a feedback controller. Only the feedback controller can be selected and reset. The four types of feedback controllers available are P, PI, LQG, and LQGI. The LQG and LQGI feedback controllers are based on the wafer linear model as Equation (2.41).

How to set the process temperature trajectory can be found in Section 4.3.5.1.

A.4.1 Selection of a Feedback Controller

In our example cases, we only have four feedback controllers implemented (accord-

ing to four types of feedback controllers designed in Chapter 3). Users can select any one of them.

These four feedback controllers have three inputs and three outputs, which means that the controller systems can only receive signals from three sensors and can only control lamp powers for three lamp zones. The data for these four feedback controllers are stored in text files *ct_p.dat*, *ct_pi.dat*, *ct_lqg.dat* and *ct_lqgi.dat*. For P and PI controllers, the values of gain matrix and reset time are presented in Section 5.3.3. For LQG and LQGI feedback controllers, the values of controller parameters are presented in Section 5.3.4 and Section 5.3.5; the parameter values of the plant are listed as example values in Table 4.1; the sensor positions are located as Table 4.2. The controller sampling times for all our cases are set as 0.1 seconds.

To select a feedback controller, users need to run the corresponding executable file listed in Table 4.6. For an example, if we want to select the LQG feedback controller in the control system, we need to run the file named *sim_lqg*. More details will presented in Section A.5.

A.4.2 Resetting of a P or PI Feedback Controller

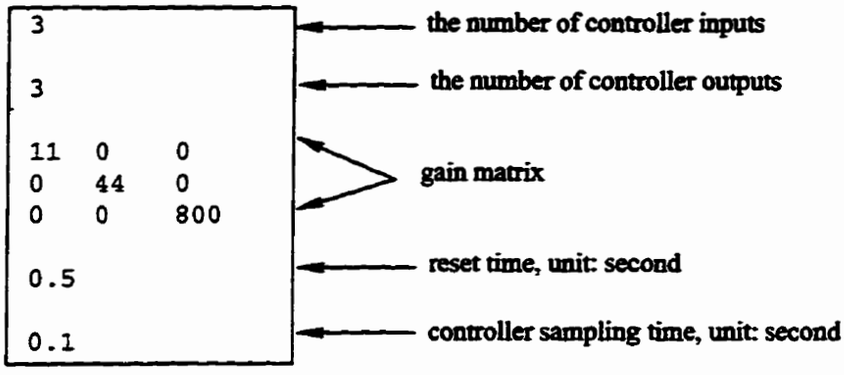
To reset a P or PI feedback controller, users only have to:

- copy *ct_p.dat* or *ct_pi.dat* from DEFAULT directory
- edit *ct_p.dat* or *ct_pi.dat*
- change, add or delete the **element values** of feedback gain, **scale values** of reset time or controller sampling time (Section 4.3.5.1), note that the dimensions of the feedback gain must match the numbers of the controller inputs and the con-

troller outputs. The controller sampling time must be several times larger than the emulator sampling time defined in *model_w*

- save *ct_p.dat* or *ct_pi.dat*

ct_pi.dat looks like this:



If an error appears inside of *ct_pi.dat* or *ct_p.dat*, an error message will be shown during the controller initialization process, and the simulation will be terminated.

A.4.3 Resetting of a LQG or LQGI Feedback Controller

To reset a LQG or LQGI feedback controller, users have to:

- copy *model_c* and *set_lqg.m (*or *set_lqgit.m)* from CONTROLLER directory
- edit *model_c*
- change the **element** or **scalar values** of the parameters listed in Table 4.5, ensure that the conditions in "Parameter Structure" are reached and the controller sampling time must be several times larger than the emulator sampling time defined in *model_w*
- save *model_c*
- edit *set_lqg.m* or *set_lqgi.m* to set the number of controller inputs as the same as

the number of sensors as defined in *model_w*, to set the number of controller outputs as the same as the number of lamp zones as defined in *model_w*, and to build a plant according to the parameters defined in *model_w*. (This step is quite complicated. If users are not familiar with Matlab, users should find their own way to generate *ct_lqg.dat* or *ct_lqgi.dat*. This step could be omitted if users have selected the same emulator model as our example).

- be within Matlab programming environment
- type *set_lqg* for resetting a LQG feedback controller or type *set_lqgi* for resetting a LQGI feedback controller, and press <<return>> key
- type **quit** to get back to Unix Shell

The Matlab m file, *set_lqg.m*, will generate the text file *ct_lqg.dat*, and *set_lqgi.m* will generate the text file *ct_lqgi.dat*.

The file *model_w* looks like this:

```
INPUTS          3
OUTPUTS         3
STATES          10
R               9.8 0     0
                0   0.6   0
                0   0     0.005
C1              10
C2              10000
CV              0.001
W               120 0      0
                0   2200   0
                0   0      8000
STEPC           0.1
```

If an error appears in *model_c*, an error message will be shown when *set_lqg.dat* is creating *ct_lqg.dat* or *set_lqgi.dat* is creating *ct_lqgi.dat*. As a result, *ct_lqg.dat* or *ct_lqgi.dat* will not be generated and simulation can not be started.

A.5 How to Run the Program and How to View the Results

Before starting this process, the user has to ensure that all files read by applications are copied, edited or generated correctly under the user's directories.

To run the program, users simply:

- type the selected executable file name, *sim_lqg, (sim_p, sim_pi* or *sim_lqgit*) in Unix Shell and under the directory created by themselves
- press <<return>.

Graphic user interfaces are not implemented and only two log files, *LAMP_P.dat* and *WAFER_T.dat* will be generated after simulation stops.

WAFER_T.dat stores the values for real-time temperatures (Kelvin degrees) of all wafer elements at each emulator sampling point. This file has I values at each line. The values in the first column are for the center element, the values in the second column are for the second element and so on. This file has,

$$1 + floor\left(\frac{TotalSimulationDuration}{STEPC}\right) \times floor\left(\frac{STEPC}{STEPW}\right)$$

rows, where floor rounds the values to the nearest integers towards minus infinity. (To plus 1 is to include the initial point).

LAMP_P.dat stores the values for total lamp powers calculated by digital control system at each controller sampling point (the unit is Watts). This file has three values at each row since we use a three lamp zone system. The values of the first column are for the inner lamp zone, the values of second column are for the middle lamp zone, and the values of the

third column are for the edge lamp zone. This file has,

$$1 + floor\left(\frac{TotalSimulationDuration}{STEPC}\right) \text{ rows.}$$

To view the information inside of *LAMP_P.dat* or *WAFER_T.dat*, we can either open this text file directly to see the values or go through some software packages, like Matlab or Xmgr, to load the data and generate figures. For example, if users want to see the real-time temperatures for the center element, the emulator sampling period is 0.01 seconds and the total sampling duration is 24 seconds, we can use the Matlab package to:

- type **load WAFER_T.dat;**
- type **TIME = [0 : 0.01 : 24]';**
- type **plot(TIME, WAFER_T(:,1)).**

Therefore, the values inside *WAFER_T.dat* are loaded as a matrix named WAFER_T and plotted in a figure with x-axis in seconds and y-axis in Kelvin degrees (See [Mat92] for reference).

Bibliography

[ApS92] Pushkar P. Apte and Krishna C. Saraswat. *Rapid thermal processing uniformity using multivariable control of a circularly symmetric 3 zone lamp*. IEEE Trans. Semicond. Manufact.,Vol. 5, No. 3, Aug. 1992

[ÅsW84] Karl J. Åström and Björn Wittenmark. *Computer controlled systems theory and design*. Prentice-hall, Inc., Englewood Cliffs, N.J. 07632, 1984

[AuC90] David M. Auslander and Cheng H. Tham. *Real-time software for control*. Prentice-Hall, Inc. 1990.

[Ben88] Stuart Bentnett. *Real-Time computer control: an introduction*. Prentic hall international (UK) Ltd., 1988.

[Bel95] Trudy E. Bell. *Electronics and the stars*. IEEE spectrum, Vol. 32, No. 8, Aug. 1995, pp. 16-24.

[BMKM95] Sergey Belikov, helen Martynov, Michael Kaplinsky, and Constantine Manikopoulos. *Using wavelength-dependent emissivity of semiconductor wafer to model heat transfer in rapid thermal processing station*. IEEE Tran. on Semicond. Manufact., Vol. 8, No. 3, Aug. 1995.

[CAKLL91] Stephen A. Campbell, K. -H. Ahn, Karson L. Knutson, Benjamin Y.H. Liu and John D. Leighton. *Steady-state thermal uniformity and gas flow patterns in a rapid thermal processing chamber*. IEEE Tran. on Semicond. manufact., Vol. 4, No. 1, Feb. 1991.

[ChF93] Tongwen Chen and Bruce Francis. *Sampled-Data control systems*. Version

4.0., Sept. 1993

[CPKX94] Y. M. Cho, Arogyaswami Paulraj, Thomas Kailath, and Guanghan Xu. *A contribution to optimal lamp design in rapid thermal processing.* IEEE trans. Semicond.Manufact.,Vol. 7, No. 1, Feb. 1994.

[CuS95] Charles W. Cullen and james C. Sturm. *Temperature measurement of metal-coated silicon wafer by double-pass infrared transmission.* IEEE Tran. on Semicond. Manufact., Vol. 8, No. 3, Aug. 1995.

[DaZ95] M.H.A. Davis and M. Zervos. *A new proof of the discrete-time LQG optimal control theorems.* IEEE Tran. on Automatic control, Vol. 40, No. 8, Aug. 1995, pp. 1450-1453.

[DMMP87] Richard A. DeMillo, W. Michael McCracken, P.J. Martin, and John F. Passafiume. *Software testing and evaluation.* The Benjamin/Cummings Publishing Company Inc., 1987.

[FPEN94] Gene F. Franklin, David Powell and Abbas Emami-Naeini. *Feedback control of dynamic systems.* Addison-Wesley Publishing Company, Inc. 1994.

[Gaw94] W. Gawronski. *A balanced LQG compensator for flexible structures.* IEEE Trans. Automatica, Vol. 30, No. 10, Aug. 1994

[GRS91] Ronald S. Gyurcsik, Terrence J. Riley, and F. Yates Sorrell. *A model for rapid thermal processing: achieving uniformity through lamp control.* IEEE Trans. Semicond. Manufact.,Vol. 4, No. 1, Feb. 1991

[KaB89] R. Kakoschke, E. Bubmann. *Simulation of temperature effects during rapid thermal processing.* Siemens AG, HL T 312, Otto-hahn-ring 6, D-8000

München 83, FRG, 1989.

[KwS72] Huibert Kwakernaak, Raphael Sivan. *Linear optimal control systems*. Wiley-Interscience, a division of John Wiley & Sons, Inc, 1972.

[Lor88] H. A. Lord. *Thermal and stress analysis of semiconductor wafers in a rapid thermal processing oven*. IEEE Trans. Semicond. Manufact.,1(3); 105-114, August 1988.

[LuP95] Peter Lute and Dolf van Paassen. *Optimal indoor temperature control using a predictor*. Control systems, Vol. 15, No. 4, Aug. 1995, pp.4-10.

[Mat92] MathWorks, Inc.. *MATLAB High-Performance numeric computation and visualization software reference guide*. The MathWorks, Inc., July 1992.

[MatC92] MathWorks, Inc.. *Control system toolbox user's guide*. The MathWorks, Inc., July 1992.

[MatS92] MathWorks, Inc.. *SIMULINK Dynamics system simulation software user's guide*. The MathWorks, Inc., March 1992.

[MHADSP95] David J. Musliner, James A. Hendler, Ashok K. Agrawala, Edmund H. Durfee, Jay K. Strosnider and C.J. Paul. *The challenges of real-time AI*. IEEE Computer, Vol. 28, No. 1, Jan. 1995, pp. 58-65.

[Nat95] National Instruments Corporation. *Instrumentation reference and catalogue*. National Instruments Corporation, 95.

[Nor92] Stephen A. Norman. *Wafer temperature control in rapid thermal processing*. Ph.D. dissertation, Stanford University, 1992.

[Oga70] Katsuhiko Ogata. *Modern control engineering*. Prentice-Hall, Inc., Engle-

wood Cliffs, N.J., 1970.

[RoC87] Brian Roffel and Patrick Chin. *Computer control in the process industries.* Lewis Publishers, Inc.,1987.

[SABCDDF94]Krishna C. Saraswat, Pushkar P. Apte, leonard Booth, Yunshong Chen, paul C.P. Dankoski, F.levent Degertekin, Gene F. Franklin, B.T. Khuri-Yakub, M.M. Moslehi, Charles Schaper, paul J. Gyugyi, Y.J. Lee, J. Pei, and Samuel C. Wood. *Rapid thermal Multiprocessing for a Programmable factory for adaptable manufacturing of IC's.* IEEE Trans. Semicond. Manufact.,Vol. 7, No. 2, May 1994.

[SCK92] C.D. Schaper, Y.M. Cho, and T. kailath. *Low-order modeling and dynamic characterization of rapid thermal processing.* Appl, Phy. A: Solids and surfaces, Vol. A54, No. 4, pp. 317-326, Apr. 1992.

[ShC89] Tsay-Jiu Shieh, and Ronald L. *RAPS - A rapid thermal processor simulation program.* IEEE Tran. on Electron devices, Vol. 36, No. 1, Jan. 1989,pp. 19-24.

[ShC95] K.G. Shin and X. Cui. *Computing time delay and its effects on real-time control systems.* IEEE Trans. Control systems technology, Vol. 3, No. 2, June 1995, pp. 218-224.

[SKN87] B. Van Schravendijk, W. De Koning, and W. Nuijen. *Modeling and control of the wafer temperatures in a diffusion furnace.* J. Appl. Phys., Vol. 61, No. 4, Feb. 1987

[SMSK94] Charles Schaper, Mehrdad Moslehi, krishna Saraswat, and Thomas Kailath.

Control of MMST RTP: repeatability, uniformity, and integration for flexible manufacturing. IEEE Trans. Semicond. Manufact.,Vol. 7, No. 2, May 1994

[SRLG] E. Schurack, W. Rupp, T. Latzel, and A. Gottwald. *Analysis and measurement of nonlinear effects in power amplifiers caused by thermal power feedback*. Wermer-Heisenberf-Weg 39 D-8014 Neubiberg

[Ste87] James Stewart. *Single Variable Calculus*. Wadsworth, Inc., Belmont, California 94002.

[SYK94] F. Yates Sorrell, Seungil Yu, and William J. Kiether. *Applied RTP optical modeling: an argument for model-based control*. IEEE Tran. on Semiconductor manufacturing, Vol. 7, No. 4, Nov. 1994.

[Too88] Michael Tooley. *Bus-based industrial process control*. Heinemann Professional Publishing Ltd., 1988.

[TzP90] Spyros G. Tzafestas and J.K. Pal. *Real time mirocomputer control of industrial processes*. Kluwer Academic Publishers 1990

[VrZ89] Zvonko G. Vranesic and Safwat G. Zaky. *Microcomputer structures*. Saunders College Publishing, a Division of Holt, Rinehart and WInston, Inc., 1989.

[XHM95] Y. Xu, J.M. Hollerbach, and D. Ma. *A nonlinear PD control for force and contact transient control*. Control system, Vol. 15, No. 1, pp. 15-21.

[ZWMG95] C. Zhou, J.R. Whiteley, E.A. Misawa, and K.A.M. Gasem. *Application of Enhanced LQG/LTR for distillation control*. Control system, Vol. 15, No. 4, Aug. 1995.