This is to certify that the
dissertation entitled

PROCESS OPTIMIZATION FOR MOIST AIR IMPINGEMENT COOKING OF MEAT PATTIES

presented by

SANGHYUP JEONG

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Biosystems Engineering

Major Professor's Signature

December 14, 2005

Date

PROCESS OPTIMIZATION FOR MOIST AIR IMPINGEMENT COOKING OF MEAT PATTIES

By

Sanghyup Jeong

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Biosystems and Agricultural Engineering

2005

ABSTRACT

PROCESS OPTIMIZATION FOR MOIST-AIR IMPINGEMENT COOKING OF MEAT PATTIES

By

Sanghyup Jeong

Process conditions of a moist air impingement cooking system were optimized to achieve maximum yield and to satisfy safety and quality constraints, simultaneously. To accomplish this goal, various strategies were tested by combining different modeling approaches, global optimization algorithms, and parameterization of control profiles.

In this study, a finite element model (FEM) predicting yield, *Salmonella* inactivation, internal color change, and surface color change was considered as an actual experiment with which all the results were compared. Static neural network models (SNNM) and a dynamic neural network model (DNNM) were utilized as potential, faster alternatives to the finite element model. For the global optimization algorithms, genetic algorithms (GA), simulated annealing (SA), and integrated controlled random search in dynamic system (ICRS/DS) algorithms were tested along with the finite element model and alternative models. In addition, piecewise linear interpolation (PLI) and Fourier series (FS) were used for the control profile parameterization.

This study was conducted in two different ways. In the first part, overall aspects of this optimization problem and the effectiveness of the various strategies were investigated to identify the best strategy for ideal dynamic control profiles. Secondly, based on prior knowledge, the optimization strategies were applied to several industrially-relevant case studies.

The performance of the alternative models (DNNM and SNNM) was fast, general, and robust, with a few exceptions. Even though the accuracy and the power of classification were not as high as the finite element model results, the neural network models showed potential as reliable alternative models. The highest goal (yield) was 73%, which was obtained by using the ICRS algorithm, FEM, and PLI. However, the optimization strategies with alternative models could not find such a high yield; rather, they committed critical classification errors at the later stages of the optimization process. Generally, all the global optimization algorithms showed convergence to an optimal solution, albeit with different convergence speed and goal achievement. Although comprehensive evaluation was impossible, ICRS was observed as the most recommendable algorithm.

Single-stage, double-stage, and multi-zone processes were studied by using three different models (FEM, DNNM, and SNNM) and the ICRS algorithm. The maximum yield (67%) was achieved in the double-stage process. The case studies showed that a simple and minor design change of the single-stage oven might improve the performance.

In addition, the objective function (yield) for the single-stage oven was replaced with a cost function, and the operating conditions for maximum profit were determined, which were different from the results when the objective function was yield. Finally, Monte Carlo simulation showed that all the optimal profiles were highly sensitive to small perturbations, which implied difficulties in the actual application of the optimal solutions, due to unavoidable control errors of a cooking system.

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dr. Bradley P. Marks, a truly inspiring teacher and counselor; for his guidance and friendship during the course of this investigation. Grateful appreciation is also extended to Dr. Alden M. Booren, Dr. Kirk Dolan and Dr. James Steffe for serving on my guidance committee.

The author is indebted to the United States Department of Agriculture (USDA) Cooperative State Research, Education, and Extension Service for supporting project 2003-51110-02081 of which this study is a part.

Thanks go to our research team members and colleagues of graduate office 5. Special thanks also goes to college group of the New Hope Baptist Church for their prayer and good memories in God. Dr. Sukjung Chang, MD and his wife deserve to have my sincere thanks for their spiritual encouragement all the way through this journey. Of course, my family and parents have been the essence of refreshment for every new challenge in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Roman Letters:

a	a parameter of a sigmoid activation function
a	color value of redness
a_0	coefficient of Fourier series
a_k, b_k	k^{th} coefficients of Fourier series
b	color value of yellowness
C	concentration, [kg/m^3]
C_s	surface color change
c	cooling rate
c_m	mass capacity, [kg/kg]
c_T	specific heat capacity, [J/(kg °C)]
D	decimal reduction time, [s]
D_{cap}	capillary diffusivity, [m^2/s]
$D_{cap.fat}$	fat capillary diffusivity, [m^2/s]
E	expected value
E_a	activation energy, [J/gmol]
F	failure counter in ICRS/DS
f	nonlinear activation function, or probability density function
H	humidity (% moisture by volume), [%MV]
h	output of hidden layer neuron, or enthalpy, [kJ/kg]
h_m	convective mass transfer coefficient, [m/s]
h_T	convective heat transfer coefficient, [W/(m^2 °C)]
C_c	internal color change at center
J	objective function, or performance index
k	Boltzmann constant, or inactivation rate constant, [1/s]
k_1	heuristic parameter of ICRS/DS
k_2	heuristic parameter of ICRS/DS
k_m	moisture conductivity, [$kg_{moisture}$/(m s)]
k_T	thermal conductivity, [W/(m °C)]
L	color value of lightness
L	log cycles of microbial destruction
m	a multiplier, or moisture or fat content-decimal dry basis
M	number of control variables
$\overline{m}_{ff}$	final averaged fat content based on non-fat solids, [kg/kg]
m_{fo}	initial fat content based on non-fat solids, [kg/kg]
$\overline{m}_{wf}$	final average water content based on non-fat solids, [kg/kg]
m_{wo}	initial water content based on non-fat solids, [kg/kg]
N	neighborhood function, number of discretization, total number of cooling cycle or number of microorganism, [CFU/g]
n	normal to surface
n_e	heuristic parameter in ICRS/DS

p	parameters of control functions, or probability
R	ideal gas constant, [J/(kmol K)]
r	random number, or r-direction in radial coordinates
S	step size
T	temperature, [°C]
t	time, [s]
t_{exe}	total execution time, [s]
T_{dew}	dew point temperature, [°C]
V	impingement exit velocity, [m/s]
v	sum of neuron
w	weight
y	output of neuron
Y	patty yield, [%]
z	vertical direction in cylindrical coordinates, or microbial z-value, [°C]
ΔE	total color change

Greek Letters:

α	search direction
Γ	vector of random numbers (Gaussian distribution)
ε	error, or small number
ε_A	efficiency of algorithm [% yield/s]
η	learning rate constant
θ	time coordinates of control profile parameterization grid
λ	minimum interval, or latent heat [kJ/kg]
ξ	decision variables vector
ρ	density, [kg/m^3]
σ	standard deviation vector, or constant for the size of receptive field
ω	control coordinates of control profile parameterization grid

Bold style (vector or matrix):

$\boldsymbol{y}$	output vector of individual neuron
$\boldsymbol{t}$	target vector
$\mathbf{u}$	control vector, or training vector
$\mathbf{W}$	weight matrix
$\mathbf{x}$	input column vector, or control vector

Superscript:

HL	hidden layer
i	i^{th} iteration , point, or initial state
j	j^{th} neuron
k	k^{th} iteration, or time step, or neuron
OL	output layer

Subscript:

0	initial state
c	critical value
f	final state
L	lower limit
n	number of step
r	reference state
s	surface
U	upper limit

Abbreviations:

ANN	artificial neural network
BB	branch and bound
FEM	finite element model
FFNN	feed-forward neural network
GA	genetic algorithm
GO	global optimization
GRG	generalized reduced gradient
GRNN	generalized regression neural network
ICRS/DS	integrated controlled random search for dynamic system
LP	linear programming
MILP	mixed integer linear programming
MINLP	mixed integer nonlinear programming
MIP	mixed integer programming
NLP	nonlinear programming
OMb	oxymyoglobin
PLI	piecewise linear interpolation
QP	quadratic programming
RBF	radial basis-function, or radial basis-function neural network
RMSE	root mean squared error
SA	simulated annealing
SLP	successive linear programming
SSE	sum of squared error

1 INTRODUCTION

1.1 Background

1.1.1 Food Quality and Safety

Food processing technology has been advanced along with human history. However, the fundamental concept of food processing has not changed considerably, even though many innovative processes and products are being developed. Basically, manufacturing of foods encompasses two types of conversions, physical and chemical. The main purposes of these conversions are preserving and improving the quality of processed food products. One of the predominant unit operations, inducing both physical and chemical conversions, is heat or thermal processing.

Thermal processing is transferring heat to a food material to induce desirable results, but this also can cause concurrent undesirable results. Thermal processing increases digestibility (*e.g.*, protein denaturation), reduces enzyme and microorganism activity, and enhances food characteristics (*e.g.*, carbohydrate gelatinization, color development, texture, and flavor changes). However, thermal processing also produces undesirable results, such as loss of heat sensitive nutrients and undesirable color and flavor changes due to over-cooking. Therefore, a compromise must be found between intensity of thermal processing and its various effects on the product (Trystram, 2004).

According to the results of a recent survey published in "Trends In The United States - Consumer Attitudes and The Supermarket 1999"(FMI, 1999), the top food selection concerns and the percentages of the shopping public that consider these factors "very important" in their food selection were as follows: (1) Taste, (92% of those interviewed), (2) Nutrition, (70%), (3) Product Safety (70%), (4) Price, (63%), (5)

Storability, (42%); (6) Ease of Preparation, (35%); (7) Food Preparation Time (35%) and (8) Product packaging that can be recycled, (29%). Therefore, food manufacturing must address not just production volume, but also a variety of other competing criteria.

For the consumer's food safety concern, the USDA Economic Research Service (ERS) reports that the percent of consumers "completely confident" in the safety of the food supply increased from a low of 72% in 1992-93 to a high of 83% in 1996, with levels declining to 74% in 2000 (ERS, 2002).

From the economic viewpoint, food safety is the most critical and nonnegotiable quality factor. In the United States, foodborne diseases have been estimated to cause 6 million to 81 million illnesses and up to 9,000 deaths each year (Mead *et al.*, 1999). For six specific bacterial pathogens, the costs of human illness are estimated to be $9.3-12.9 billion annually (Buzby *et al.*, 1996). In 2000, ERS estimated the annual costs[1] due to selected foodborne pathogens[2] as $6.9 billion (ERS, 2000). These estimated costs are an enormous burden for society and also for food manufacturers.

Therefore, it is clear that the major sectors (*i.e.*, consumers, industry, and regulatory agencies) in the food market must collaborate to improve the current situation.

---

[1] includes medical costs, productivity losses, and costs of premature deaths

[2] *Campylobacter* (all serotypes), *Salmonella* (nontyphoidal), *E. coli* O157, *E. coli* non-O157 STEC, and *Listeria monocytogenes*

1.1.2 The Drive for Food Quality and Safety Innovation

Driving forces can be passive or active. Since the 1993 outbreak of *E. coli* O157:H7, consumer awareness and demand for food safety has increased (Golan *et al.*, 2004). The passive driving forces often come from foodborne illness outbreaks and recalls[3], and they trigger consumer awareness, regulatory changes, or industrial innovation.

"The number and size of recalls have increased dramatically over the last decade. During 1993-96, the number of meat and poultry Class I[4] recalls averaged about 24 per year and amounted to 1.5 million pounds annually; during 1997-2000, Class I recalls averaged 41 per year and reached 24 million pounds annually (Ollinger and Ballenger, 2003)." These increasing recall cases are not because of loose control, but because the ability to detect pathogens on products has increased dramatically, which can generate more recalls (AMI, 2002). In addition, the ability to track foodborne disease and tie it to a specific food product has evolved into a practical technology (AMI, 2002). Recalls result in bad reputation and catastrophic financial damage to a manufacturer.

Therefore, efficient quality assurance has become a critical issue for consumers, manufacturers, and related government organizations. Instrumentation, food safety practices, and lethality criteria are of central importance, with particular emphasis on very high sanitary and hygienic operating standards. Evolving federal regulations, such as

---

[3] A food recall is a voluntary action by a manufacturer or distributor to protect the public from products that may cause health problems or possible death. The purpose of a recall is to remove meat or poultry from commerce when there is reason to believe it may be adulterated (injurious to health or unfit for human consumption) or misbranded (false or misleading labeling and/or packaging) (FSIS, "FSIS Recalls", *USDA*, http://www.fsis.usda.gov/Fsis_Recalls/index.asp, March 22, 2005.).

[4] Recalls that involve meat or poultry products that could, especially without cooking to safe temperature, cause serious illness or death.

9CFR318.17 (FSIS, 1999 & 2001), change safety regulations from passive to active compliance required of the food industry. Traditionally, regulations have provided a specific endpoint temperature and holding time to achieve target lethality in a meat and poultry product. Thus, the traditional approach discourages the food industry from adopting new technology and voluntary compliance to the regulations. However, the evolving regulations require manufacturers to prove, via scientifically supportable means, that their process or operating policy achieves a target lethality performance standard. The transition from command-and-control to performance standards allows more freedom of choosing process design and operation policy, but also moves more responsibility to the industry.

Contrary to the above passive driving forces, there might be an active driving force that originates from industry. Food manufacturers invest in the development of new methodologies to improve safety and quality of their food product. When industry successfully innovates to produce safe foods, a win-win situation arises, with the innovating firm, consumers, and government all benefiting from improved food safety (Golan *et al.*, 2004).

Many attempts have been made to maximize desirable quality and simultaneously minimize undesirable effects by adding ingredients, developing innovative process equipment, improving process conditions, and so forth. Among those approaches, improving process conditions is advantageous, in that it uses existing systems. Therefore, additional capital investment is not necessary to resolve these contradicting factors to achieve both safety requirements and maximize quality and profit. Finding the best operating condition is critical from the perspective of industry, because the need to

improve efficiency, reduce energy consumption, increase productivity, and comply with regulations pushes industry to adopt improved safety practices if they can see simultaneous benefits in yield, which translate to profit.

1.1.3 Economic Significance of the Meat Industry

In spite of the increased safety concerns, the U.S. meat and poultry industry contributes significantly to the U.S. agricultural economy. Total meat and poultry production in 2000 exceeded 80 billion pounds, a 31 percent increase since 1987. The meat and poultry industry is the largest segment of U.S. agriculture, contributing over $100 billion in annual sales to the GNP (AMI, 2001).

The products affected by regulatory changes related to ready-to-eat products account for over $28 billion in annual sales (FSIS, 2001), and consumer trends for ready-to-eat products also suggest continued rapid growth in this category. The size of this market is important as a spur to greater profit in this industry. Accordingly, the industry aims to develop novel products and to increase the efficiency of its production lines. For an example, even a modest 0.5% improvement of cooking yields based on the $28 billion annual sales in this category would give an impact of approximately $140 million increase in annual revenue for ready-to-eat products in the U.S.

Therefore, given the regulatory changes and the economic importance of ready-to-eat (RTE) meat products, there are compelling needs for integrated simulation tools that will allow industry to design and operate processes that meet the lethality performance standards and simultaneously increase quality and profit. Optimization techniques that find the conditions for the best result from a given situation are needed to meet these demands. Process optimization is the most economic approach for industry to

maximize yields while ensuring safety and quality factors with existing facilities. Therefore, the information and tools that will enable the industry to design and operate the optimal processes are essential.

1.1.4 Recent Innovation in Meat Patty Processing

Impingement cooking technology has been popular in certain segments of the food processing industry, because of its efficiency. Specifically, moist air impingement cooking systems are widely used in the ready-to-eat meat product industry, because the system (Figure 1.1) results in short cooking time and relatively high cooking yield.

The moist air impingement cooking systems jets a steam-air mixture through arrays of nozzles onto products, yielding a high heat transfer rate by reducing the thickness of the boundary layer at the surface of the product. Also, at the initial stage of cooking, steam is condensed on the surface of the product, which results in effective transfer of latent heat into the product at low temperature. Therefore, moist air impingement cooking systems are characterized by fast cooking and suppression of moisture loss. Moist air impingement cooking systems involve many control variables, such as cooking duration, air temperature, air moisture content, impingement exit velocity, and impingement geometry (*e.g.*, jet width, spacing, and height).

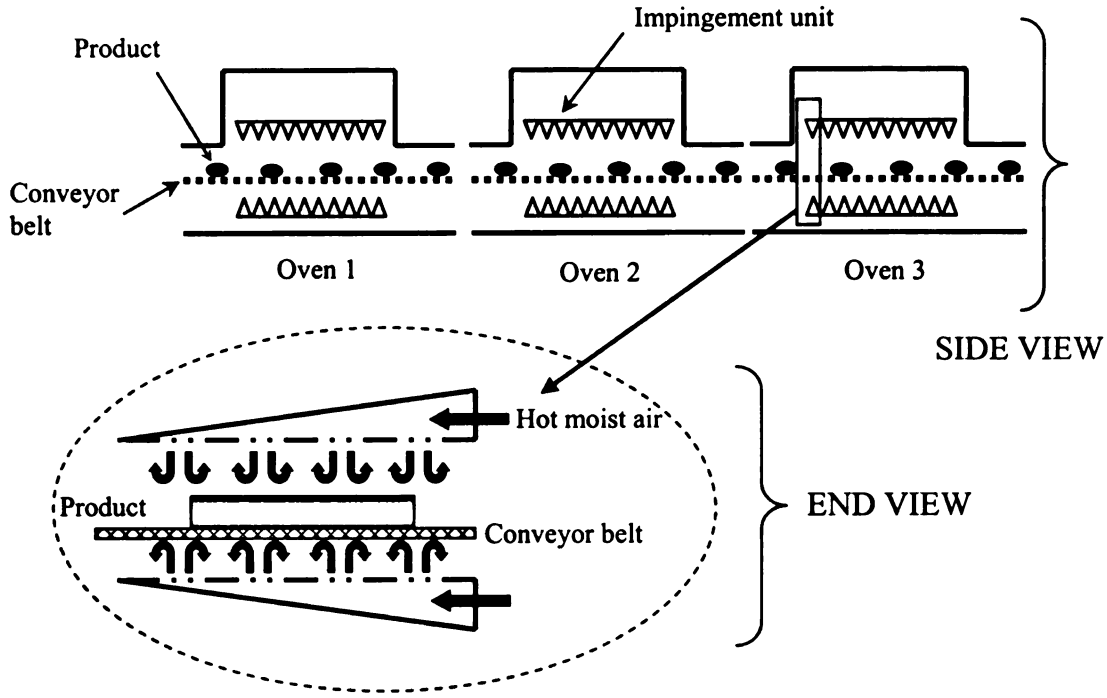


Figure 1.1 Schematic diagram of a multi-staged, moist-air impingement cooking system.

Currently, most oven operators select the oven operating conditions (*i.e.*, the control variables) based on their experience or simple rules of thumb, which are not necessarily proven scientifically. For a single-stage oven, one might operate the oven within sub-optimal conditions. However, if multiple ovens (Figure 1.1) are connected to increase the rate of production with various cooking zones, then the complexity and size of the problem is too large to select optimal conditions with experience. Also, meat cooking under moist air impingement cooking environments involves multiple control variables, mass transfer coupled with heat transfer, phase transitions, and complex condensing boundary conditions, which results in complex combinations of control profiles.

Therefore, optimizing the process conditions of moist air impingement cooking systems is a significant challenge and a worthwhile endeavor from the perspective of researchers, food processors, oven manufacturers, and government agencies.

1.2 Objectives

The overall goal of this study was to develop an efficient process optimization method, in terms of speed and objective-achievement, and to evaluate the method for maximizing cooking yield, while ensuring the microbial safety and quality of ready-to-eat meat products (ground and formed meat and poultry products) cooked in commercial moist air impingement cooking systems. To achieve the overall goal, the specific objectives were:

1. To add color prediction capabilities to an existing finite element model.
2. To develop an alternative process model for moist air impingement cooking of meat patties by using artificial neural network (ANN) to replace an existing finite element model.
3. To identify the best strategy for process optimization among many combinations of optimization algorithms, process models, and parameterization of control functions.
4. To apply the developed optimization strategies to three case studies: single-stage, double-stage, and multi-zone oven systems.
5. To examine the performance of the optimization strategy for a single-stage oven system, given an economic-based objective function.

2 LITERATURE REVIEW

2.1 Achieving Extrema in Food Processing

An ultimate goal of most production activity is to increase profitability at given conditions, and food processing is no exception. Typically, improving the efficiency, reducing process time, and increasing yield and quality are concerns for most food processors and equipment designers. Generally, a typical industry consists of management, process design and equipment specification, and plant operations (Edgar *et al.*, 2001). Because these components are inter-connected, achieving those improvements are not simple tasks. Therefore, depending on the level of complexity and difficulty, the scope of a problem can be the entire enterprise, a plant, a process, a single unit operation, a single piece of equipment in that operation, or any intermediate stage between these (Beveridge and Schechter, 1970).

Generally, these improving activities are maximizing the capabilities of existing facilities or equipments by changing their conventional operating policies, conditions, numbers, and so on. All these attempts can be described in a single word: "optimization." A more formal definition would be "the collective process of finding the set of conditions required to achieve the best result from a given situation (Beveridge and Schechter, 1970)."

2.2 Characteristics of Optimization in Food Processing

Food processing is unique in that the process involves materials having irregular shapes, non-homogeneous compositions, and individual variance even in the same material. In addition to the materials, quality factors for product evaluation can be

subjective, such as taste, aroma, and flavor. Also, food processing encompasses various techniques, such as frying, baking, boiling, blanching, fermenting, drying, and so on. Therefore, modeling food processing phenomena is very challenging.

Models are essential components of modern process systems engineering (*i.e.*, simulation, optimization, and control), and they are usually classified into three categories, which are first-principle models (or white-box), data-driven models (or black-box), and hybrid models (or gray-box) (Banga *et al.*, 2003). Without adequate models, it is impossible to carry out optimization. From the view point of the application of optimization technique, the first-principle models are highly desirable, because the response time of the models is short, and it is convenient to apply various mathematical operations, such as differentiation. However, because first-principle models are difficult to obtain, data-driven models and hybrid models are popular in food processing. Generally, mathematical modeling of a food process requires knowledge of transport phenomena and reaction kinetics. Transport phenomena involve heat, mass, and momentum transfer into a food, and reaction kinetics cover degradation or inactivation of nutritional and organoleptic factors or microbial and enzymatic activity (Oliveira and Oliveira, 1999). Usually, these multi-physical phenomena are expressed as sets of algebraic, partial, and ordinary differential equations in the mathematical model. Due to the lack of theoretical methods for solving those highly complex, nonlinear systems of equations, most of the problems are solved by using numerical techniques, such as the finite difference method and the finite element method. Considering that most optimization techniques require numerous iterations of a process model, the numerical,

computational requirements of the model are often significant impediments to optimization.

One of the popular methods to overcome the above difficulties of modeling is alternative modeling, which can be fast, simple, and robust, with reliable accuracy. Even though alternative models sacrifice some degree of accuracy, compared to numerical models, the overall benefits of the alternative modeling techniques must be considered when evaluating performance in an actual optimization problem.

Artificial neural networks (ANN) have emerged as a potential alternative to physical-based models for food process engineering, because of their simple structure, robustness, no requirement of prior knowledge, and adaptive performance (Torrecilla *et al.*, 2004). ANN have been successfully applied to the modeling of food processes (Mittal and Zhang, 2000; Sablani and Shayya, 2001; Chen and Ramaswamy, 2002; Chen and Ramaswamy, 2003; Horiuchi *et al.*, 2004; Torrecilla *et al.*, 2004), property and quality prediction (Berg *et al.*, 1997; Xie and Xiong, 1999; Raptis *et al.*, 2000; Albert *et al.*, 2001; Therdthai and Zhou, 2001; Tominaga *et al.*, 2001; Hussain *et al.*, 2002; Boillereaux *et al.*, 2003; Ganjyal *et al.*, 2003), machine vision and image analysis (Chao *et al.*, 2002; Marique *et al.*, 2003; Diaz *et al.*, 2004), extrusion (Ganjyal and Hanna, 2002), microbial growth and inactivation (Geeraerd *et al.*, 1998; García-Gimeno *et al.*, 2003), high-pressure processes (Torrecilla *et al.*, 2005), and fluid flow (Adhikari and Jindal, 2000; Sablani and Shayya, 2003; Singh and Jindal, 2003). Once an ANN is established, the computation time of the network is very small with reliable accuracy. ANN is ideal for system identification and replacement of an existing first-principle model. For example, it

was shown that ANN were very effective for replacing a finite difference computer simulation of retort processes (Chen and Ramaswamy, 2002).

Another difficulty in food processing optimization arises at the characteristics of the response space of a problem. If the response space has just a single unique maximum (or minimum), finding the extremum can be guaranteed. However, food processing models are generally characterized as a system of nonlinear partial differential algebraic equations, which usually exhibit a multimodal nature (Banga *et al.*, 2003). In this situation, an effective and systematic procedure (or algorithm) is essential for optimization.

2.3 The State of the Art of Optimization Techniques

In the previous section, characteristics of food processing were discussed by focusing on model related issues. However, to solve practical optimization problems, effective techniques that are capable of consistently finding the best solution to the problems must be available.

A popular optimization technique is nonlinear programming (NLP, Section 3.1.2), which is usually using gradient information to decide the search direction (*e.g.*, steepest ascent path). If NLP is applied to highly nonlinear, constrained, and multimodal problems, it usually converges to the "nearest" local solution[5], because its search direction and size are determined from its starting point (Edgar *et al.*, 2001). Therefore, NLP cannot guarantee a global solution[6], if the starting point is not close enough to the global

[5] refers to local maxima or minima in a section of the entire solution space

[6] refers to the highest or the lowest point among other local maxima or minima.

optimum. However, the local nature of NLP algorithms can be overcome by global optimization (GO, Section 3.1.2.5) techniques designed to find a global solution.

GO algorithms are designed to escape from local solutions and explore promising regions where a global solution may exist. Also, GO can be utilized with ANN in parallel, without any modifications to the process model. In food process engineering, GO has been used with numerical models, such as finite difference models (FDM), finite element models (FEM), and ANN (Chen and Ramaswamy, 2002). Banga *et al.* (2001) and Zorrilla *et al.* (2003) coupled a FDM model of double-sided cooking of meat patties directly with the Integrated Controlled Random Search for Dynamic Systems (ICRS/DS), which is an adaptive stochastic GO algorithm.

Response surface methodology (RSM[7]) is currently the most popular optimization technique in food science, because of its comprehensive method, reasonably high efficiency, visualization, and simplicity, even though RSM is inefficient and cannot be automated in finding the overall optimum (Arteaga *et al.*, 1994). Also, Banga *et al.* (2003) pointed out some important drawbacks of RSM methods, such as the empirical, local, and stationary nature of these statistical techniques. RSM has uncertainty of model equations, which means the model might not represent a real physical model, because RSM is a statistically designed experimental optimization method. Generally, the method is not considered as a formal optimization technique. However, RSM has been

---

[7] RSM uses quantitative data from an appropriate experimental design to determine and simultaneously solve multivariate problems. The equations describe the effect of the test variables on the response, determine interrelationships among test variables, and represent the combined effect of all test variables in the response. This approach enables an experimenter to make efficient exploration of a process or system (Ponciano S. Madamba, "The response surface methodology: an application to optimize dehydration operations of selected agricultural crops", *Lebensm.-Wiss. u.-Technol.*, v. 35, p. 584)

successfully applied to product development and static process identification when the system behavior or the process is unknown, complicated, and static.

2.4 Applications in Food Process Engineering

2.4.1 Sterilization of Canned Product

Optimization techniques have been rigorously applied to sterilization of prepackaged conduction-heated food, such as retorting of canned foods. The basic form of this application is to find the best combination of retort temperature and process time for a constant heating process or the best retort time-temperature history for optimal control, which can simultaneously achieve the required lethality of the target microorganism and maximize quality factors.

The first attempt by Teixeria *et al.* (1969) found the best combination of retort temperature and process time for a constant retort process of conduction-heated foods. The best set of time and temperature was found by plotting thiamine retention against equivalent process conditions producing the same level of lethality. Even though the attempt introduced optimization concepts to food process engineering, the study did not apply formal optimization methods to the problem.

Saguy and Karel (1979) applied Pontryagin's maximum principle (PMP) to maximize thiamine retention in retort process and found a single optimal variable retort temperature profile. The significance of this study was the first application of formal optimization theory to food process engineering. However, because the application of PMP requires quite a modification of the original problem, it might not be suitable for a complex problem.

Numerous attempts have been made since the work of Teixeira *et al.* (1969) to optimize retort operation. However, the essential features of those optimization problems have many similar aspects (Table 2.1). The popular objectives of sterilizing prepackaged conduction-heated foods were maximizing retention of a single nutrient, minimizing quality degradation, and minimizing energy consumption. Most of the prior studies dealt with microbial lethality as an inequality constraint and a two-dimensional numerical model (heat conduction only) as an equality constraint.

Although there was similarity in the formulation of the problem among these many studies, the major difference among them was the method used to find the optimal condition or profile. Application of formal optimization methods has been increasing, so that several studies (Saguy and Karel, 1979; Nadkarni and Hatton, 1985; Banga *et al.*, 1991; Chalabi *et al.*, 1999; Kleis and Sachs, 2000; Erdoğdu, 2002; Erdoğdu and Balaban, 2003) found optimal retort solutions, with respect to specific assumptions. The type of optimal solutions can be categorized into: (a) combinations of process time and constant temperature, (b) piecewise continuous temperature profiles, and (c) on-off type of control profiles. Even though the piecewise continuous temperature profile is the true optimal solution, the solution cannot always be considered as the best, because it is not practically possible to implement a continuous profile in many conventional processes.

The biggest advantage in applying process optimization to retorts is the relatively simple process model, as compared with unpackaged food processes. Usually, the retort optimization encompasses a single control variable, temperature, which limits the burden of computation. Also, the process model does not involve mass transfer, which is a very complex phenomenon in meat cooking process.

Table 2.1 Analysis of basic elements used to optimize sterilization processes of prepackaged conduction-heated foods.

References	*Objective function* [a]	*Constraints*		*Optimization method* [d]	*Optimal Solution* [e]
		IC [b]	*EC* [c]		
Teixeira *et al.* (1969)	Max. (thiamine retention)	VAL	2D FDM	GS	(T, t)opt.
Teixeira *et al.* (1975)	Max. (thiamine retention)	VAL T_L / T_U	2D FDM	GS	f(T,t)opt.
Saguy and Karel (1979)	Max. (thiamine retention)	VAL T_L / T_U	2D FDM	PMP	Piecewise continuous profile
Ohlsson (1980)	Min. (surface & volume averaged cook value)	Fc	2D FDM	Comparison (diagrams)	(T, t)opt w/ linear come-up-time
Barreiro *et al.* (1984)	Min. (energy consumption)	Fc	Chart (Unsteady-state heat conduction)	Comparison (time-temperature)	Constant time & temperature
Nadkrani and Hatton (1985)	Max. (nutrient retention)	VAL T_L / T_U	2D numerical solution	PMDP	On-off control
Banga *et al.* (1991)	Max. (nutrient and surface quality retention) Min. (process time)	VAL Fc	2D FDM	ICRS/DS	VRT
Silva *et al.* (1992)	Max. (surface quality retention)	Fc T_{end} t_{end}	1D FDM	Davis, Swann and Campey method	Step function w/ linear come-up-time
Chalabi *et al.* (1999)	Max. (nutrient retention)	Fc	2D Theoretical 2D FDM	Open-loop optimal control	Bang-bang control
Kleis and Sachs (2000)	Max. (vitamins retention) Min. (energy consumption)	Fc T_L / T_U T_{end}	1D FEM	SQP	f(T,t)
Erdoğdu (2002)	Max. (thiamine)	Fc T_L/ T_U T_{end}	1D/2D FDM	Complex method	Equidistant ramp function
Erdoğdu and Balaban (2003)	Max. (thiamine of two different objects) – multi-objective	Fc T_L/ T_U T_{end}	FDM	Complex method Weighting method	Piecewise continuous profile

IC: inequality constraints; EC: equality constraints; T_L: lower temperature limit; T_U: upper temperature limit.

[a] Max., maximizing; Min., minimizing

[b] VAL, volume averaged lethality; Fc, critical lethality

[c] FDM, finite difference method; FEM, finite element method

[d] GS, graphical search; PMP, Pontryagin's maximum principle; PMDP, Pontryagin's minimum distributed principle; ICRS/DS; Integrated Controlled Random Search for Dynamic Search; SQP, Sequential quadratic programming

[e] VRT, variable retort temperature; opt., optimum

Chen and Ramaswamy (2002) developed an ANN model with training and testing data produced by a finite element model, and coupled the ANN model with GA. In that study, the control function (*i.e.*, retort temperature) was parameterized with sine and exponential functions to replace the traditional constant retort process. The coupled ANN-GA model was able to identify the relationship between the operating variables and control function parameters. Even though the ANN-GA found the optimal processing condition of a retort process, the optimal solution could be improved if additional parameters for the control function were used to increased flexibility. Also, the work of Chen and Ramaswamy (2002) was for prepackaged food retorting, a simpler process model than convection cooking of meat patties.

2.4.2 Food Dehydration

Dehydration is a common method to extend the shelf life of foods. This operation is normally removing water in a foodstuff via evaporation or sublimation (Brennan, 1990). Among many techniques, such as freeze drying, spray drying, super-heated drying, infrared drying, microwave drying, heated air drying of a solid food block is the focus of this review, because of the physical similarities with meat patty cooking under moist air impingement cooking. Using heated air is the typical method of dehydration, in which a food product is placed in contact with a moving stream of heated air. Therefore, drying can be an optimization problem, in which the optimal conditions are sought to maximize retention of nutrients, such as ascorbic acid, while minimizing enzyme or microbial activities (Banga and Singh, 1994).

Drying processes are generally driven by evaporation at the surface, which causes water transport within a foodstuff. Fick's equation of diffusion is often used to describe

water transport in drying. For hot air drying, the mass transfer model must be coupled with a heat transfer model, often assuming a one-dimensional thin slab and an evaporation term to account for the energy balance. These partial differential equations are usually solved by the finite difference method or rarely by existing analytical solutions with empirical equations for model parameters, such as diffusion coefficient, convective heat transfer coefficient, and first-order rate constant for ascorbic acid degradation (Mishkin *et al.*, 1982). Food drying quality parameters are often modeled by using first-order reaction kinetics (Mishkin *et al.*, 1983; Banga and Singh, 1994).

Mishkin *et al.* (1982) used Pontryagin's maximum principle (PMP) and the complex method to find the optimal air temperature profile maximizing ascorbic acid retention in a model system (a slab composed of water, cellulose, and ascorbic acid) with fixed relative humidity. The complex method was selected, because the method was convenient to use along with any type of process model and constraints without modification. Mishkin *et al.* (1983) extended their research to a multi-stage drying process. They found optimal stepwise temperature and humidity profiles for three stages by using the complex method.

Banga and Singh (1994) set up four different optimization problems for drying of a thin slab of cellulose: (a) maximizing ascorbic acid retention with a constraint on the final moisture content, using air dry bulb temperature as the control variable; (b) minimizing process time with final retention of ascorbic acid, using air dry bulb temperature control; (c) maximizing ascorbic acid retention with final retention of enzyme, using dry bulb temperature and relative humidity control; (d) maximizing energy efficiency with final ascorbic acid retention, with dry bulb temperature control. These

problems were solved by using an ICRS/DS algorithm, and they found optimal piecewise linear control profiles in all cases.

In spite of many similarities with meat cooking processes under moist air condition, drying processes are different from meat cooking, in that cooking involves more complex mass transfer phenomena (water-fat mixing and fat dripping), microbial inactivation, phase change, and air humidity sufficiently high to cause condensation.

2.4.3 Meat Patty Cooking

The abundance of prior research in retort operation is due to the availability of the process model, which encompasses relatively simple phenomena, such as depletion of certain nutrients coupled with heat transfer. However, the nature of meat patty cooking is an unpackaged process that generally involves various mass transfer phenomena, such as evaporation and dripping of fat and water. In addition, geometry change and phase transition is typical for this process. These phenomena must be coupled with heat transfer and solved. Other difficulties in the modeling of meat patty cooking arise in the heating medium. The heating medium in retort processes is water or steam, which does not interact with the food material in the package and has simple thermal properties. However, in an unpackaged food product, the heating medium interacts with the surface of the food material, which therefore leads to more elaborate boundary conditions.

Banga *et al.* (2001) and Zorrilla *et al.* (2003) applied the dynamic optimization technique (Section 3.1.2.6) to contact cooking of meat patties, which is considered as the first attempt to optimize a meat patty cooking process. The objective of those studies was to minimize cooking loss of patties, while ensuring inactivation of *E. coli* O157:H7 and final product center temperature. One-dimensional coupled heat and mass transfer solved

with finite difference method (Pan, 1998) was used as a process model. The above optimization problem was solved by a global optimization algorithm, ICRS/DS (Integrated Controlled Random Search for Dynamic Systems), which found the optimal piecewise step grill surface temperature profile (Banga *et al.*, 2001; Zorrilla *et al.*, 2003).

The process model is an important part of an optimization problem, because the model is used to predict the objective value and constraints. The process model developed by Pan, (1998) and used by Banga *et al.* (2001) and Zorrilla *et al.* (2003), assumed the patty as a one dimensional infinite slab, which is less accurate than two-dimensional modeling. Even though the model could predict water and fat transfer and microbial log reduction, other important quality factors, such as internal color change and surface color change, were not considered. Also, the nature of the cooking method was contact cooking, which was modeled using simplified, effective boundary conditions, compared with the condensing-convective boundary conditions during meat patty cooking under a moist air environment. In their optimization problem, grill surface temperature was the single decision variable. However, moist-air impingement meat patty cooking involves the additional control variables of air humidity, impingement velocity, and impingement geometry. Even though the performance of ICRS/DS was good enough to locate the global optimum for the contact cooking problem, comparison with other GO methods was not conducted.

2.4.4 Various Processing Areas

In addition to the application for canning, cooking, and drying, optimization techniques have been applied to the other processes, such as ultra-filtration, baking, extrusion, cheese manufacturing, mixing, and so forth. Each application provides some

lesson about the implementation of formal optimization techniques to the unique nature of various food processes.

Usually, the objectives of wine filtration are to minimize colloid content, maximize color intensity, and maximize flux by varying pore size and the recycle rate of membranes (Gergely *et al.*, 2003). Gergely *et al.* (2003) expressed the objective function as a second-order form of regression functions of the membrane pore size and recycle flow rate, which was a response surface model (RSM). Even though the RSM can solve some optimization problems, generally the method is not recognized as a formal optimization method (Section 2.4).

In optimizing commercial bread baking, the biggest challenge is getting a reliable process model. Therdthai *et al.* (2002) used four different temperature zones of a commercial oven and baking time to find optimal condition for minimizing loss while controlling the top crust color, side crust color, and average crust color within acceptable ranges. Statistical methods were used to construct a model equation, which was a RSM. However, some researchers have used neural networks coupled with a Genetic Algorithm (GA) for leavening process optimization in a bread-making industrial plant (Fravolini *et al.*, 2003). Fravolini *et al.* (2003) used a nonlinear system identification method, called NARMA (Nonlinear Autoregressive-Moving Average), to model the leavening process.

Extrusion is very complex process. The most common objectives are expansion ratio, shearing strength, and sensory texture, which are functions of temperature, feed moisture, and process variables, such as screw speed, screw compression ratio, feed speed, and die diameter. RSM is the most popular method to model the process and to apply optimization methods, because the results of RSM are analytically differentiable

mathematical expressions (Chávez-Jáuregui *et al.*, 2000); Frazier *et al.*, 1983; Iwe *et al.*, 1998; Karwe and Godavarti, 1997; Olkku *et al.*, 1983; Quintero-Ramos *et al.*, 1998; Vainionpaa, 1991). Therefore, optimal condition could be found by applying theoretical optimization techniques directly to the model.

In addition to applications for food processes, well-organized dynamic optimization problems also exist in biochemical processes, such as fermentation (Tartakovsky *et al.*, 1995; Banga *et al.*, 1997; Berber *et al.*, 1998; Tsoneva *et al.*, 1998; Faqir, 1998);Lee *et al.* 1999; Radhakrishnan *et al.*, 1999; Halsall-Whitney *et al.*, 2003; Levisauskas *et al.*, 2003). The nature of biochemical processes is similar to food processes, in that their dynamic behavior is inherently nonlinear. However, most biochemical systems can be modeled by a set of ordinary differential equations, which makes many optimization theories applicable, because the model equations are differentiable. Generally, biochemical processes have been treated as problems of optimal control, which use a special form of the performance function.

3 THEORIES

3.1 Overview of Optimization Theory

3.1.1 Introduction

3.1.1.1 What is Optimization?

Our daily life is always full of choices in various activities, such as traveling, shopping, or eating. Decision-making is the cognitive process of selecting a course of action from among multiple alternatives. The purpose of decision-making is to choose the best alternatives fitting with our goals. Even though the purpose of decision-making is clear, making decisions involves many considerations and uncertainties. In the case of a simple problem, we can guess the results of some trials or test the effect of each possible alternative. However, as the number of variables and the interactions between variables increase, these attempts lose the ability of identifying the best solution among the many possible good solutions. For example, we can increase the thickness of insulation to decrease energy loss, but increased insulation thickness increases cost, which is the trade-off. In that case, the problem is to find the thickness of insulation minimizing the total cost, which cannot be solved easily.

Therefore, systematic methods and procedures are necessary to solve those problems containing many variables, restrictions, and factors, which compete with each other for the best solution. Optimization can be defined as "the collective process of finding the set of conditions required to achieve the best result from a given situation" (Beveridge and Schechter, 1970). By the help of optimization techniques, we can explore more complex decision-making situations with more certainty and effectiveness.

3.1.1.2 Essential Features of Optimization Problems

By observing optimization problems, some common features can be found. For example, if a driver wants to travel from city "A" to city "B" in the least time, then the minimum traveling time will be the primary objective for the driver. To accomplish this goal, the driver has to consider the speed limit of each state, weather conditions, physical limits of driver and car, paths, and so on.

The basic components of an optimization problem are the objective function, decision variables, constraints, and mathematical model. These components are well explained by Evans (1982).

1. Objective function (performance function, or cost function): This is the quantity to be maximized (or minimized). It is often referred to as the cost or performance function. Whether measured in dollars, efficiency, or other terms, the performance function evaluates alternative solutions to the problem to determine which one is the best.
2. Decision variables: These are the parameters in the process or system that can be adjusted to improve the objective. They are the free or independent variables that must be specified in the traditional case-study approach to problem-solving.
3. Constraints: All optimization problem have constraints on the allowed solutions. These constraints may limit values of the decision variables or of other dependent variables that describe the behavior of the system.
4. Mathematical Model: If the problem is to be solved other than by trial-and-error physical experimentation, we must have a model of the system. The model is the mathematical representation of the system that determines the objective function

in terms of the decision variables. It also determines other dependent variables that may be subject to constraints.

Now, we can define optimization by using the terms described above. Therefore, optimization is finding a set of conditions maximizing or minimizing the objective function while satisfying the imposed constraints of the problem.

3.1.1.3 Solving Optimization Problems

To solve an optimization problem, one must analyze the essential features of problem. This means that the objective and constraints of the problem should be identified first, but not necessarily in the form of mathematical expressions. Then the objective function must be expressed in terms of the system variables by using mathematical expressions. Interrelationships, internal restrictions, and system models must be expressed in mathematical form. Now, the problem is reconstructed according to the essential components of a formal optimization problem. Upon this mathematically redefined optimization problem, an appropriate optimization method and algorithm can be applied to obtain the optimal conditions to achieve the objective.

3.1.1.4 Hierarchy of Optimization Problems

Optimization can be employed at any level in a plant, ranging from a small piece of equipment to management of a whole company. Someone may need more workers to maximize the production rate. However, from the management perspective, this optimization result might not be favorable, because of labor cost. Therefore, the scope of optimization is important, because one level of optimization does not guarantee the

optimality of another level of a system. Typical industrial company optimization encompasses three levels: management, process design and equipment specification, and plant operation (Edgar *et al.*, 2001). The highest level of optimization of a company is the management to maximize net profit. To accomplish the objective, someone would need to model the whole company, including every piece of equipment, which is practically impossible. Therefore, the scope of an optimization problem must be reviewed before proceeding to the next step.

3.1.2 Theory and Methods

3.1.2.1 Basic Concepts of Optimization

The basic concepts of optimization can be clearly illustrated by observing an unconstrained one-dimensional case. "Unconstrained" means that there are no equality or inequality constraints, so that the independent or dependent variables can be simulated without any limitations. Let's take a simple arithmetic example. If we define a quadratic function y with an independent variable x and coefficients a, b, and c, then:

$$y = f(x) = ax^2 + bx + c \qquad [3.1]$$

The objective is to find x minimizing or maximizing the function value. One might plot the function to see the actual shape of the curve and find a maximum or minimum point graphically (Figure 3.1). However, there is a mathematical tool to see the feature of the curve, which is the first derivative (y') information obtained by differentiating the above equation with respect to the variable x. This is shown in Equation [3.2].

$$y' = \frac{df(x)}{dx} = 2ax + b \qquad [3.2]$$

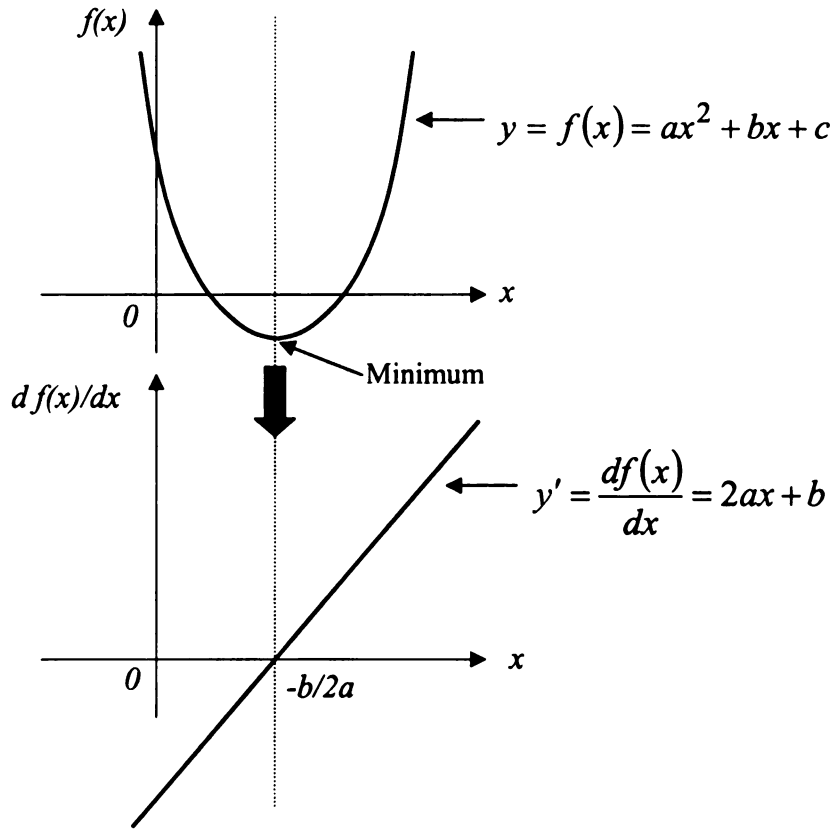


Figure 3.1 Locating minimum point of a quadratic function with first derivative information.

This first derivative information implies that there is an extreme point between the sign changes of the gradient (Equation [3.2]), which is an inflection point. The inflection point (stationary point) of zero gradient can be calculated by setting the right hand side of Equation [3.2] equal to zero. Thus, the stationary point is $x = -b/2a$. If the x value is inserted to the Equation [3.1], then we can have an extreme value of $y = (-b^2 + 4ac)/(4a)$. However, we still do not know whether the value is maximum or minimum if we do not have graphical information. To determine this, the second derivative information of the Equation [3.1] is necessary. The second derivative (y'') is:

$$y'' = 2a \qquad \textbf{[3.3]}$$

If the second derivative is negative, then the extreme value is a maximum or vice versa. Hence, a necessary condition for a minimum or maximum of $f(x)$ is that the first derivative of $f(x)$ becomes zero at x. However, the necessary condition does not tell whether the extremum is the minimum or maximum. So, we need the second derivative information as a sufficiency condition. Now, we can identify a minimum or maximum of $f(x)$ mathematically with the necessary and sufficient conditions.

The above case is very simple, but the basic concepts can be extended to the multivariable cases with some help of mathematical techniques. Let $f(x_1, x_2,...,x_n)$ be an n-dimensional function. To meet the necessary condition, the first derivative of each variable $x_1, x_2...x_n$ must be zero as follow.

$$\frac{\partial f(x)}{\partial x_1} = \frac{\partial f(x)}{\partial x_2} = \cdots = \frac{\partial f(x)}{\partial x_n} = 0 \qquad \textbf{[3.4]}$$

A set of variables satisfying the set of Equations [3.4] represents an extreme point. The sufficient condition for the extreme point can be checked with the nature of the matrix of second partial derivatives (Hessian matrix) of $f(x_1, x_2,...,x_n)$ at the point, which is the extension of Equation [3.3] to the multivariable case. If all the objective functions of our problems were quadratic and differentiable, then life would be simple. However, this analytical method has many limitations in real life applications. Therefore, we need a more general approach to solve optimization problems.

Before addressing this issue, imagining a movie scene will give us insight for the generalized optimization method. Let's imagine a situation that two special soldiers (captain "A" and captain "B") are dropped from an airplane into an enemy region in a night having no moon light (Figure 3.2). Their mission is to find the highest point from sea level and communicate with the nearby resistance. Their landing location is different from each other, so that they have to complete the mission individually. Captain A has the map of the region, compass, and GPS (global positioning system), but captain B lost his equipment because of a tough landing. Captain A saw the map and located the highest point and also his current location with GPS. Then he set the direction toward the highest point and finally got to the point. However, captain B does not have any information or equipment, except a small flashlight in the complete darkness, which means he only can get terrain features a few yards around him. How could he get to the highest point? So, he marked his current location on the ground and looked around with the flashlight. He found that the steepness of terrain was increasing in the northeast direction. So, he kept moving a few yards from his current location to the next location in this direction if there was an increase of elevation. He repeated this strategy until he could observe no more

increase of elevation. Fortunately, he reached the same location where captain A arrived, because there was only one peak in the region. However, if there were several peaks, it could take more time or he might be stranded at a sub-peak.

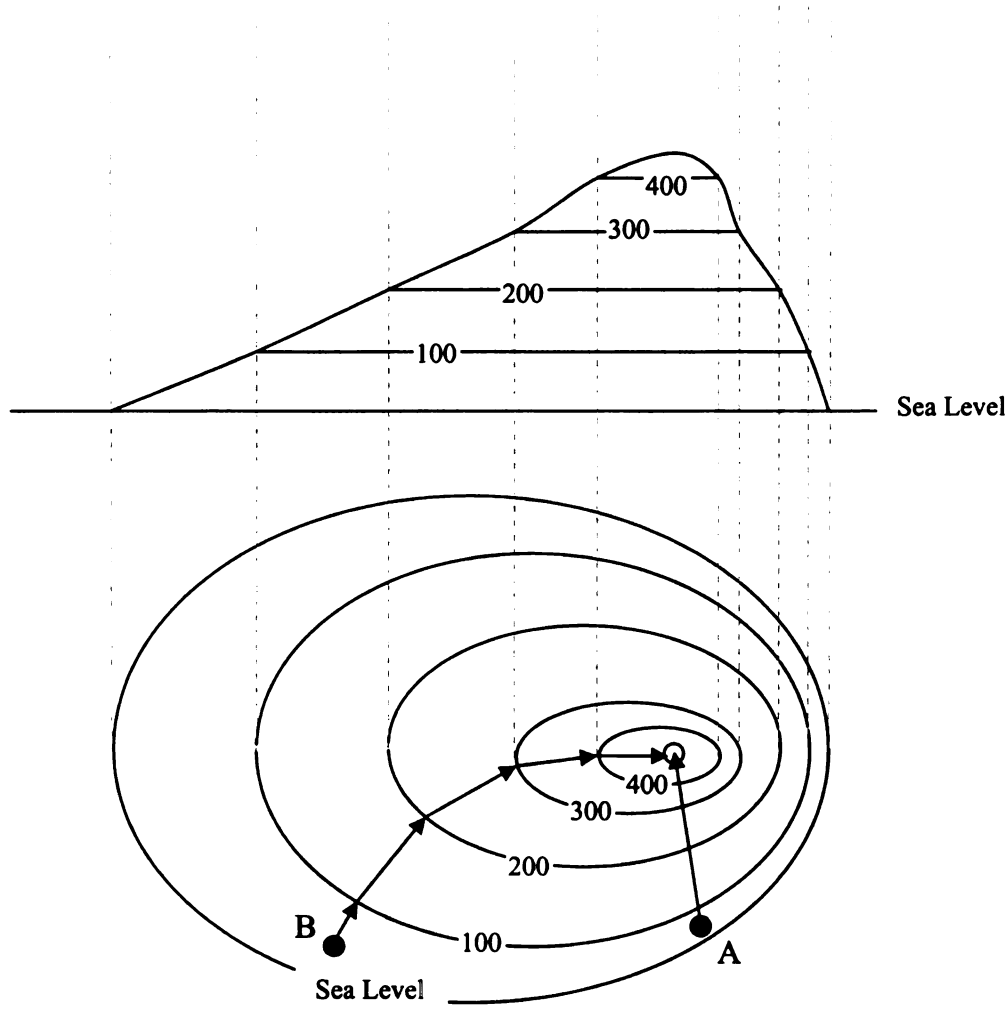


Figure 3.2 Illustration of the operation of captain "A" and captain "B".

In the above story, our focus is captain B's approach, because his situation is very similar to ours, like finding a maximum or minimum within a region that we cannot

figure out, which means no analytically differentiable mathematical expressions. Captain B's approach gives us a clue for setting up a general rule for finding such an extreme point in an unknown region. There are three factors to determine the next point from the current point. Current point (x^k), search direction (α^k), and step size (S^k) are needed to determine the next point (x^{k+1}). Also, the relationship can be expressed mathematically as following:

$$x^{k+1} = x^k + \alpha^k \cdot S^k \qquad [3.5]$$

The above iterative search procedure stops based on some criteria. If there is no significant improvement, the search stops. There are many methods to perform the search; however, the differences between methods are mainly in how they generate the search direction. A popular method is using the first derivative information. However, function value, and finite difference approximation are also used in lieu of derivatives (Edgar *et al.*, 2001). Table 3.1 shows some methods according to their approaches.

Table 3.1 Unconstrained multivariable optimization methods categorized by how they generate the search direction (Edgar *et al.*, 2001;Venkataraman, 2002).

Function values only (Direct Method)	**Gradient based (Indirect Method)**
Random Search	Steepest Descent
Grid Search	Conjugate Gradient
Univariate Search	Davidon-Fletcher-Powell Method (DFP)
Simplex Search	Broydon-Fletcher-Goldfarb-Shanno Method (BFGS)
Conjugate Search	

Until now, we discussed handling unconstrained function optimization. However, the real life optimization problem always has equality or inequality constraints. The constraints have to be handled so that the constrained optimization problem can be converted into an unconstrained optimization problem. The basic idea is to set the

objective function free from constraints by using mathematical manipulations of the constraints.

Let us first look at the handling of equality constraints. Three methods are mainly used for the solution of such problems: direct substitution, constrained variation, and Lagrange multipliers.

Direct substitution is to substitute equality constraints to the objective function when the equality constraints are all linear equations. Then, the constraints vanish, and the problem can be handled as an unconstrained case.

Constrained optimization subjected to inequality constraints is treated as in the case of equality constrained problem after transformation of inequality constraints to equality ones by introducing a slack variable, which is a buffer between the original inequality constraint and the transformed equality constraint.

Therefore, a general approach to solve a constrained optimization problem is to convert the constrained problem into an unconstrained problem. Then, unconstrained multivariable optimization techniques (Table 3.1) can be used to find optimal solution.

3.1.2.2 Linear Programming

A linear programming[8] (LP) problem is one in which the objective and all of the constraints are linear functions of the decision variables, so that the linear constraints (lines in 2-dimensional case) form boundaries. Therefore, an objective function, represented as a line, is moving through the bounded region to find an optimal set of variables to get an extreme value in a two-dimensional case (Figure 3.3).

---

[8] "The word *programming* here does not refer to computer programming, but means optimization." Edgar, T. F., D. M. Himmelblau and L. S. Lasdon (2001). Optimization of chemical processes. New York, McGraw-Hill.

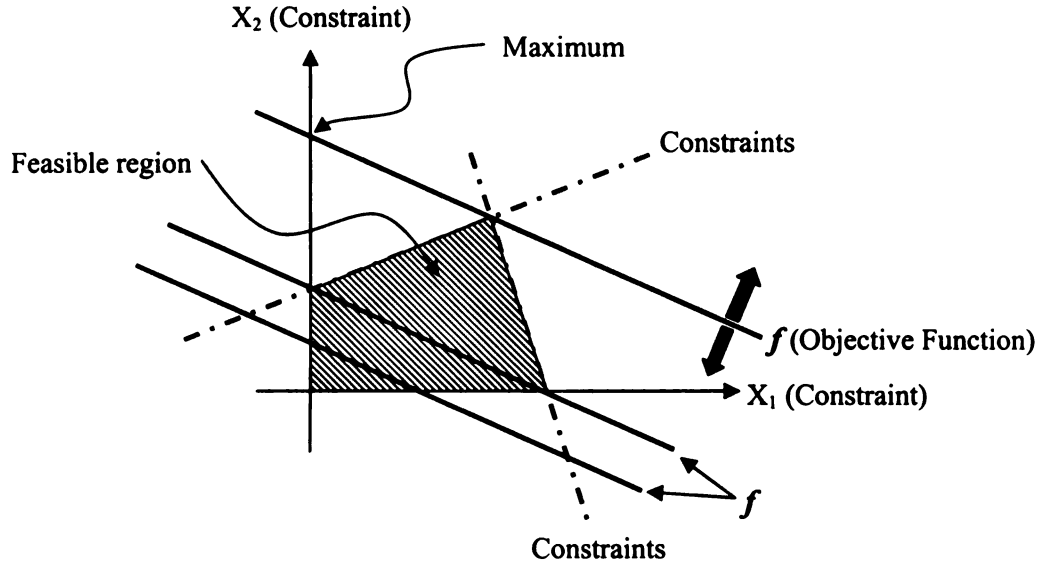


Figure 3.3 Concept of linear programming problem in a two-dimensional case.

One of the important characteristics of LP is that the extremum of a linear program always occurs at a vertex or corner of the system boundaries, which is the intersection of constraints in the feasible region (Beveridge and Schechter, 1970). The basic idea is a systematic examination of these boundaries, which is converting a set of constraints into a set of equality equations and applying linear algebra and matrix manipulation (Venkataraman, 2002). For instance, the simplex algorithm is designed to explore one intersection after another to the direction of improving the objective function, according to a set of rules, until the best objective function attainable is found (Beveridge and Schechter, 1970).

3.1.2.3 Nonlinear Programming

Nonlinear programming (NLP) problems contain at least one nonlinear equation of the objective function or constraints (Venkataraman, 2002). No single optimization algorithm can possibly be efficient or even successful in all cases of interest. However, unique techniques are well developed for each specific case. For instance, a nonlinear objective function with linear equality constraints can be handled with the direct substitution method, which solves the objective function explicitly for one variable and eliminates that variable from the problem formulation (Edgar *et al.*, 2001). A nonlinear objective function with linear inequality constraints can be solved by using Kuhn-Tucker conditions[9] and Lagrange multipliers[10]. If an optimization problem has a quadratic objective function and linear inequality or equality constraints, quadratic programming (QP) can be used (Edgar *et al.*, 2001). Another strategy to solve nonlinear optimization problems is to replace all nonlinear functions in the problem with their Taylor series approximations and apply linear programming, which is called successive linear programming (SLP) (Edgar *et al.*, 2001).

The most robust and generally accepted nonlinear optimization technique is the generalized reduced gradient (GRG) method, which is also implemented as the "Solver" in the spreadsheet program Microsoft Excel (Edgar *et al.*, 2001). The concept of GRG is to reduce the dimension of the first derivative of the objective function by using the first

---

[9] At any local constrained optimum, no (small) allowable change in the problem variables can improve the value of the objective function (Edgar, T. F., D. M. Himmelblau and L. S. Lasdon (2001). Optimization of chemical processes. New York, McGraw-Hill.). This is a generalization of the Lagrange multiplier method.

[10] By introducing an unknown scalar variable to the constraints, a linear combination is formed, which reduces a constrained problem into an unconstrained problem (Venkataraman, P. (2002). Applied optimization with matlab programming. New York, John Wiley & Sons).

derivative of constraints. All major NLP algorithms are based on estimation of first derivatives of the problem to obtain a solution and to evaluate the optimality conditions (necessary and sufficient conditions) (Edgar *et al.*, 2001). Because NLP is mainly based on the gradient information, the solution has a local nature, which means the solution is a local maximum or minimum, but not necessarily the global one in general cases of nonlinear optimization problems (Figure 3.4).

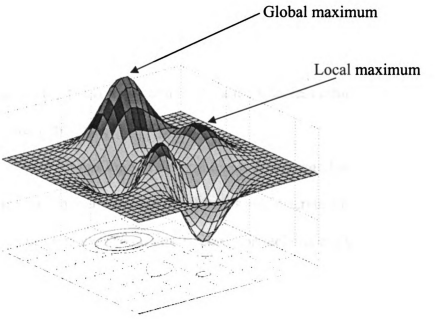


Figure 3.4 Global and local optimum in 2-dimensional case (The picture was adopted from the help manual of MATLAB®).

3.1.2.4 Mixed-Integer Programming

Another special type of optimization problem is mixed integer programming (MIP). As the name implies, this type of problem includes integer variables that are not continuous. The integer variable is common in plant operation, design, location, and scheduling (Edgar *et al.*, 2001). For example, number of equipment, yes-no decisions, and number of stages are integer variables. If the objective function and constraints are linear in a MIP, then it is called mixed-integer linear programming (MILP). And if the MIP involves nonlinear objective and constraints, then it is called mixed-integer nonlinear programming (MINLP) (Edgar *et al.*, 2001). If the number of integer variables is small, then exhaustive or complete enumeration is possible. However, the effort grows exponentially to examine all possible solutions (Venkataraman, 2002). Therefore, systematic ways are necessary to get to the optimal solution with less effort and time and also to handle large scale problems.

One popular solution for this kind of the problem is branch and bound (BB). The basic concept of BB is systematically finding the closest set of discrete variables from the optimum set of continuous variables. To do this, a relaxation technique is used to convert the discrete variables into continuous variables bounded by their maximum and minimum value. BB uses two strategies to find the best set of integer variables. Branching is the efficient way of covering the feasible region by several smaller feasible sub-regions, and bounding is comparing and selecting the sub-region having its upper bound that is less than the lower bound of any other sub-region.

3.1.2.5 Global Optimization

We have discussed unconstrained, constrained multi-dimensional cases, and some special cases, such as LP, NLP, MILP, and MINLP. One of the underlying assumptions for optimality is that the problem must be convex, which means there is only one minimum or maximum. Thus, if the convex condition is not met, then the above methods can be stranded at a local optimum, not at a global optimum (Figure 3.4). This situation frequently happens when the model has nonlinear equality constraints, such as a nonlinear material balance, nonlinear physical property relations, nonlinear process models, and so on (Edgar *et al.*, 2001).

In addition, if a gradient-based method is involved, then the search method can become stranded around the vicinity of local minima or maxima, without guarantee of optimality. However, in spite of this fundamental limitation, there are several effective and practical optimization techniques that explore infinite solution spaces systematically with minimum effort and high possibility of locating the global optimum. This type of optimization is called global optimization (GO).

In a GO problem, a local solution can be considered as a global optimum if the local solution is the best among many local solutions or if multiple search trials from different starting points reach to the same local solution. Widely used GO methods can be classified as deterministic (exact) or stochastic strategies (heuristic) (Banga *et al.*, 2003).

Deterministic global optimization methods are based on the systematic gradient-based local search methods following the systematically divided sub-regions of attraction until they meet their termination criteria. This type of method can guarantee global

optimum in certain problems, but not in a general GO problem. Branch-and-bound[11] methods, methods based on interval arithmetic (Kearfott, 1996), and multi-start methods are classified as deterministic GO techniques (Edgar *et al.*, 2001). Even though deterministic GO methods have sound theoretical convergence properties, the associated computational effort increases very rapidly (often exponentially) with the problem size (Banga *et al.*, 2003).

Stochastic GO methods do not follow systematically divided regions of attraction. Instead, they set their search direction based on a logic found in natural processes, such as cooling of metals and genetic evolution processes. Many stochastic GO methods can locate the vicinity of global solutions with relatively good efficiency, but the downside is that global optimality cannot be guaranteed (Banga *et al.*, 2003). Scatter search, tabu search, simulated annealing, and genetic and evolutionary methods can be classified as stochastic GO methods (Edgar *et al.*, 2001). Stochastic GO methods are applicable to almost any problem, without modification of the original process model (Edgar *et al.*, 2001). In the case of Genetic and evolutionary GO methods, the method produces a population that is a set of solutions and keeps updating the population with improved solutions according to the rule of biological processes of crossover and mutation. Although stochastic GO cannot guarantee a global optimum, it is widely adapted to solve real life problems, because of its relative simplicity and robustness.

---

[11] Branch and bound algorithms are a variety of adaptive partition strategies that have been proposed to solve global optimization models. These are based upon partition, sampling, and subsequent lower and upper bounding procedures: these operations are applied iteratively to the collection of active ('candidate') subsets within the feasible set D. Their exhaustive search feature is guaranteed in similar spirit to the analogous integer linear programming methodology (Eric W. Weisstein et al. "Branch and Bound Algorithm." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/BranchandBoundAlgorithm.html, March 22, 2005).

3.1.2.6 Static and Dynamic Optimization

In the previous sections, we have discussed how to solve optimization problems in each special case, especially when all the constraints are algebraic equations. In other words, the problem is not changing with time. For instance, finding the optimum diameter of a pressure vessel does not involve a time factor. However, what if the constraints have time varying equations, such as ordinary differential equations in which a variable is changing with respect to time? This kind of optimization problem can be called "dynamic" optimization, as opposed to the "static" optimization problem. To be precise, this type of problem is generally referred to as an optimal-control problem, which is the area of control. However, some optimization techniques can be employed to solve such problems.

There are two general approaches. The first approach is discretization of the control function, which can be understood as dividing the control function into pieces. This method is replacing a differential term by the first order Eulerian difference expression, which is:

$$\frac{dx}{dt} \approx \frac{x(t_i) - x(t_{i-1})}{\Delta t} \qquad \textbf{[3.6]}$$

Now the ordinary differential equation can be expressed as a set of algebraic equality constraints, which is the standard form of a constrained nonlinear programming technique.

The second method is parameterization of the control function. We need an infinite number of points to express a varying control function, which is not practical. However, if the control function is expressed with Fourier series or a simple polynomial

form, the infinite solution space can be explored with a finite number of parameters. By applying NLP, the optimal set of parameters can be obtained as the optimal control function, even though there is no guarantee of optimality.

3.2 Global Optimization Techniques

3.2.1 Genetic Algorithm (GA)

Genetic algorithms (GA) are well-known stochastic global optimization algorithms based on biological evolution theory. Holland (1962) was the first to use the technique, but its use as an optimization tool began in earnest in the late 1980s, developed momentum in the mid-1990s, and continues to attract serious interest today (Venkataraman, 2002). Genetic algorithms have been successfully applied to a wide range of problems, such as engineering design, scheduling, signal processing, optimal control, transportation, and so on.

The biological evolution theory is the combination of Darwin's theory of natural selection and Mendel's theory of genetics. According to the theory, a chromosome in a gene pool is modified by simple rules of genetics, such as crossover and mutation. The biological system having the gene that is the fittest among others to the environment survives and produces the next generation, having individuals with more desirable characteristics. These biological concepts are implemented in genetic algorithms via numerical operations. For example, a vector of design variables is considered as a chromosome in genetic algorithms. The following is a description of genetic algorithm datails; the explanations are largely adopted from the work of Venkataraman (2002).

Figure 3.5 shows the general procedures of a genetic algorithm. The initial population is constructed with a certain number of design vectors that take higher rank in

terms of a performance index or an objective function value, in the case of an unconstrained problem among randomly produced individuals[12]. For the constrained problem, these individuals must satisfy the specific constraints. The initial population size usually remains the same throughout the whole generation. However, there are no specific criteria to determine the optimal initial population size (Venkataraman, 2002).

Next generation candidates are now produced by using selected parents and genetic operators. A crossover genetic operator exchanges a piece of the chromosome of each parent. A simple crossover exchanges a piece of chromosome in the same location of each parent, and the ratio of the piece is generated randomly at each operation (Figure 3.6). Arithmetic crossover uses a linear combination of parent chromosomes to produce two children.

[12] This refers to a vector of design variables. A piece of the chromosome or a portion of a design vector is called as *allele*.

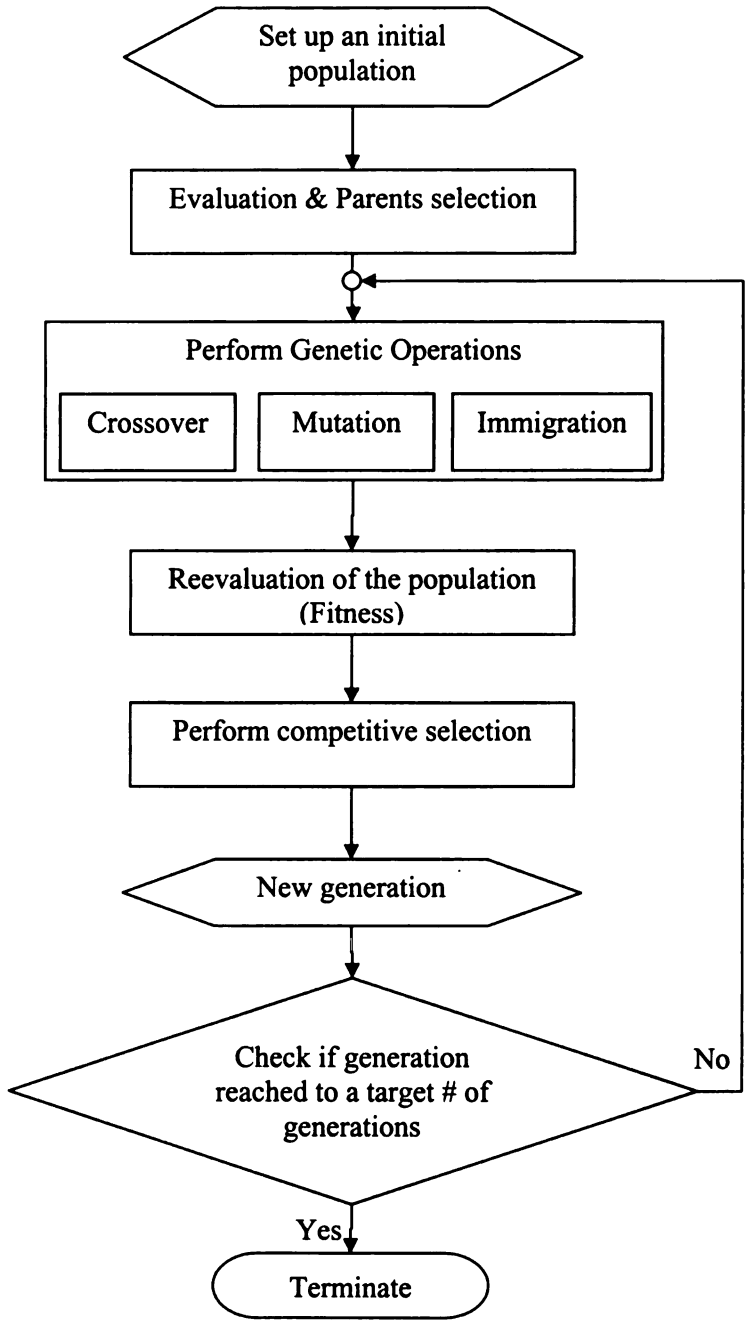


Figure 3.5 General procedure of a genetic algorithm.

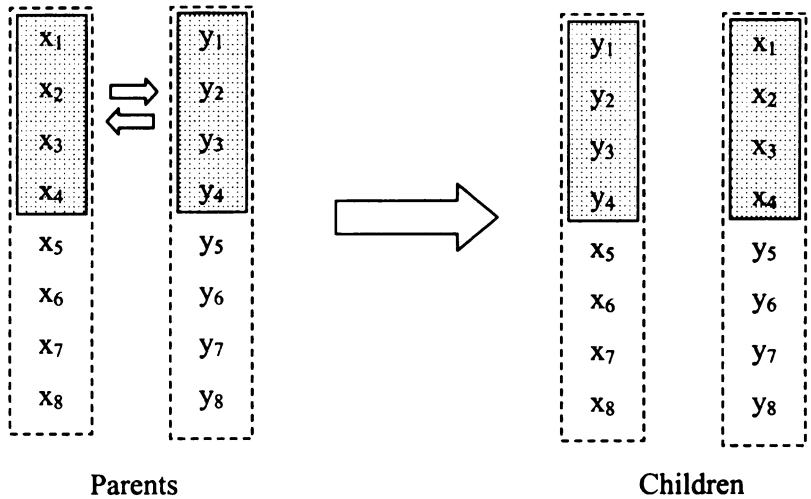


Figure 3.6 An example of simple crossover operation in a genetic algorithm.

The children are defined as:

$$C_1 = \lambda_1 \mathbf{X} + \lambda_2 \mathbf{Y}$$
$$C_2 = \lambda_2 \mathbf{X} + \lambda_1 \mathbf{Y} \qquad \text{[3.7]}$$

where, $\lambda_1 + \lambda_2 = 1;\ \lambda_1, \lambda_2 > 0$
X, **Y**: parents
C_1, C_2: children

The parameter λ_1 and λ_2 are generated randomly. Mutation selects a design variable randomly and replaces it with a randomly generated value. Generally, these genetic operators, crossover and mutation, narrow a search to a promising region that might have a local extremum (Venkataraman, 2002). However, it is hard for a global optimization algorithm to find a global optimum if it is stranded around a local solution. Another

feature of GA is immigration, in which a set of randomly generated, unbiased population is added to the existing population before the selection is made for the next generation. By using this feature, GA can reduce the possibility of being trapped in a local solution and effectively explore an entire solution space.

The newly generated population now must be evaluated to identify suitable parents for the next generation. A simple method of selecting parents is ranking or sorting the individuals of the population according to their objective function value. A fraction of the best individuals can be used for reproduction of the next generation. In a strategy called tournament selection, the remaining portion of the population, excluding the fraction of the best individuals, is fed back to the next generation as new immigrants.

The above genetic operation and fitness evaluation are performed until the generation number reaches a certain number or the solution does not show any improvement. GA is very useful for handling ill-behaved, discontinuous, and nondifferentiable problems, because the algorithm generates a possible solution group at each iteration, instead of a search direction, and is effective for handling continuous problem (Venkataraman, 2002).

3.2.2 Simulated Annealing (SA)

Annealing is a heat treatment technique to alter material characteristic by removing internal stresses and crystal defects. When a material is in molten state, atoms can move freely before they form crystal structure. However, if a molten material is cooled down rapidly, atoms lose chances to form crystal structure, which creates internal stresses and crystal defects. Therefore, the rapidly cooled material is in a higher energy

state than the material cooled slowly. The cooling schedule is also called the annealing schedule, which reduces the surrounding temperature from the critical temperature to a sufficiently lower temperature step-by-step. In each cooling step, the material being treated is allowed to cool in the furnace until it reaches thermal equilibrium with its surrounding temperature. By using the technique, internal energy of a material can be minimized, which means stable crystalline order. Usually, annealing is used to soften a material and make it more ductile, to relieve residual stresses, and to refine the crystal structure (Shigley and Mischke, 1989).

The above ideas can be applied to finding the global optimum in an optimization problem, which has become popular for combinatorial optimization[13] problems. The molten state of the material can be an initial design space in which every combination of variables has equal opportunity to be searched out to find the best combination. The objective is to reduce the internal energy level to the lowest state, which is crystalline order. As the annealing temperature goes down, the mobility of atoms decreases, and the total internal energy level goes down at the same time, which means that the search direction is being fixed to a direction without considering other opportunities. When the temperature reaches the final scheduled temperature, atoms form crystalline structure, which represents the lowest energy level, a global minimum.

The above analogy is realized by the basic SA procedure (Floquet *et al.*, 1994) summarized by Edgar *et al.* (2001) as follows:

- Choose an initial solution $\mathbf{x}$, an initial temperature T, a lower limit of temperature T_{LOW}, and an inner iteration limit L.

---

[13] A method to search for the best possible solution out of very large number of discrete feasible solutions.

- While ($T>T_{LOW}$), do
    - For $k = 1, 2,\ldots, L$, do
    - Make a random choice of an element $\mathbf{x}' \in N(\vec{x})$, where $N(\mathbf{x}) = \mathbf{x} + \alpha \cdot S$ (α: search direction; S: stepsize)
    - Move_value= $f(\mathbf{x}') - f(\mathbf{x})$
    - If move_value ≤ 0 (downhill move), set $\mathbf{x} = \mathbf{x}'$
    - If move_value > 0 (uphill move), set $\mathbf{x} = \mathbf{x}'$ with probability $p = exp(-$ move_value / T) > r (uniformly distributed random number in [0, 1]) (Kirkpatrick *et al.*, 1983)
    - End inner loop
- Reduce temperature according to an annealing schedule. An example is new $T=c \cdot T_0$, where $0 < c < 1$ (c is the rate of annealing schedule).
- End temperature loop

Basically, the algorithm is more supportive of a solution that improves the objective function, while permitting adverse solutions to give potential for the algorithm to discover the global optimum and to escape a local optimum (Venkataraman, 2002). Some of the critical parameters are the perturbation method of neighbor selection, transitional probability,[14] and the rate of annealing schedule (c).

To obtain a neighbor state of X_0, a search direction and a stepsize must be determined. Given a neighborhood structure, simulated annealing can be viewed as an algorithm that continuously attempts to transform the current configuration into one of its

---

[14] Probability to accept a worse solution.

neighbors. Practically, the calculation of the search direction and stepsize could be determined by using the traditional one-dimensional case (Venkataraman, 2002).

For the transitional probability function, Metropolis algorithm[15] and Glauber algorithm[16] are related to Boltzman probability distribution[17] (Edgar *et al.*, 2001).

The way in which the temperature is decreased is known as the *cooling schedule.* The rate of annealing schedule must be appropriate for its application so as not to get trapped in a local minimum due to fast cooling. The annealing rate can be fixed to a value or can vary in each step of cooling (Laarhoven and Aarts, 1987). For the algorithm to be effective, it is recommended that the probability be in the range of $0.5 \leq p \leq 0.9$ (Venkataraman, 2002).

This mechanism is mathematically best described by means of a Markov chain[18], a sequence of trial, where the outcome of each trial depends only on the outcome of the previous one (Laarhoven and Aarts, 1987). An example of a Markov chain is a random walk[19].

A Simulated Annealing program consists of a pair of nested DO-loops. The outer-most loop sets the temperature, and the inner-most loop runs a Metropolis Monte Carlo simulation at that temperature.

---

[15] $p = exp$ (-move_value / T)

[16] $p = exp$ (-move_value / T) / (1 + exp (-move_value / T)

[17] $p = -k$ / T (k: Boltzmann constant; T: annealing temperature)

[18] A collection of random variables $\{X_t\}$ (where the index t runs through 0, 1,...) having the property that, given the present, the future is conditionally independent of the past. In other words, $P(X_t = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_{t-1} = i_{t-1}) = P(X_t = j \mid X_{t-1} = i_{t-1})$. (Eric W. Weisstein. "Markov Chain." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/MarkovChain.html, May 18, 2005)

[19] Idea of taking successive steps in a random direction.

Like other discrete optimization techniques, such as Branch-and-Bound and Tabu Search, SA is a popular method to solve combinatorial optimization problems, which seek the best combination out of many possible combinations. Typically, these techniques are being used in production system planning, scheduling, transportation, and logistics.

3.2.3 Integrated Controlled Random Search for Dynamic Systems (ICRS/DS)

ICRS/DS is a modification of the ICRS algorithm (Banga and Casares Long, 1987; Casares and Rodriguez, 1989), which was a generalization of the method proposed by Goulcher and Casares Long (1987) for steady-state optimization problems (Banga *et al.*, 1997). As a generic algorithm, ICRS/DS is not as popular as GA or SA, but it has been applied to various problems, such as bioprocesses (Banga *et al.*, 1997), wastewater treatment (Banga and Casares Long, 1987), retorting (Banga *et al.*, 1991), drying (Banga and Singh, 1994), and meat patty cooking (Banga *et al.*, 2001), and proved its ability to solve dynamic optimization problems.

Basically, ICRS/DS uses two strategies: control vector parameterization and a stochastic direct search procedure. The original constrained dynamic optimization problem is transformed into a constrained nonlinear programming (NLP) problem by using a flexible parameterization of the control function. The constrained NLP problem is solved using the stochastic direct search procedure (Banga *et al.*, 1997). Like GA and SA, the search uses search direction and step size to determine the next feasible move. This search mechanism starts with a user-specified feasible control vector and perturbs the vector randomly with a normal probability distribution, which has the control vector as the average and a standard deviation vector. The standard deviation vector is a set of

smaller distances between the control values and their upper and lower bounds. If the algorithm cannot find any improvement within a limited number of trials, the standard deviation vector is reduced by a heuristic parameter, which results in smaller search step. The algorithm can be terminated by checking user-specified tolerances for the decision vector and/or the performance index at the end of each iteration (Banga *et al.*, 1997).

The detailed procedures of ICRS/DS can be found in the work of Banga *et al.* (1997). These procedures consist of control profile parameterization and the modified ICRS algorithm.

Control profile parameterization:

The control function $\mathbf{u}(t)$ over $t \in [t_0, t_f]$ is parameterized using N points, (θ_i, ω_i) $(i = 1...N)$. The value of $\mathbf{u}(t)$ at iteration k can be calculated using variable-length piecewise linear interpolation (Equation [3.8]) within the optimization procedure and $\theta_i^k \le t \le \theta_{i+1}^k$;

$$u^k(t) = \frac{\omega_{i+1}^k - \omega_i^k}{\theta_{i+1}^k - \theta_i^k}(t - \theta_i^k) + \omega_i^k \qquad \textbf{[3.8]}$$

By using the above parameterization technique, the original optimal control problem is transformed into a $2N$ (or $2NM$ if there are M control variables) dimensional nonlinear programming (NLP) problem. For implementation purpose, the parameterized time and control values (θ_i, ω_i) are replaced with the decision variable vector ξ for any iteration k by using the following rule:

$$\begin{aligned} \xi_i^k &= \theta_i^k, \quad i = 1....N \\ \xi_i^k &= \omega_{i-N}^k, \quad i = N+1...2N \end{aligned} \qquad \textbf{[3.9]}$$

Also, the upper and lower bounds for the decision variables are expressed in a simple way:

$$\begin{aligned} \xi_i^U &= \theta_i^U, \quad \xi_i^L = \theta_i^L; \qquad i = 1...N \\ \xi_i^U &= \theta_{i-N}^U, \quad \xi_i^L = \theta_{i-N}^L; \qquad i = N+1...2N \end{aligned} \qquad \textbf{[3.10]}$$

In addition, the parameterization must satisfy following conditions.

For time-intervals:

$$\left.\begin{aligned} &\theta_i^L \le \theta_i^k \le \theta_i^U \\ &\theta_i^k \le \theta_{i+1}^k \\ &\theta_i^L \le \theta_i^U \le \theta_{i+1}^L \end{aligned}\right\} i = 1...N, \quad \forall k \qquad \textbf{[3.11]}$$

For control vector limits:

$$\omega_i^L \le \omega_i^k \le \omega_i^U, \quad i = 1...N, \quad \forall k \qquad \textbf{[3.12]}$$

For initial and final time-interval limit:

$$\theta_1^L = \theta_1^U = t_0, \ \theta_N^U = t_f \qquad \textbf{[3.13]}$$

The above parameterization method is designed for variable-length intervals. However, if a fixed constant discretization ratio of time is desired, it suffices to take

$$\theta_i^U = \theta_i^L + \varepsilon = \left(\frac{t_f}{N-1}\right)(i-1) + \varepsilon \qquad \textbf{[3.14]}$$

where ε is a small number.

The Modified ICRS Algorithm (Figure 3.7):

The algorithm starts with a feasible decision vector, which can be chosen randomly. Subsequently, a standard deviation vector is calculated by multiplying minimum intervals (smaller interval between the decision vector and upper and lower bound vector) by a heuristic parameter (k_1). Based on the initial feasible vector, the algorithm generates a next decision vector by using the standard deviation vector (stepsize) and Gaussian distribution (random direction). Once the next decision vector is evaluated as feasible (satisfying all the constraints) and improved (increasing or decreasing objective function value), the above iteration is repeated until a convergence criteria is satisfied. However, if the next generation decision vector is determined as an infeasible or retrogressive (or remains the same), the failure counter (F) is increased by one up to a pre-set value (recommending $n_e \times$ total dimension of a problem). Every failure counter increment, another new decision vector is generated with the same standard deviation vector and tested. If the algorithm fails to get an improved decision vector within a pre-set failure counter value, the standard deviation vector (stepsize) is decreased by multiplying another heuristic parameter (k_2). This stepsize reduction is continued until the algorithm encounters a feasible and improved decision vector with the frequency of the maximum failure counter value. Once the feasible and improved decision vector is found, the failure counter is set to zero.

In this algorithm, the three heuristic parameters (k_1, k_2, and n_e) are critical to be successful in optimization process. Pan (1998) recommended 1/3, 1/2, and 4 as default value for k_1, k_2, and n_e, respectively.

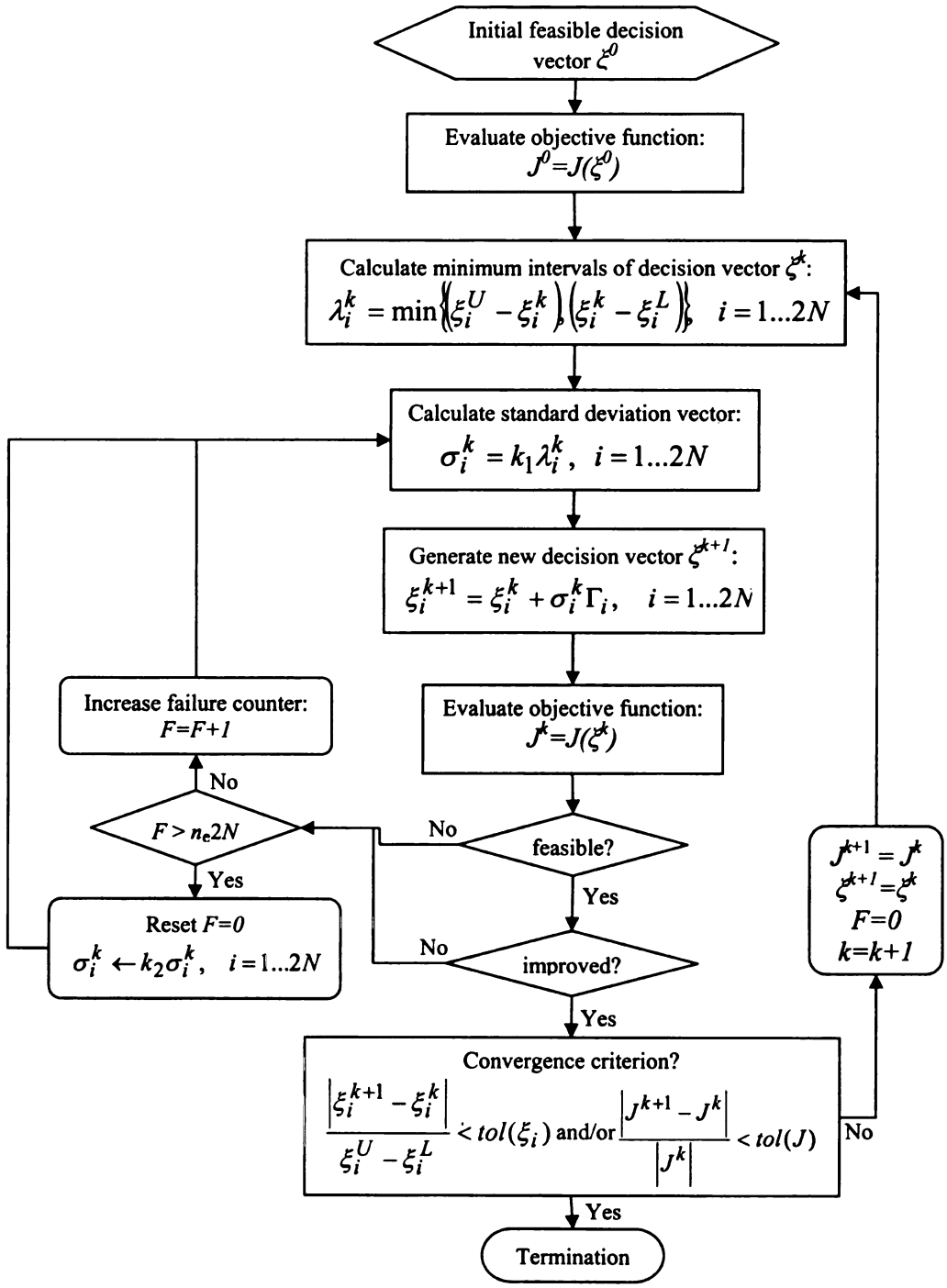


Figure 3.7 Flowchart of modified ICRS algorithm.

3.3 Artificial Neural Network

3.3.1 Fundamentals

As mentioned before, availability of a model is essential to solving optimization problems in various applications. Even when a model is available, the computational demand and the compatibility with optimization algorithms are critical to run an optimization procedure effectively. Taking into account the above facts, simpler and faster models often must be considered. Therefore, artificial neural network (ANN) appears to be a good candidate for fast nonlinear dynamic models (Trelea *et al.*, 1997).

In cognitive neuroscience, a neural network (also known as a neuronal network or biological neural network to distinguish from artificial neural networks) is a population of interconnected neurons. As the primary cells in the nervous system, neurons are structural constituents of the brain. Generally, neurons are five to six orders of magnitude slower than a silicon chip; events in a silicon chip happen in the nanosecond (10^{-9} s) range, whereas neural events happen in the millisecond (10^{-3} s) range. However, the brain processes tremendous amounts of information by using massive interconnections between approximately 10 billion neurons and 60 trillion synapses (Haykin, 1999). Neurons react to electrochemical impulses. Generally, neurons consists of several dendrites (receptive zone) and one axon (transmission line to output). If the sum of the input signals from dendrites surpasses a certain threshhold, the neurons encodes their outputs (action potentials or spikes), which originate at the cell body (soma) of neurons and transmit the electrical signal along the axon (Figure 3.8).

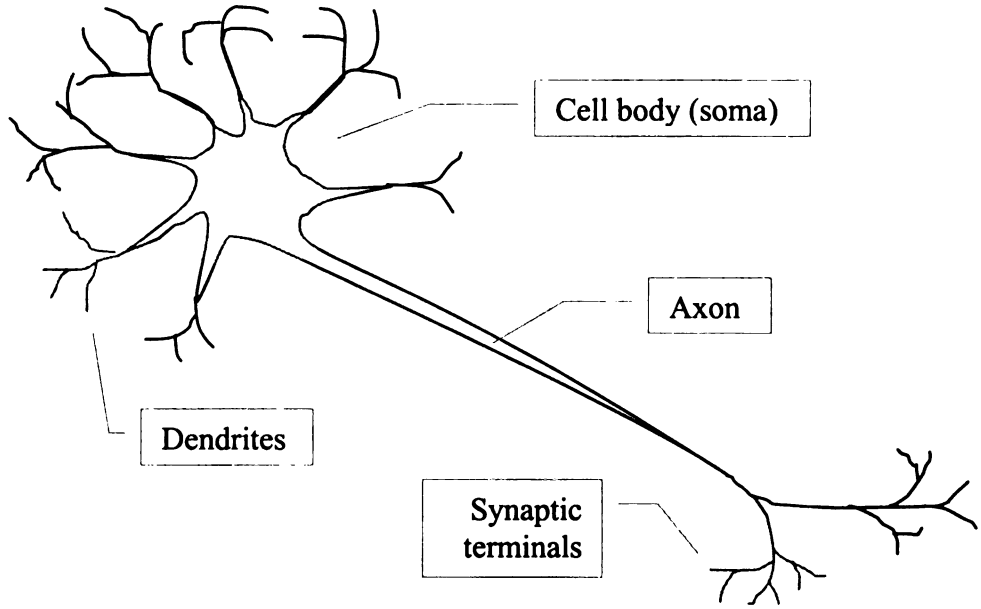


Figure 3.8 Physical structure of a neuron.

An artificial neural network is defined as "a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use" (Haykin, 1999). The mathematical model of a neuron is comprised of three basic elements that are a set of synapses[20] (connecting links), an adder (summing junction), and an activation function (squashing function) in the Figure 3.9.

[20] Elementary structural and functional units that mediate the interactions between neurons (Simon Hayakin, *Neural Networks: A Comprehensive Foundation*, p. 6).

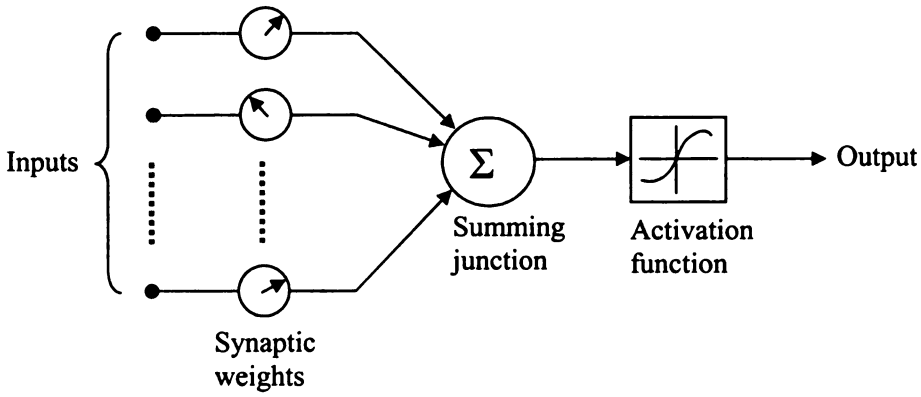


Figure 3.9 Mathematical model of biological neuron.

An input signal is multiplied by the synaptic weight while passing through the synapse. All the weighted input signals are summed up at the adder. Finally, the activation function regulates the amplitude of the output of a neuron. Basically, training a neural network or learning process is the process of adjusting the synaptic weights to minimize the error between outputs and targets. Depending on the methods of learning and the architectures of network, a large variety of neural networks exists.

3.3.2 Back-Propagation Feed-Forward Neural Network (FFNN)

The feed-forward neural network trained by back-propagation is reckoned as a major advance in the history of artificial neural network, because it provides a theoretically sound technique for training multilayer, feed-forward networks with nonlinear neurons when there is no efficient, theoretically sound method for training

(Wasserman, 1993). As a training method, back-propagation became highly popular in neural network training (Haykin, 1999).

In the feed-forward neural network, the input data flows only in one direction from input layer to output layer. Basically, training a neural network means to find the best set of weights of neurons that minimizes the error between neural network prediction and measured data. Thus, the back-propagation method starts with initial weights to produce prediction and uses its error to adjust the weights in the direction of reducing the error by using a learning rule or weight updating rule. Because adjusting weights starts from outmost layer to inner hidden layer, the method goes by the name of "back-propagation." The following is the fundamental principles of the back-propagation training method. The notations of Figure 3.10 will be used for further mathematical representation. Details can be found in the work of Wasserman (1993).

The first step for back-propagation is to calculate outputs with given inputs and initial weights of each layer. The input column vector (**x**) is multiplied by hidden layer weight matrix ($\mathbf{W^{HL}}$), then the result is transferred to the nonlinear activation function (f). Thus,

$$y_j = f\left(w_{ji}^{HL} x_i\right) \qquad \textbf{[3.15]}$$

Finally, at the output layer, the vector y is multiplied by output layer weight matrix ($\mathbf{W^{OL}}$). Therefore,

$$y_k = w_{kj}^{OL} y_j = w_{kj}^{OL} f\left(w_{ji}^{HL} x_i\right) \qquad \textbf{[3.16]}$$

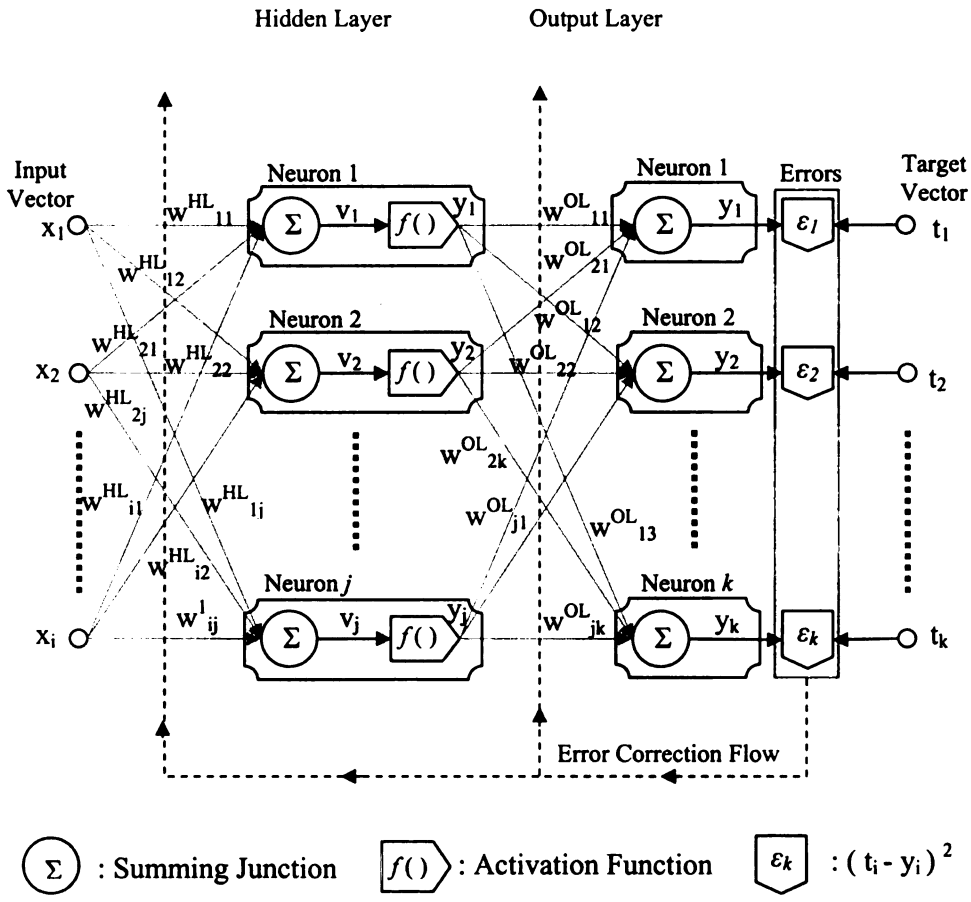


Figure 3.10 Topology of back-propagation of feed-forward neural network.

After obtaining the network output, error between output vector and target vector is computed. Sum of squared error (SSE) is used for error measurement. Hence,

$$\varepsilon = SSE = \sum_k \varepsilon_k = \sum_k \left[(t_k - y_k)^2 \right] \qquad [3.17]$$

Now, the error information is propagated from the output layer to the hidden layer in the form of a gradient vector (∇), which is the set of derivatives for all weights with

respect to the output error. Thus, a positive gradient of a weight means that overall error is increased, which requires the weight to be decreased to reduce the overall error. In case of a negative gradient, the opposite is true. The gradient vector of output layer can be evaluated as follows:

$$\nabla_{jk}^{OL} = \left(\frac{\partial \varepsilon}{\partial w_{jk}^{OL}} \right) = \left(\frac{\partial \varepsilon}{\partial y_k} \right) \left(\frac{\partial y_k}{\partial w_{jk}^{OL}} \right) \quad \textbf{[3.18]}$$

where,

$$\frac{\partial \varepsilon}{\partial y_k} = -2(t_k - y_k) \quad \textbf{[3.19]}$$

$$\frac{\partial y_k}{\partial w_{jk}^{OL}} = y_j \quad \textbf{[3.20]}$$

∇_{jk}^{OL} : the gradient vector component associated with the weight from neuron j in the hidden layer to neuron k in the output layer

w_{jk}^{OL} : the weight connecting neuron j in the hidden layer to neuron k in the output layer

y_j : output of neuron j in the hidden layer

y_k : output of neuron k in the output layer

t_k : the target value for neuron k in the output layer

By using the calculated gradient of each weight, the following weight update rule or learning rule can be defined.

$$w_{jk}^{OL}(n+1) = w_{jk}^{OL}(n) - \eta \cdot \nabla_{jk}^{OL} \quad \textbf{[3.21]}$$

where

$w_{jk}(n)$: the value of the weight at time n

η: a learning rate constant, typically < 1.0

According to the weight update rule, a weight of a negative gradient gains more weight and a weight of a positive gradient loses its weight.

The next step is to find the gradient vector of the hidden layer. According to the chain rule of calculus, this gradient can be expressed as:

$$\nabla_{ij}^{HL} = \frac{\partial \varepsilon}{\partial w_{ij}^{HL}} = \sum_{k} \left(\frac{\partial \varepsilon}{\partial y_k} \right) \left(\frac{\partial y_k}{\partial y_j} \right) \left(\frac{\partial y_j}{\partial v_j} \right) \left(\frac{\partial v_j}{\partial w_{ij}^{HL}} \right) \quad \textbf{[3.22]}$$

By substituting the following relations to the Equation 3.27

$$\frac{\partial \varepsilon}{\partial y_k} = -2(t_k - y_k) \equiv \delta_k \quad \textbf{[3.23]}$$

$$\frac{\partial y_k}{\partial y_j} = w_{jk}^{HL} \quad \textbf{[3.24]}$$

$$\frac{\partial y_j}{\partial v_j} = f'(v_j) \quad \textbf{[3.25]}$$

$$\frac{\partial v_j}{\partial w_{ij}^{HL}} = x_i \quad \textbf{[3.26]}$$

and defining

$$\delta h_j = \sum_k \delta_k w_{jk}^{HL} f'(v_j) \qquad [3.27]$$

where f' is the derivative of the nonlinear function. Thus, Equation 3.22 becomes

$$\nabla_{ij}^{HL} = \delta h_j \cdot x_i \qquad [3.28]$$

Now, the weights of the hidden layer can be updated by using the same updating rule of the output layer.

$$w_{ij}^{HL}(n+1) = w_{ij}^{OL}(n) - \eta \cdot \nabla_{ij}^{HL} \qquad [3.29]$$

The above two-layer example can be generalized for multi-layered feed-forward neural network. By calculating δ for each neuron using δ from the previous layer, ∇ for each weight can be calculated, and the weight can be adjusted.

The training can be done by using sequential method and batch method. Sequential method updates weight at the time each input vector is applied. In batch mode, however, changes of weights are done after presenting all input vectors to the network. Because the batch training averages the derivatives over a pass through the training set, the batch training gives a more accurate estimate of the overall gradient.

The back-propagation method is generally slow to train a network. However, some methods are available to speed the process. These methods are using second derivative information for fast convergence. Due to the second-order information, training time may be reduced by up to a factor of 100. Conjugate gradient descent and quasi-Newton method are examples of these fast convergence methods (Wasserman, 1993).

3.3.3 Generalized Regression Neural Network (GRNN)

GRNN is a neural network architecture that has many similarities with the radial basis-function[21] neural network (RBF), in which each hidden neuron uses a radial basis-function as the activation function, and the output neurons implement linear combinations of these radial basis-function. Among many neural networks, nonlinear regression theory based GRNN can approximate any arbitrary function between input and output vectors (Wasserman, 1993). Therefore, if the GRNN is used for predicting the future value of observed variables that are dependent variables related to input variables in a process, plan, or system, the GRNN can be used to model the process, plant, or system (Christodoulou and Georgiopoulos, 2001).

GRNN is based upon well-established nonlinear regression theory, which is the following formula (Wasserman, 1993):

$$E[y|\mathbf{x}] = \frac{\int_{-\infty}^{\infty} y \cdot f(\mathbf{x}, y)\, dy}{\int_{-\infty}^{\infty} f(\mathbf{x}, y)\, dy} \qquad \textbf{[3.30]}$$

where

y = output of the estimator

$\mathbf{x}$ = the estimator input vector

$E(y|\mathbf{x})$ = the expected value of out, given the input vector $\mathbf{x}$

$f(\mathbf{x}, y)$ = the joint probability density function (pdf) of $\mathbf{x}$ and y

---

[21] An activation function which is centered at a point specified by the connection weight vector and whose position and width are adjusted by learning. The most popular radial basis-function is the Gaussian function.

GRNN is, in essence, a method for estimating $f(\mathbf{x}, y)$, given only a training set. Detail architecture is shown in Figure 3.11. Specht (1991) shows that y_i (function value) is estimated optimally as follows:

$$y_i = \sum_{i=1}^{n} h_i w_{ij} \Big/ \sum_{i=1}^{n} h_i \qquad \textbf{[3.31]}$$

where

w_{ij} = the target (desired) output corresponding to input training vector x_i and output j

$h_i = \exp\left[\frac{-D_i^2}{2\sigma^2}\right]$, the output of a hidden layer neuron

$D_i^2 = (\mathbf{x} - \mathbf{u}_i)^T (\mathbf{x} - \mathbf{u}_i)$ (the squared distance between the input vector $\mathbf{x}$ and the training vector $\mathbf{u}$

$\mathbf{x}$ = the input vector (a column vector)

$\mathbf{u}_i$ = training vector i, the center of neuron i (a column vector)

σ = a constant controlling the size of the receptive region

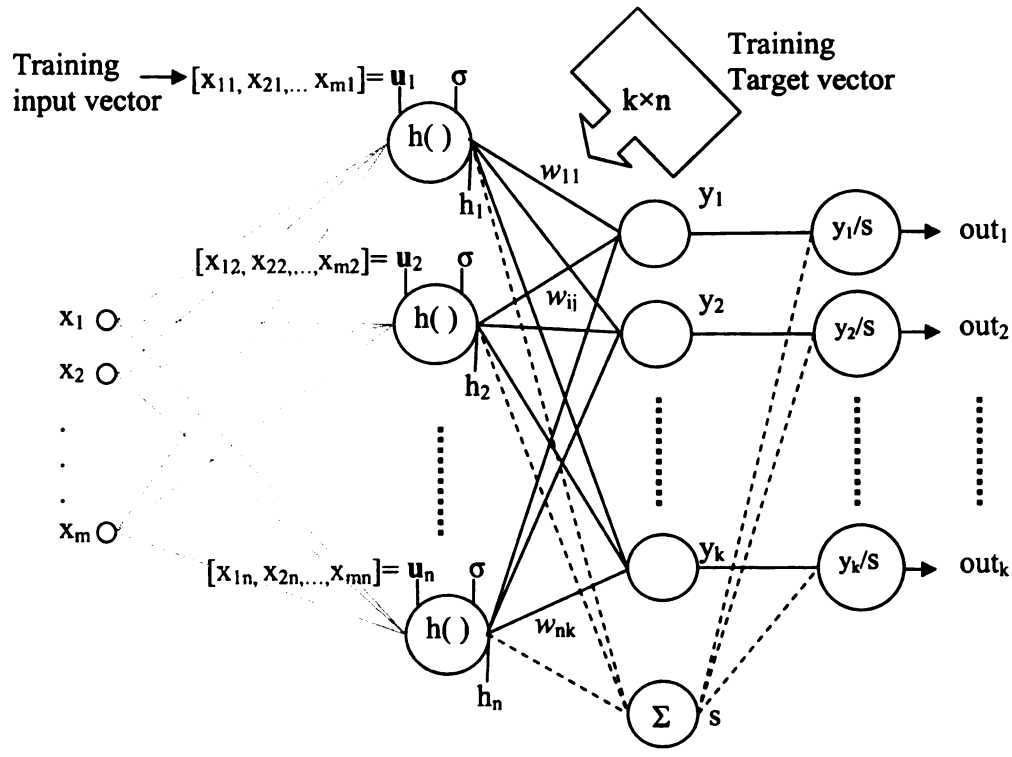


Figure 3.11 Architecture of generalized regression neural network (GRNN).

The GRNN copies the training cases into the network to be used to estimate the response on new points. In GRNN, one simply assigns to w_{ij} the target value directly from the training set associated with input training vector i and component j of its corresponding output vector (Wasserman, 1993). Because of the above fact, a GRNN trains almost instantly. Thus, as the number of pairs of inputs/output increases, the more computation time is required, because of the corresponding increase of the number of hidden neurons. The σ is the standard deviation of the response curve of the neuron. Therefore, points nearby contribute most heavily to the estimate. If σ is small, the neuron responds only to the input vector close to the weight of the neuron (Figure 3.12). Again,

if σ is large, the neuron responds to the wide range of input vector. GRNN can only be used for regression problems, because of the nature of the fundamental equation. Like a radial basis-function (RBF) network, a GRNN does not extrapolate. Thus, it is important to construct the training data set carefully, so that the data set covers the region of interest.

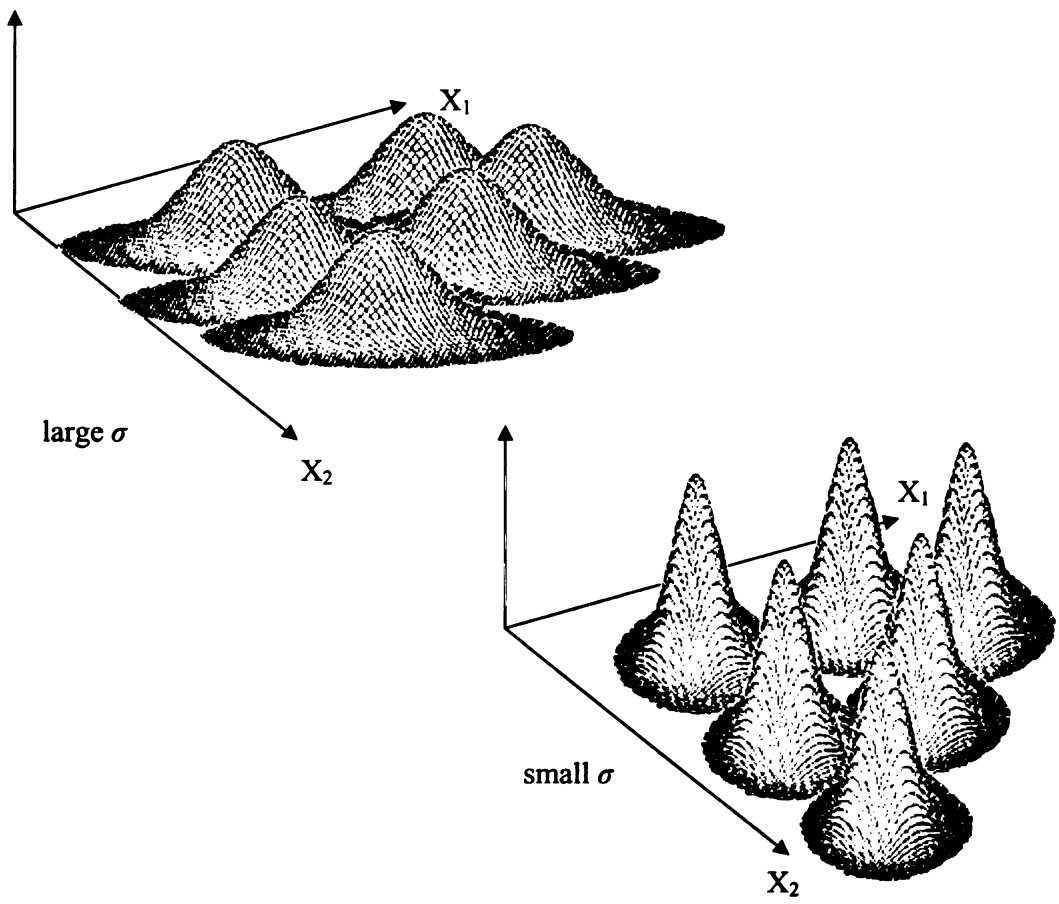


Figure 3.12 GRNN response depending on the size of receptive field (σ) in two-dimensional case.

4 METHODS AND PROCEDURES

4.1 Overall Methods and Procedures

Figure 4.1 is a schematic diagram to show the overall procedures and methods of this project. In this study, the model outputs from an existing, validated finite element model (FEM) of meat patty cooking (Watkins, 2004) were considered as actual field experiments. For this study, the model was expanded to include quality prediction capability, such as internal color change (Section 4.3.3) and surface color change (Section 4.3.4). By using the integrated FEM, various data groups (train and test or validation) were generated to develop various artificial neural networks (ANN) as alternatives to the FEM (Section 4.5). Then, various optimization strategies were applied to both models (FEM and ANN's) to find a theoretically possible optimal condition and to evaluate the various strategies (Section 4.6). Optimization strategies were then applied to industrially-relevant case studies (single-stage, double-stage, multi-zone, and an economic-based problem) to illustrate potential utility of these technologies (Section 4.7).

4.2 Impingement Cooking Technology

As the term "impingement" represents, it is a sharp collision produced by striking or dashing a heating medium against a product. Because of the "sweeping away" effect around the product, the boundary layer thickness is reduced greatly, which increases the heat transfer rate. Therefore, impingement technology can be characterized as a fast and efficient means of processing.

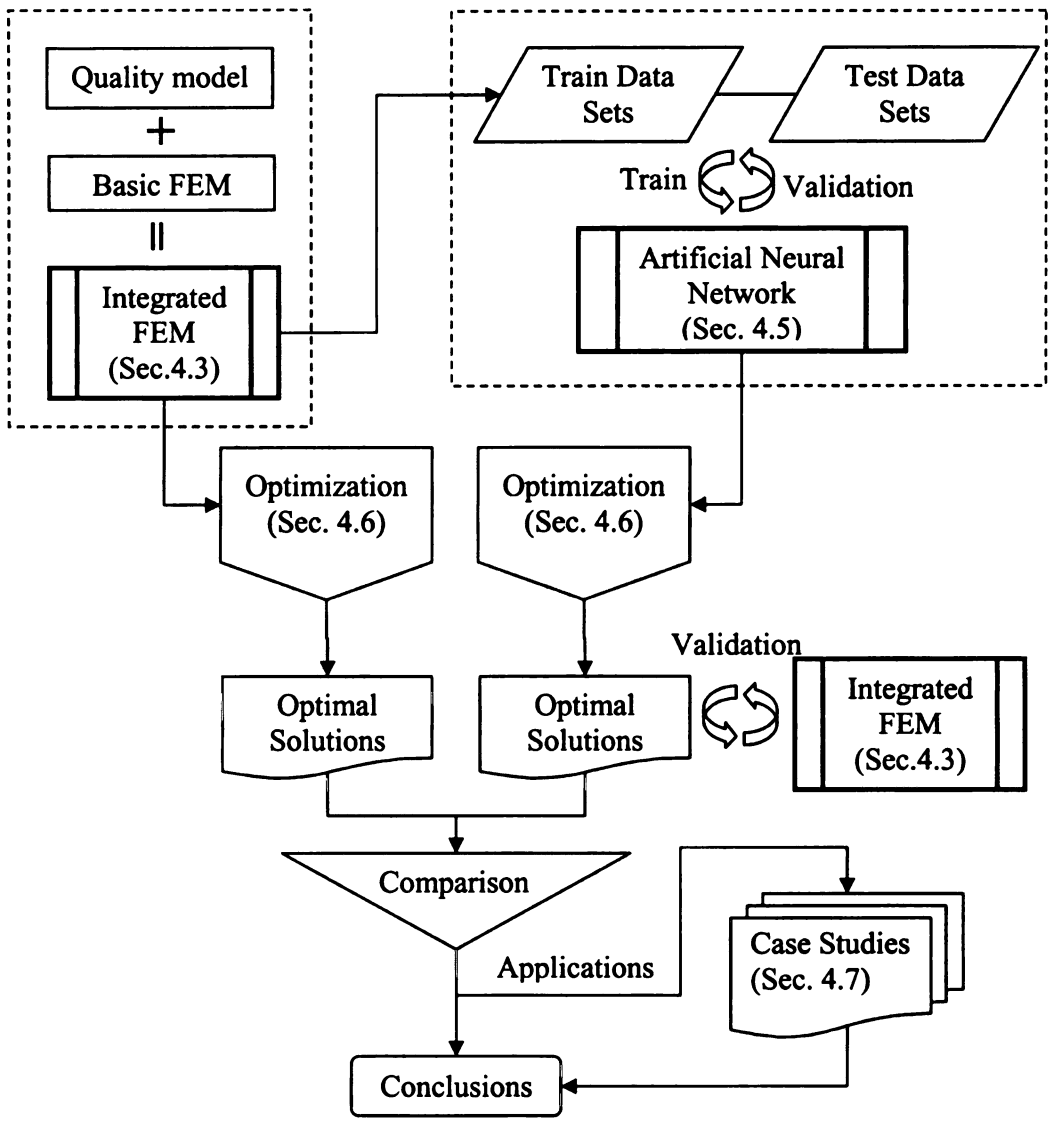


Figure 4.1 A schematic diagram of overall procedures and methods.

This technology is widely adopted in the industrial applications, including: annealing of non-ferrous sheet metals, tempering of glass, drying of paper and textile, cooling of electrical components and turbine blades, and processing of food products (Saad *et al.*, 1980). Especially, if the impingement technology is used with moist air instead of dry air, some unique advantages can be obtained

When the moist air meets a surface cooler than the dew point temperature of the moist air, water condenses on the product surface. Due to condensation at the initial stage of cooking, the condensed steam gives off its latent energy to the product. This efficient heat transfer happens until the product surface temperature increases beyond the dew point temperature of the moist air.

In this research, a moist air impingement oven (model JSO-IV, Figure 4.2), which is a commercial product of Stein-DSI (a business of FMC FoodTech, Sandusky, OH), was the example oven used for process modeling. The major process variables of the oven are impingement exit velocity, process duration, moisture content of the impinging gas (volumetric basis), and impinging gas temperature. These four process variables were used for the process modeling and optimization. Even though there are some physical system configuration parameters, such as open slot ratio, jet spacing, and gap between jet exit and surface, these parameters were held constant for this study and not included in the control variable group.

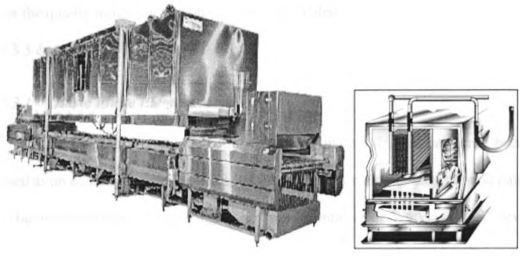


Figure 4.2 An actual and a cross-sectional image of the JSO-IV Jet Stream® Oven of Stein-DSI (FMC FoodTech).

4.3 Integrated Process Model Development

As a complex, multi-physical phenomenon, meat cooking involves many factors, such as yield, microbial inactivation, and quality changes. A finite element model developed by Watkins (2004) could predict yield (via fat and moisture prediction) and microbial inactivation within a meat patty subjected to moist air impingement cooking. However, the model could not predict other quality factors.

Internal cooked appearance of ground beef patties is used to evaluate doneness by many consumers (Hunt *et al.*, 1999). In addition, flavor is a very important component, which is affected by the Maillard reaction on the surface of meat product (Mottram, 1998). Therefore, internal color change and flavor (with surface color change) models were incorporated into the finite element model (Section 4.3.3 and 4.3.4).
Many of the reactions in food processing follow first-order kinetics, which can be expressed by the familiar *D*-value and *z*-value (Toledo, 1991). Those kinetic parameters

for the quality indices in this project were calculated based on literature data (Section 4.3.3 & 4.3.4).

4.3.1 Basic Finite Element Model

The process model is a central component in optimization problems, because it is used as an equality constraint. Thus, moist air impingement cooking of meat patties needs to be modeled before the cooking system can be optimized. Watkins (2004) developed this model by using a coupled heat and mass (moisture) transfer model. The model also encompassed fat transport and *Salmonella* inactivation. The model utilized the finite element method to solve separate equations for heat, moisture, and fat transport. These equations were coupled through boundary conditions and interdependent thermo-physical property relationships. An enthalpy formulation for heat transfer was utilized to avoid discontinuities related to solid-to-liquid phase changes of water and fat within the product. A solution for modeling condensing-convective boundary conditions during moist air impingement cooking, developed by Millsap (2002), was incorporated into the model. These boundary conditions accounted for the additional heating effects of surface condensation that can occur within moist air impingement systems (Watkins, 2004).

A ground and formed meat patty (2-D cylindrical object) was used as a model product. Input temperatures were converted to enthalpy using an equation based on the work of Voller and Cross (1981). Heat transfer was modeled via following equation.

$$\frac{\partial h}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r \cdot \frac{k_T}{c_T \cdot \rho}\frac{\partial h}{\partial r}\right) + \frac{\partial}{\partial z}\left(\frac{k_T}{c_T \cdot \rho}\frac{\partial h}{\partial z}\right) \qquad \textbf{[4.1]}$$

Moisture transfer within the product was modeled using a two-dimensional equation for diffusion in radial coordinates.

$$\frac{\partial m_{water}}{\partial t} = \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{k_{m,water}}{c_{m,water} \cdot \rho} \frac{\partial m_{water}}{\partial r} \right) + \frac{\partial}{\partial z} \left(\frac{k_{m,water}}{c_{m,water} \cdot \rho} \frac{\partial m_{water}}{\partial z} \right) \right) \quad [4.2]$$

For fat transfer modeling, a two-dimensional formulation of Darcy's law for diffusion of liquids through porous media was used (Datta, 2002).

$$\frac{\partial m_{fat}}{\partial t} = \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \cdot D_{cap,fat} \frac{\partial m_{fat}}{\partial r} \right) + \frac{\partial}{\partial z} \left(D_{cap,fat} \frac{\partial m_{fat}}{\partial z} \right) \right) \quad [4.3]$$

The heat transfer boundary condition was formulated with a convection term and a moisture transport (evaporation/condensation) term:

$$k_T \frac{\partial T}{\partial n} = h_T (T_{air} - T_{surface}) + h_{m,water} \cdot \lambda_{vaporization} \cdot (C_{air} - C_{surface}) \quad [4.4]$$

Mass transfer at the product surface was modeled by using a convective boundary condition.

$$k_{m,water} \frac{\partial m_{water}}{\partial n} = h_{m,water} (C_{air} - C_{surface}) \quad [4.5]$$

The fat content at the surface of the patty was modeled using an equation derived from experimental data (Equation [4.6]) (Watkins, 2004). The equation [4.6] was utilized to set the values of the fat content at each boundary node as a function of temperature and product composition.

$$m_{fat} = 0.7062 - 0.0193 \cdot T + 0.0001 \cdot T^2 + 0.0069 \cdot m_{i,fat} + 0.0002 \cdot T \cdot m_{i,fat} \quad [4.6]$$

In addition to these boundary conditions, heat and mass transfer at the radial and vertical centerlines of the patty were assumed to be zero due to product symmetry. These coupled partial differential and ordinary differential equations were solved by using the finite element method (Watkins, 2004). The finite element solution of this time-dependent field problem was solved with a finite difference approximation in the time domain to generate the time-step solution, and the time step was 1 second. The time step was smaller than the allowable time step (Δt=2.16 s), which was calculated to prevent the finite element model from oscillating and deviating from physical limits for a right triangle element, by using Equation [4.7] (Segerlind, 1984).

$$\Delta t < \frac{\alpha}{1-\theta} \qquad \textbf{[4.7]}$$

where

α: $\det([\mathbf{c}^{(e)}]-\alpha[\mathbf{k}^{(e)}])=0$; det: determinant; **c** & **k**: element matrices

θ=1/2 for central difference method

However, in some cases of actually running this FEM model, instability of the prediction of yield and surface temperature were observed at the very early stage and rarely at the later stage. Therefore, the time step was decreased from 1 to 0.5 second to reduce the instability of the model.

Watkins (2004) validated the finite element model using data generated in industrial cooking tests with beef patties in a JSO-IV (Stein-DSI, a business of FMC FoodTech, Sandusky, OH) and additional published data for ground chicken breast patties cooked in a pilot-scale impingement oven (Stein, FMC FoodTech). Predictions of

cooking yield had errors ranging from 0.1 to 15.4%, with an average deviation of 5.9% (27 sets of process conditions) in the industrial cooking tests. Comparisons with published data were also favorable (standard errors of prediction for yield ranged from 1.1 to 15.7%).

4.3.1.1 Model Product

The dimension of the patty was 120 mm in diameter and 10 mm in thickness, which was modeled as a two-dimensional, axisymmetric body. Figure 4.3 shows the element mesh configuration, which consisted of 36 nodes and 50 elements.

The model patty was assumed to initially be 60% water, 20% fat, and 20% protein. The initial temperature was set to 5 °C.

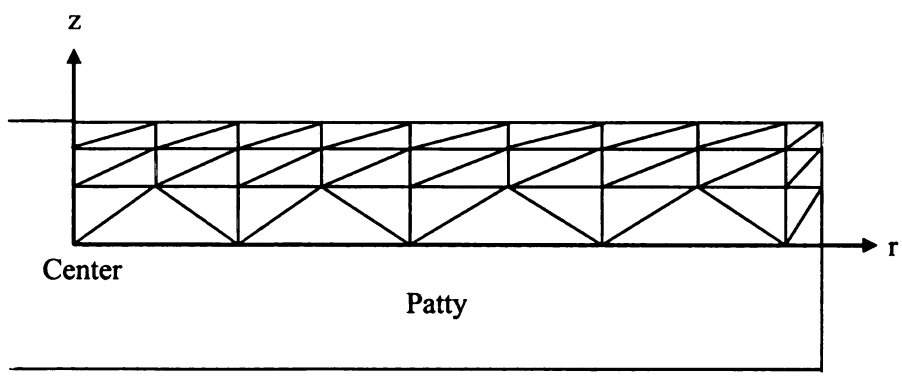


Figure 4.3 Finite element mesh utilized for one quarter of the 2-D cylindrical model product.

4.3.2 Kinetic Parameters for Microbial Inactivation in a Meat Patty

Salmonella is the most heat-resistant organism among other food related microorganisms, such as *E. coli* O157:H7 and *Listeria monocytogenes* (Murphy *et al.*, 2004). A few works (Juneja *et al.*, 2001; Murphy *et al.*, 2002; Murphy *et al.*, 2004) of thermal inactivation kinetics for *Salmonella* cocktails in ground beef were found and compared to get a conservative reference D and z-value.

In Table 4.1, D values at 62.5 °C, which is the center between 55 and 70 °C, were compared, and the D value of 2.62 min was selected as a conservative value.

For z-value, the highest value (9.14 °C) was chosen among other values, because the value reflects the least sensitivity to temperature change.

Table 4.1 Comparison chart of thermal inactivation parameters of *Salmonella* cocktail for ground beef (1:(Murphy *et al.*, 2002); 2:(Juneja *et al.*, 2001); 3:(Murphy *et al.*, 2004)).

		1	2	3
Fat [%]		18.56	12.45	34.4
Water [%]		51.30	65.5	49.7
Method		0.7 mm thin metal container	1-2 mm thin bag	Thin bag
Culture Type		Cocktail	Cocktail	Cocktail
	T_r [°C]	[min]	[min]	[min]
D-value	55	9.09		37.05
	57.5	7.70		18.35
	58		8.65	
	60	4.80	5.48	6.90
	62.5	2.40	1.50	2.62
	63			
	65	0.97	0.67	1.03
	67.5	0.57		0.30
	68			
	70	0.25		0.066
z-value		9.14 [°C]	6.01 [°C]	5.74 [°C]

4.3.3 Kinetic Parameters for Internal Color Change of a Meat Patty

Kinetic modeling is a very efficient tool to model food quality, which otherwise can be very subjective. Once the rate and temperature dependence of a reaction is known, the reaction can be controlled and predicted (Martins *et al.*, 2001).

The color of the center region of a meat patty is an important quality index related to acceptability. As cooking proceeds, the center color gradually changes from red-pink to brown (*i.e.*, "well done") due to the heat denaturation of various types of myoglobin. The rate of cooked meat hemoprotein formation (via the rate of loss of myoglobin solubility) was found to obey first-order kinetics in aqueous muscle extracts and mixtures of myglobin and bovine serum albumin (Geileskey *et al.*, 1998). They measured first-order rate constants for the loss of myoglobin solubility in various muscles at 60, 65, 70, and 80 °C.

To implement first-order kinetics in the basic finite element model, a reference decimal reduction time (D_r) and *z*-value were calculated based on the rate constants at the various temperatures (Toledo, 1991). The reference temperature was 60 °C. D_r was calculated by using Equation [4.8] and the first-order rate constant (Geileskey *et al.*, 1998) at the reference temperature.

$$D_r = \frac{\ln(10)}{k} \qquad \textbf{[4.8]}$$

To calculate the z-value in the temperature ranging from 60 to 80 °C, activation energy (E_a) was obtained by using temperature and rate constant data from (Geileskey *et al.*, 1998), assuming an Arrhenius relationship (Toledo, 1991). The *z*-value was then computed as follows:

$$z = \frac{\ln(10)}{(E_a / R)} T_1 T_2 \qquad [4.9]$$

Table 4.2 shows the derived D_r and z-values for different muscles. Considering that most ground beef comes from the chuck, sirloin, or round, the D_r and z-value of beef chuck were used for estimating acceptability (via center color change) of model product. Even though the color for acceptability can be predicted with the above parameters, an adequate log reduction value of internal color change must be specified to use the quality as a constraint in this optimization problem.

Table 4.2 Reference decimal reduction time (D_r) and z-value of internal color change in the various muscles at the reference temperature of 60 °C (Geileskey *et al.*, 1998).

Muscle	D_r [min]	z [°C]
Beef shin	85.28	9.63
Beef chuck	65.79	9.79
Beef m. I. dorsi	35.42	10.41

Hunt *et al.* (1999) observed that 95.9% denaturation of oxymyoglobin (OMb), which is contained in significant amount in most ground beef, was visually scored 5 (no evidence of pink color or "well done"). Also, the ground beef used in that research contained 20% fat, and the pigment was predominantly OMb (Hunt *et al.*, 1999), which implies a good match with the model product in this study. Thus, the target level of denaturation (95.9%) was converted to the equivalent value of 1.387 log reductions, which was used as a minimum criterion for internal color change of meat patty. Even though there is some variation in the concentration of myoglobin in beef, the differences in concentration do not appear to be a factor in the dependence of denaturation on muscle type (Geileskey *et al.*, 1998).

4.3.4 Kinetic Parameters for Surface Color Change of a Meat Patty

Non-enzymatic browning in food processing is the major cause of surface browning, aroma, and taste (Martins, 2003). Non-enzymatic browning of baked products is a very complex chemical reaction, which encompasses two major reactions, the Maillard reaction and caramelization (Zanoni *et al.*, 1995).

The Maillard reaction starts with an initial reaction of a reducing sugar with an amino compound, followed by consecutive and parallel reactions to form a variety of colored and colorless products (Martins and Van Boekel, 2005). In addition to the desirable effects, the reaction generates undesirable results, such as discoloration, off-flavor, and mutagenic and carcinogenic components. Therefore, it is necessary to optimize the reaction by finding the best balance between the favorable and unfavorable effects of the reaction in a given process (Lingnert, 1990). The Maillard reaction is very complex reaction and very difficult to control (Martins *et al.*, 2001). For instance, over 1,000 volatile compounds are formed during cooking (Mottram, 1998). Many factors influence the reaction, such as temperature, time, pH, water activity, type of reactants, and availability of reactants (Lingnert, 1990).

The Maillard reaction kinetics has been studied in various applications, such as frying and drying. However, detailed observations for the Maillard reaction during cooking of meat patties, especially in the moist air impingement convection cooking environment, were not found in the literature.

Another non-enzymatic browning is caramelization. Caramelization is defined as the thermal degradation of sugars leading to the formation of volatiles (caramel aroma)

and brown-colored products (caramel colors). Caramelization usually occurs at high temperature (>120 °C), compared to the Maillard reaction temperature (>50 °C).

Non-enzymatic browning is very complex and encompasses not just a single reaction pathway, but a whole network of various chemical reactions (Martins *et al.*, 2001). Although complex reactions are involved, general perceptions for non-enzymatic browning reactions are color and aroma. The resultant effect of the various reactions can be lumped into crust color development during a meat cooking process. Dagerskog and Bengtsson (1974) studied the relationship among crust color formation, yield, composition, and processing conditions for double-sided pan frying of meat patties. They found kinetic parameters for crust color changes, in terms of reaction rate and activation energy. Ateba and Mittal (1994) obtained kinetic parameters for crust total color change and firmness. The above studies for crust color change of meat product used the total crust color change (ΔE) to describe the overall quality changes due to non-enzymatic browning. The total color change was calculated with the following equation (Dagerskog and Bengtsson, 1974):

$$\Delta E = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2} \qquad \textbf{[4.10]}$$

where,

ΔL: lightness

Δa: redness

Δb: yellowness

Color development is a surface phenomena, so that the plot of surface temperature (recipe A) and color change (recipe C) (Dagerskog and Bengtsson, 1974) were digitized

to obtain a D_r and z-value (Table 4.3). Even though the recipes were different, both data were assumed for one recipe in this study, because only two graphical data sets were available. Recipe "A" consisted of 59.6% water, 15.6% fat, and 9% breadcrumb, and recipe "C" consisted of 66.6% water, 14.1% fat, and no breadcrumb. Therefore, recipe "C" is closer to the model patty in this study, in terms of composition.

Table 4.3 Digitized data of surface time-temperature plot (recipe A) and time-color change plot (recipe C) of Dagerskog and Bengtsson (1974).

	Pan Temperatures							
	140 [°C]		160 [°C]		180 [°C]		200 [°C]	
Time [min]	T_s [°C]	ΔE	T_s [°C]	ΔE	T_s [°C]	ΔE	T_s [°C]	ΔE
0	10.00	0	10	0	10	0	10	0
1	101.40	5.51	107.60	3.95	121.60	6.14	130.30	7.70
1.5	103.40	3.48	110.20	6.76	123.50	7.80	132.60	9.99
2	103.70	6.02	111.20	9.46	122.20	9.88	129.40	12.17
2.5	104.70	8.62	112.50	9.04	122.90	12.27	130.70	14.82
3	105.60	8.57	115.10	12.21	126.40	15.03	135.90	15.60
4	109.50	12.36	122.20	15.17	134.90	16.37	148.90	18.40
5	113.40	14.90	127.70	17.51	142.70	17.87	158.60	20.37
6	116.70	16.04	132.00	19.01	149.80	19.48	166.10	20.21

T_s: product surface temperature

The ΔE was replaced with the difference with the maximum color change (ΔE_{max}=22). Then, a log reduction of color change was calculated. By using the Microsoft Excel SOLVER (Microsoft Excel Version 2000: Redmond, WA) and Equation [4.11], the D-value and z-values for this kinetics were estimated based on minimization of the RMSE of color change.

$$\log\left(\frac{C}{C_0}\right) = -\int_0^{t_f} \frac{1}{D} dt \qquad [4.11]$$

where,

$$D = D_r \cdot 10^{\frac{T_r - T}{z}} \qquad [4.12]$$

The estimated D-value and z-value for the surface color kinetics were 7.75 min and 90.55 °C, respectively (average RMSE=1.33 [ΔE]).

Now, dynamic surface color change can be predicted by integrating Equation [4.11] over time with the given D and z-values. However, there are several hurdles for the non-enzymatic browning to take place on the surface of a meat patty in the moist air cooking environment. The first barrier is water film formation due to condensation on the meat patty surface when the surface temperature is lower than the dew point temperature of the moist air. The water film keeps the surface temperature under 100 °C, which is far less than the critical caramelization temperature of 120 °C. For the Maillard reaction, some water is needed, because water participates in the later stage of the reaction. However, excessive water suppresses the reaction and also dilutes surface concentration of amino acids and sugars, which result in retarded or no reactions. Therefore, under the presence of condensed water on the surface, it is hard to expect non-enzymatic browning mainly due to the Maillard reaction. Even after overcoming the water film formation, evaporation occurs, which decreases the surface temperature until the film disappears. When the product surface temperature reaches 100 °C, the water evaporation zone recedes toward the center. The surface continues to lose water, the temperature rises, and

structural and chemical changes in the protein result in crust formation (Ateba and Mittal, 1994). Finally, non-enzymatic browning of the surface takes place rapidly when the surface temperature passes the critical temperature of the caramelization reaction. After this point, caramelization is accelerated significantly. The above limitation to non-enzymatic browning on the surface of meat patty under moist air cooking conditions need to be implemented in a simple non-enzymatic browning model to predict surface quality of the meat patty.

To account for the adverse effects, a function cueing the non-enzymatic browning on the product surface is necessary. A sigmoid function, which is the common form of an activation function, was adopted for this purpose via the logistic function (Equation [4.13]).

$$f = \frac{1}{1 + e^{-a \cdot (Ts - Tdew)}} \qquad \textbf{[4.13]}$$

The sigmoid function is ideal, because it is convenient to account for an abrupt change around a critical point and also it is continuous (Figure 4.4). The logistic function parameter a was determined at the value where 99.5% change is achieved in the range of +/- 5 °C around the critical value. Therefore,

$$a = -\frac{\ln(0.005)}{5} \approx 1.06 \qquad \textbf{[4.14]}$$

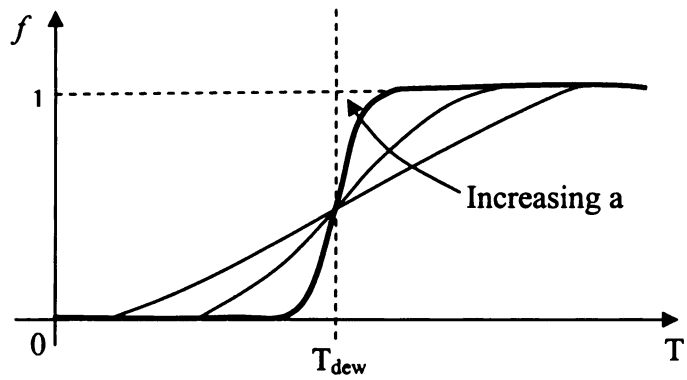


Figure 4.4 A sigmoid function (logistic function) of surface temperature (Ts) showing gradual change in a certain factor around a critical value, T_{dew} in this case.

The sigmoid function incorporating surface temperature and dew point temperature was inserted into Equation [4.11] to provide a status of surface browning for integration over time. Thus, the final equation of non-enzymatic browning, accounting for condensation effects, was expressed as follow:

$$\log\left(\frac{C}{C_0}\right) = -\int_{t_0}^{t_f} \frac{f}{D}\, dt \qquad \textbf{[4.15]}$$

where,

$$f(T_s) = \frac{1}{1+e^{-a(Ts-Tdew)}} \qquad \textbf{[4.16]}$$

$$D = D_r \cdot 10^{\frac{Tr-T}{z}} \qquad \textbf{[4.17]}$$

In Equation [4.14], the surface temperature was obtained from the finite element solution, and the dew point temperature was interpolated by using the tabulated data[22] of dew point temperature and moisture content by volume (which is a model input).

For non-enzymatic browning, both minimum and maximum critical values are necessary for the cooking process to control the desirable color development on the surface of the product, so that the product is neither too pale nor over-cooked. Generally, the surface non-enzymatic browning is measured by total color change (Dagerskog and Bengtsson, 1974; Ateba and Mittal, 1994). Dagerskog and Bengtsson (1974) observed that the maximum total color change was 22, and suggested 10 as a desirable color change. In this optimization problem, 10 and 15 were chosen as lower and upper bounds for surface color change values, respectively. Unlike with microbial inactivation, the intensity of color is increasing instead of decreasing. Thus, relative color change, which is defined by subtracting a color change value from the maximum observable color change value (ΔE_{max}=22), was used to calculate the log reduction as follow:

$$\log\left(\frac{\Delta E_{max} - \Delta E}{\Delta E_{max}}\right) \qquad \textbf{[4.18]}$$

Therefore, 0.263 and 0.497 were used as a minimum and a maximum log reduction of relative color change, respectively.

4.3.5 Practical Considerations for Integration of Models

Log reduction value is a good tool for describing the amount of reduction from initial concentration of a material. However, if the reference D-value is small, which

---

[22] In Humidity/Moisture Handbook by Machine Application Corporation (Sandusky, OH)

means fast reaction, the log reduction value may be too large for computer programming to handle such a large number. For computer programming, the double data type, according to IEEE Standard 754 for double precision, allows +/-308 for maximum and minimum exponent to 10. Thus, floating-point overflow problems occur when the reference decimal reduction time is too small.

Final log reduction of a non-isothermal process can be integrated as:

$$\log\left(\frac{N_n}{N_0}\right) = \log\frac{N_1}{N_0} + \log\frac{N_2}{N_1} + \ldots + \log\frac{N_n}{N_{n-1}} = \frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_1}{z}\right)}} + \frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_2}{z}\right)}} + \ldots + \frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_n}{z}\right)}} \quad \textbf{[4.19]}$$

Even though the $log(N_n/N_{n-1})$ at early stages of cooking has no overflowing problem, $log(N_n/N_{n-1})$ of a later stage can become very large, which results in the abortion of the computer program. To resolve this problem, a multiplier (m) was adopted to slow the process, so that the log reduction number stays within the computational limit. After finishing the process, the original log reduction number was calculated back by using the multiplier. For this, an increased D-value ($\hat{D}_r$) was defined as follow:

$$\hat{D}_r = m \times D_r \quad \textbf{[4.20]}$$

Then the Equation [4.19] can be re-written by using Equation [4.20]. Thus,

$$\log\left(\frac{N_n}{N_0}\right)^* = \log\frac{N_1}{N_0} + \log\frac{N_2}{N_1} + .. + \log\frac{N_n}{N_{n-1}} = \frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_1}{z}\right)}} + \frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_2}{z}\right)}} + .. + \frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_n}{z}\right)}}$$

where, * means resultant log reduction using the increased Dr-value.

[4.21]

Now, dividing Equation [4.19] with Equation [4.21] produces the following result.

$$\frac{\log\left(\frac{N_n}{N_0}\right)}{\log\left(\frac{N_n}{N_0}\right)^*} = \frac{\frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_1}{z}\right)}} + \frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_2}{z}\right)}} + \ldots + \frac{\Delta t / D_r}{10^{\left(\frac{T_r - T_n}{z}\right)}}}{\frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_1}{z}\right)}} + \frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_2}{z}\right)}} + \ldots + \frac{\Delta t / \hat{D}_r}{10^{\left(\frac{T_r - T_n}{z}\right)}}} = \frac{\frac{\Delta t}{D_r}(.....)}{\frac{\Delta t}{\hat{D}_r}(.....)} = \frac{\hat{D}_r}{D_r} = \frac{m \cdot D_r}{D_r} = m \qquad [4.22]$$

Therefore, the original log reduction value can be restored by using the multiplier m. Even though the size of m can vary depending on the size of the maximum exponent, the value of 10^6 was used for microbial inactivation calculation in this research.

4.4 Formulating the Optimization Problem

The objective of the problem is to maximize patty yield by finding optimal process temperature, humidity, impingement air velocity, and cooking duration, while ensuring the target microbial lethality, internal color change, and surface color change. This optimization problem can be formulated with mathematical expressions to apply formal optimization techniques. The formulation of the problem contains three essential components: the objective function, inequality constraints, and equality constraints.

Objective Function:

The objective is to maximize performance index J, which is the final patty yield (%) by finding the optimal process temperature profile $T(t)$, humidity $H(t)$, impingement air velocity $V(t)$, and cooking time t_f over $t \in [0, t_f]$:

$$J = f(t_f) = \frac{\overline{m}_{wf} + \overline{m}_{ff} + 1}{m_{wo} + m_{fo} + 1} \times 100 \qquad [4.23]$$

The final yield, $f(t_f)$, was calculated by dividing the final patty weight by the initial patty weight. The weights were calculated with water and fat content based on non-fat solids.

Inequality Constraints:

Final log reduction of microbial inactivation and internal color change (C_c) of the center of patty should be greater or equal than 6.5 (beef) and 1.387, respectively. Also, the acceptability via surface color change (C_s) log reduction should be between 0.263 and 0.497. These limits are:

$$L(t_f) \geq L_c (= 6.5) \quad \textbf{[4.24]}$$

$$C_c(t_f) \geq C_{c,c} (= 1.387) \quad \textbf{[4.25]}$$

$$C_{s,L} (= 0.263) \leq C_s(t_f) \leq C_{s,U} (= 0.497) \quad \textbf{[4.26]}$$

The control variables (process temperature, humidity, impingement air velocity, and cooking time) are also bounded by upper and lower limits. For temperature, the JSO-IV oven can achieve 260°C and is not normally operated below 100°C. Also, the maximum humidity level is 90%MV, and the maximum impingement velocity is 30.6 m/s. Based on the above actual capacities and reasonable range of cooking duration, the following upper and lower bounds were set:

$$T_L (= 100) \leq T(t) \leq T_U (= 250) \ [^\circ C] \quad \textbf{[4.27]}$$

$$H_L (= 0) \leq H(t) \leq H_U (100) \ [\%MV] \quad \textbf{[4.28]}$$

(Instead of 90% MV, 100%MV was used to test maximum theoretical limit)

$$V_L (= 0) \leq V \leq V_U (= 30) \ [m/s] \quad \textbf{[4.29]}$$

$$t_{f,L}(=60) \leq t_f \leq t_{f,U}(=600)\ [s] \qquad \textbf{[4.30]}$$

Equality Constraints (Process Model):

The finite element process model (Section 4.3) developed for moist air impingement cooking of ground formed meat patties was the equality constraint for this problem.

4.5 Alternative Modeling by Using Artificial Neural Networks

Artificial neural networks were used to replace the finite element process model, in order to speed simulation time. The networks were trained and tested to determine whether the networks were representing the finite element process model sufficiently well. Two different training strategies were applied to two different types of networks. Static training (Section 4.5.3) was used to develop static neural network model (SNNM), and dynamic training (Section 4.5.4) was used to develop dynamic neural network model (DNNM). The first step for training and optimization was to parameterize the control profiles.

4.5.1 Parameterization of the Control Function

To explore the infinite solution domain, an efficient method of representing the dynamic control vectors must be available. In this research, control vectors were parameterized in two ways: by using piecewise linear interpolation, which was used by Banga *et al.* (1997), and by Fourier series.

Piecewise linear interpolation (PLI):

The piecewise linear interpolation method was used to represent a control function with a fixed number of points and final process time. The interval between the

points can vary, but in this application, the control functions were evenly discretized into 20 pieces. Thus, by using those points, any other points can be calculated by using Equation [3.8] (piecewise linear interpolation). Therefore, a total of 21 points, including a flexible endpoint time, were used to parameterize each control function (process temperature, humidity, and impingement velocity). Although the size of interval varied depending on the endpoint time, due to the fixed 21 control points, the effect of the interval size was not considered in this research. Therefore, the maximum interval was 30 [s] for the duration of 600 [s], and the minimum interval was 3 [s] for the duration of 60 [s]. Thus, the total number of parameters to represent three control functions and process duration was 64 (=3×21+1). Each designated point followed the conditions of Section 3.2.3.1 (Equation [3.11-14]).

In the case of linear interpolation, the total 64 discrete values comprise the following structure.

A set of discrete values=$\{p_1, p_2, \ldots, p_{64}\}$, where

p_1~p_{21}: discrete values of temperature

p_{22}~p_{42}: discrete values of humidity

p_{43}~p_{63}: discrete values of impingement velocity

p_{64}: cooking duration

Fourier series (FS):

The Fourier series is a robust parameterization tool. Fourier synthesis can generate all possible continuous functions with a sum of sine and cosine functions called a Fourier series. The function is uniquely defined by constants known as Fourier coefficients, which are shown in Equation [4.31]:

$$T(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kt) + b_k \sin(kt)) \quad \textbf{[4.31]}$$

In this application, the infinite Fourier series was limited to ten terms, which is sufficient to closely approximate even step functions. Also, the period (2π) of cosine and sine functions needs to be adjusted to the cooking duration t_f. Thus, Equation [4.31] was rearranged on the interval $[-t_f, t_f]$ as follow:

$$T(t) = \frac{a_0}{2} + \sum_{k=1}^{10} (a_k \cos(k\frac{\pi \cdot t}{t_f}) + b_k \sin(k\frac{\pi \cdot t}{t_f})) \quad \textbf{[4.32]}$$

Therefore, according to the Equation [4.32], each control profile was parameterized by using 21 Fourier series coefficients for a given process duration.

In the case of Fourier parameterization, the 64 parameters comprise the following structure.

A set of parameters=$\{p_1, p_2, \ldots, p_{64}\}$, where

p_1~p_{20} (*i.e.* a_k, b_k): Fourier coefficients for temperature; p_{21} (*i.e.* a_0): shift

p_{22}~p_{41}: Fourier coefficients for humidity; p_{42}: shift

p_{43}~p_{62}: Fourier coefficients for impingement velocity; p_{63}: shift

p_{64}: cooking duration

4.5.2 Training and Validation Data Groups

As previously mentioned, training and validation data groups for neural networks were produced by using the integrated FEM. In this study, a total of five neural networks (one DNNM and four SNNM) were trained and validated with various groups of data. Each group of data had a different number of input-target data sets and contents

depending on process type, which creates complexity. Thus, each group of data sets was named for further reference (Table 4.4). In addition, the detailed conditions for each data group are provided in Table 4.5. For example, the alternative model SNNM_S was trained by using D3 and validated with D7 (Table 4.4), and the conditions for those data sets are reported in Table 4.5.

Data group D2 was generated with 63 Fourier coefficients and a process time. Fourier coefficients (Equation [4.32]), a_k and b_k, were selected randomly between -100 and 100, and a shift value (a_0) was also chosen between 5 and 200. Among these random combinations, only the profile satisfying the upper and lower bounds of the control variables (temperature, humidity, and velocity) were accepted and applied to the FEM to generate results.

Table 4.4 The names and contents of data groups for training and testing neural networks.

Neural Networks		Training Data Group		Testing Data Group	
Type	Name	Name	Contents	Name	Contents
DNNM	DNNM	D1	910 constant processes (Table 4.5) (54,600 conditions/states sampled out for training)	D2	1,000 processes were generated with 64 random Fourier coefficients including cooking time.
				D7	1,000 processes (randomly generated constant processes with random cooking duration)
				D4	2187 processes (double-stage, Table 4.5)
				D5	2187 processes (multi-zone, Table 4.5)
SNNM	SNNM_R	D2		D6	1,000 processes were generated with 64 random Fourier coefficients including cooking time. (≠D2)
	SNNM_S	D3	1,296 processes (single-stage, Table 4.5)	D7	1,000 processes (randomly generated constant processes with random cooking duration)
	SNNM_D	D4		D8	1,000 processes (randomly generated double-stage conditions)
	SNNM_M	D5		D9	1,000 processes (randomly generated multi-zone conditions)

Table 4.5 Conditions of each process type to produce training data groups; D1, single-stage (D3), double-stage (D4), and multi-zone (D5) process.

Process types	Control Variables								Total number of data sets
	T [°C]		H [% MV]		V [m/s]			t_f [s]	
Constant Process* (D1)	100, 112.5, 125, 137.5, 150, 162.5, 175, 187.5, 200, 212.5, 225, 237.5, 250		1, 15, 30, 40, 50, 60, 70, 80, 90, 100		1, 5, 10, 15, 20, 25, 30			600	910
Single-Stage (D3)	100, 130, 160, 190, 220, 250		0, 20, 40, 60, 80, 100		0.5, 6.4, 12.3, 18.2, 24, 30			60, 180 300, 420 540, 660	1,296
Double-stage (D4)	T_1	T_2	H_1	H_2	V_1		V_2	t_f	
	100	100	0	0	0.5		0	60	
	175	175	50	50	15.25		15.25	330	2,187
	250	250	100	100	30		30	600	
Multi-zone (Sec. 4.7.3) (D5)	T		H		V_1	V_2	V_3	V_4	t_f
	100		0		0.5	0.5	0.5	0.5	60
	175		50		15.25	15.25	15.25	15.25	330
	250		100		30	30	30	30	600

* Process conditions remain constant during cooking

4.5.3 Static Training

Depending on the characteristics of the training data sets and the architecture of neural network, the mapping relationship between input and output vectors may be static, where each application of a given input vector always produces the same output vector (Wasserman, 1993). Therefore, a neural network trained by this method cannot account for process history; rather, it merely identifies a relationship between initial conditions and results. This kind of neural network has to be trained whenever different process types are engaged.

For SNNM_R (Table 4.4), 1,000 random time-varying conditions were used to generate data sets by using the finite element process model. Some examples of random time-varying conditions are illustrated in Figure 4.5.

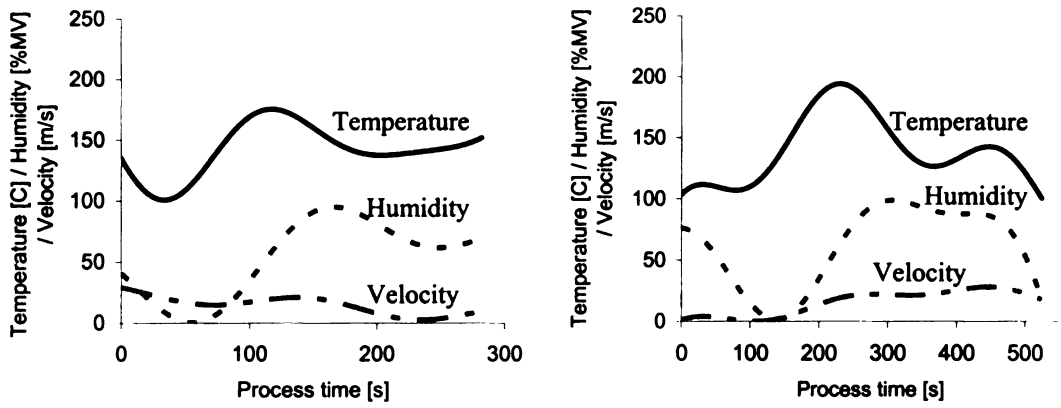


Figure 4.5 Examples of random time-varying conditions by using Fourier series control profile parameterization.

Each training data set consisted of 64 parameters, which consisted of 21 randomly generated Fourier coefficients for 3 different control vectors (temperature, humidity, and impingement velocity) plus process duration. Those 64 parameters were used as the system input variables. By using the 64 parameters and Fourier series (Equation [4.32]), three continuous random profiles were generated and used for the finite element process model input to predict the yield, microbial lethality, internal color change, and surface color. The outputs of the finite element model were used as target data sets. The data group (D2) of combined input and target data sets was used to train the SNNM_R (Section 3.3.2). Those data groups were also used for validation of GRNN_R.

For the training of SNNM_S, SNNM_D, and SNNM_M, training data groups (D3, D4, and D5) were generated according to their conditions in Table 4.5.

4.5.4 Dynamic Training

Compared to the static training method, the dynamic training method teaches system behavior so that the neural network can predict output depending upon previous, as well as current, input and/or output (Wasserman, 1993). Therefore, the dynamic neural network reacts like a physical model, not just an input-output vector mapping.

If a full factorial design is planned with 64 parameters and just two extreme values for each parameter, it results in 2^{64} combinations, which is an impractical number of trials. However, dynamic training can capacitate the neural network to understand system behavior with a limited amount of information or training data set. Morimoto *et al.* (1997) used historical input and output data to describe the dynamic characteristics of fruit color changing behavior with time and temperature. The only disadvantage of dynamic training is that it takes a little more time to produce outputs, because the neural network has to predict the next point based on the previous point until it gets to the final point.

Depending on the interval size, the accuracy of the prediction also changes. The smaller the interval is, a more accurate result is possible. In this research, process temperature, humidity, and impingement velocity were used as system control vectors, and yield, microbial lethality, internal color change, and surface color were included in states (Figure 4.6). In addition, past time was included in the inputs as a factor accounting for history of the system until it became past state. Morimoto *et al.* (1997) added linear data to the input of neural network and observed that the identification accuracy for any cumulative responses was significantly improved. However, the linear data were replaced with actual current time of the system.

To obtain the training data group D1, 910 constant process conditions (Table 4.5) were produced, and the results were calculated with the finite element model. The combination has a maximum possible duration (600 [s]) by which other durations can be covered, because the process is constant.

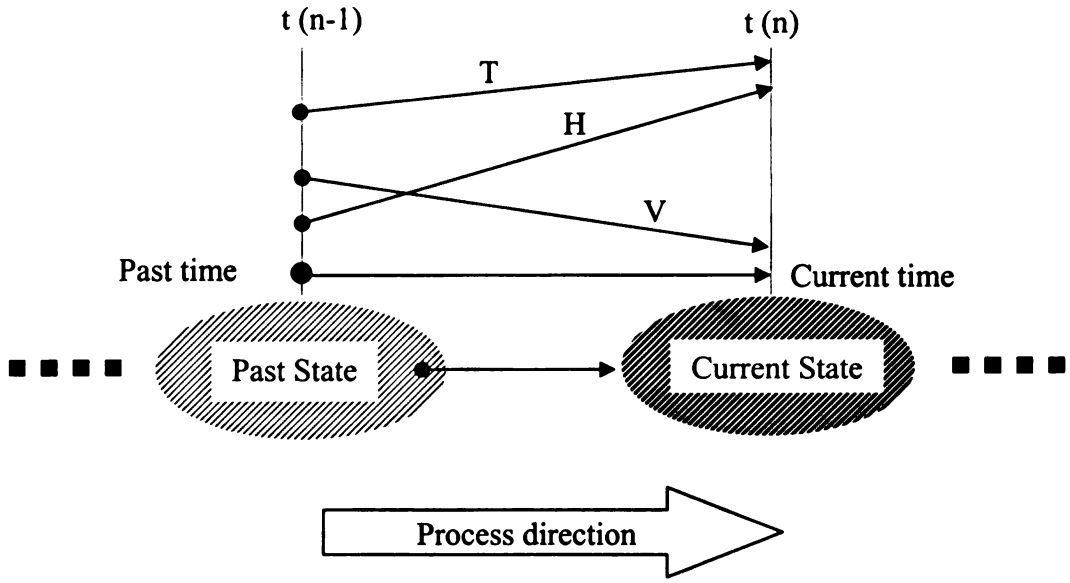


Figure 4.6 Conceptual diagram of dynamic training paradigm.

Out of the 910 resultant processes, 54,600 data sets were sampled with 10 seconds interval. By using the sampled data, input and output pairs for neural network training were reconstructed. Figure 4.7 shows the structure of input/output pairs.

The control force vectors, which have direction and magnitude, are implicitly dissolved in the past and current controls. Although the four current outputs can be predicted with a single GRNN, an individual GRNN for each output was trained to increase accuracy for each prediction (Trelea *et al.*, 1997). The network parameters were also optimized independently.

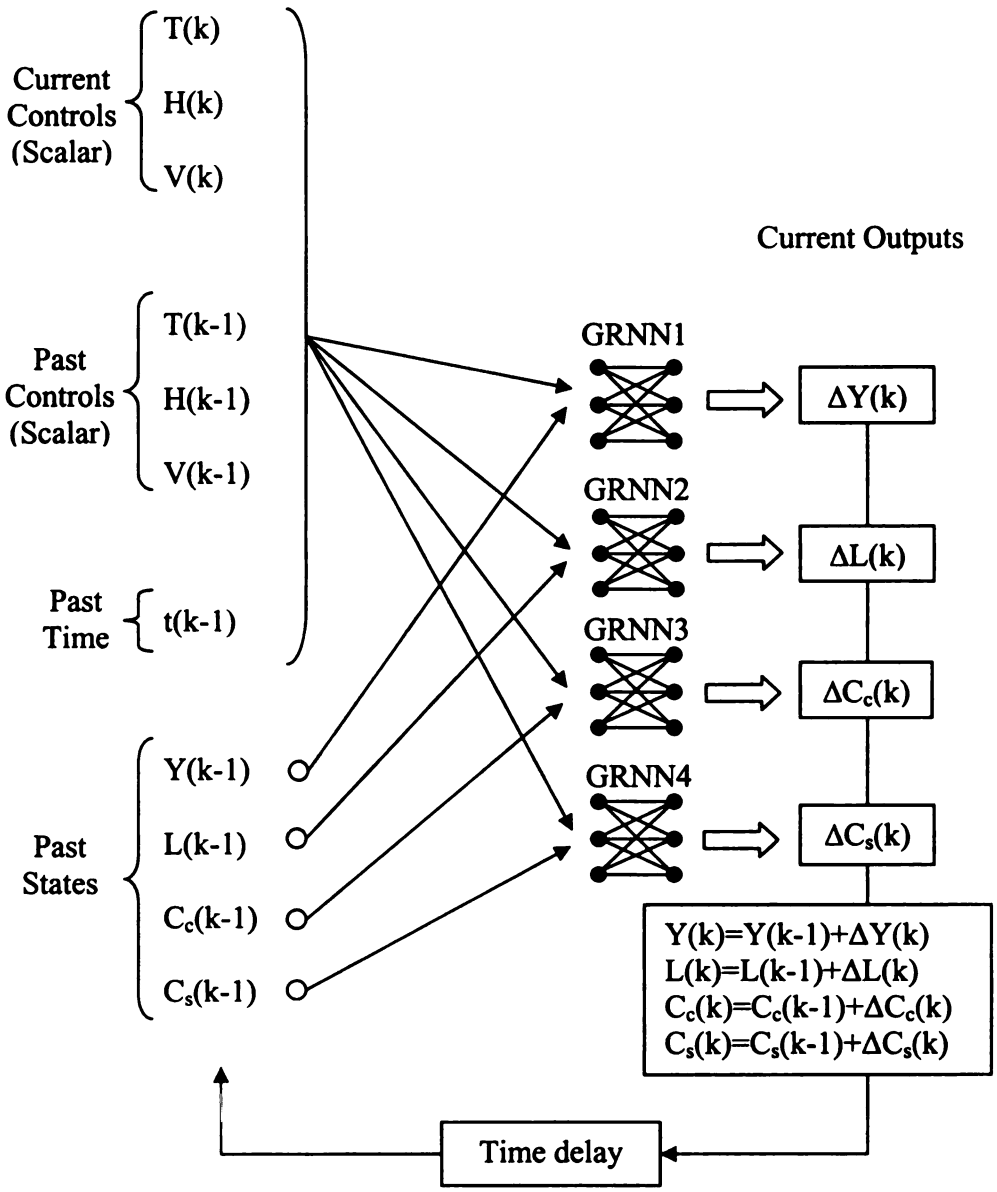


Figure 4.7 The structure of an input/output data pair for dynamic training to identify dynamic characteristics of meat patty cooking.

How well the input data are represented to the network is a critical issue to a successful application of artificial neural networks (Wasserman, 1993). There are two important points to be considered in the training of the GRNN. First, the target vectors (yield and the other outputs) are monotonic, which are always increasing or decreasing in one direction. If the GRNN is trained with actual values, then there is a possibility for the GRNN to forecast a value that does not coincide with the monotonic characteristics of output target vectors. This abnormality of forecasting can be minimized by training the network not with actual values but with the difference. At the end of prediction, a cumulative summation for each time was calculated to get the actual value for each time, which renders all the training data to be always positive or negative.

Secondly, thermal inactivation kinetics of a small *z*-value generate large log reduction values, because of high sensitivity to temperature change. The z-values for thermal inactivation of *Salmonella* and the depletion of Myoglobin (internal color change) are 5.9 °C and 10.41 °C, respectively, which are relatively smaller than 90.55 °C for kinetics of surface color change. Therefore, there are huge variations in the predicted results. The large variations in the magnitude of the components of a vector may not convey meaningful information, but can confuse the network (Wasserman, 1993). In the above case, raw data range from zero to several orders of magnitude, while the critical or valuable information is located in a small range. Wasserman (1993) suggests that "taking the logarithm of the data will adjust the range so that large values can be 'squashed' more than small values, thereby allocating a constant range to a given percentage deviation", which is called nonlinear normalization. In this research, nonlinear normalization using

base-10 logarithm was used for the training GRNN with the above two reaction kinetic data (*Salmonella* inactivation and internal color change).

4.5.5 Neural Network Validation

All the neural network based models were validated with the testing data groups that were prescribed in Table 4.4. For all the validations, root mean squared errors for yield, microbial lethality, internal color change, and surface color were calculated to quantify the performance of the neural networks.

4.5.6 Neural Network Parameter Optimization

FFNN based models consist of layers and neurons embedded in the hidden layer. For the best performance of the network, the number of layers and neurons must be optimized. Generally, there is no specific rule to set these numbers except by trial-and-error. In this specific application, the number of layers was two, and nine combinations of the numbers of neurons for each layer were examined in terms of root mean squared error to find the optimal number of neurons in each layer. For each layer, 5, 10, and 15 neurons were selected for the combinations. For each case, 500 data sets were used to test each combination.

DNNM has a fixed number of layers and neurons according to the principles of network architecture (Section 3.3.3). Therefore, the size of the receptive field that determines the robustness of the network needs to be optimized by using trial-and-error. The parameter is represented as "spread" in the actual computer code. With varying "spread", the results in terms of RMSE were compared, and the value shows the

minimum RMSE was selected as an optimal "spread." For this trial-and-error procedure, 100 data sets from D2 were used to test each case.

4.6 Process Optimization Strategies

4.6.1 Combinations of Techniques

The optimization techniques utilized in this research are global optimization algorithms, which can be coupled with any type of process model. Thus, various combinations of optimization strategies are possible, depending on the optimization algorithm, type of process model, and method of control function parameterization. Table 4.6 shows the optimization strategies that were evaluated in this study. The numbers in Table 4.6 were used throughout this study to refer to a specific optimization strategy.

Table 4.6 Optimization strategies (by strategy number), according to their process model, the method of control profile parameterization, and optimization algorithm.

		Optimization Algorithms		
Model	Control profile parameterization	GA	SA	ICRS/DS
FEM	PLI	1	2	3
	FS	5	6	7
DNNM	PLI	9	10	11
	FS	13	14	15
SNNM_R	PLI	N/A	N/A	N/A
	FS	21	22	23
FEM	Single-stage	N/A		25
	Double-stage			29
	Multi-zone			33
DNNM	Single-stage			26
	Double-stage			30
	Multi-zone			34
SNNM_S	Single-stage			27
SNNM_D	Double-stage			31
SNNM_M	Multi-zone			35

PLI: piecewise linear interpolation; FS: Fourier series

SNNM_ R was not trained with actual control values, but with Fourier coefficients. Thus, control function parameterization with PLI could not be applied, because SNNM was specific for trained data type. All of these combinations (Table 4.6) were designed to find ideal and continuous optimal control profiles.

4.6.2 GA Based Strategy

Genetic algorithms (Section 3.2.1) have some parameters that must be set before running the algorithm properly. For genetic algorithms, setting a convergence criterion using the objective function value is difficult, because sometimes the same objective function value wins the competition for several or more generations. Therefore, the total number of generation cycles was set to 400 as a default value, because most of the significant convergences tended to occur around 200 generations, according to prior observations. The algorithm was terminated when it reached the target generation number. However, if convergence was not achieved at the final number of generations, the optimization process was continued until no improvement was observed. The size of initial population and the population of each generation was set to 10 in this research, because the number is not too large to demand too much computation, nor too small to cause slow convergence. In addition, there is no specific rule for setting the number in the initial population (Venkataraman, 2002). Other parameters are also problem-specific. If large numbers are selected for the other parameters, it will demand huge computation time, depending on the process model. Thus, rate of simple crossover, arithmetic crossover, mutation, and immigrants of each generation were set to produce 6, 6, 4, and 2 offspring, respectively. Mutation rate was 0.016 (=1/64), which replaced one parameter with a random value among 64 parameters. At the end of each generation, all the

population was tested to identify qualified individuals that satisfied all the constraints, and they were moved to the next generation.

4.6.3 SA Based Strategy

As indicated in Section 3.2.2, there are several critical parameters for a simulated annealing algorithm, such as cooling schedule, number of Markov chains at a temperature, transitional probability function, and convergence criteria.

The following cooling schedule was used:

$$T_k = \frac{A}{k+1} + B \qquad \textbf{[4.33]}$$

where,

$$k = 1 \ldots N$$
$$A = \frac{(T_0 - T_N)(N+1)}{N}$$
$$B = T_0 - A$$

The above cooling schedule generates more moderate acceptance probability (between 0.5~0.8) than high acceptance probability (above 0.9), which reduces exhaustive exploration at the initial search. The initial temperature, final temperature, and the number of cooling steps were set to 25, 0.001, and 150, respectively. These optimization parameters generate a probability curve (Figure 4.8) showing less exhaustive exploration in the early stage, which is desirable in this specific application.

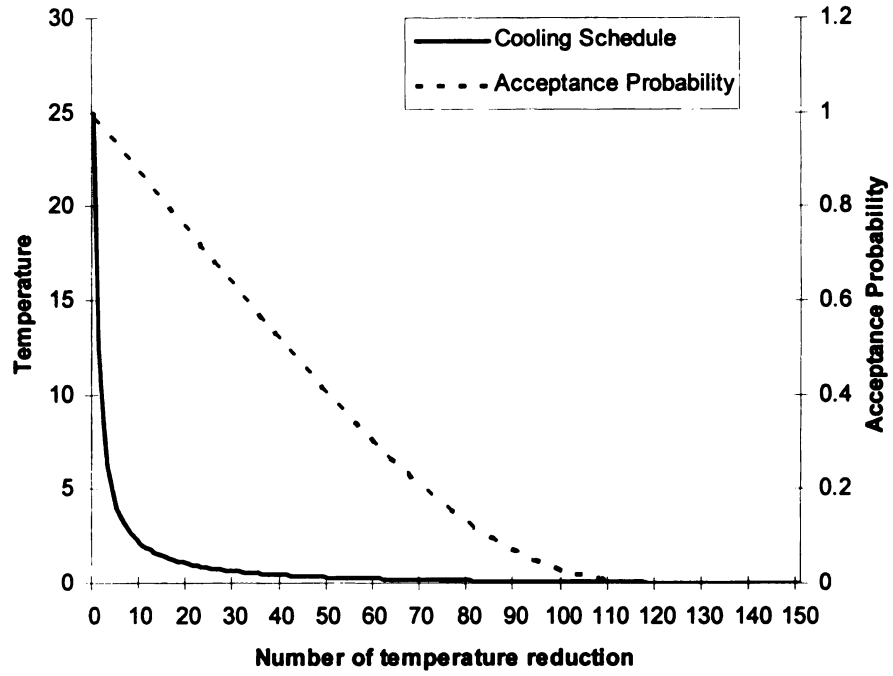


Figure 4.8 Acceptance probability (Boltzmann probability) along with cooling schedule (T_o=25, T_N=0.001, N=150, and average increment of accepted objective value=0.3) for SA.

In this research, a Boltzmann probability distribution function was used as a transitional probability function (acceptance probability), which is classic and simple. The algorithm was coded to terminate its search when there is no acceptance and no improvement of the objective function. The number of iterations at each cooling cycle, which is the length of Markov chain, was set to 20 from prior observations.

The algorithm generates the next move by using random search direction and stepsize for each parameter or discrete value. Variable stepsizes were used, depending on the control vectors (temperature, humidity, and impingement velocity) and type of control function representation (discretization or parameterization), because the

sensitivity of each parameter or value is different. Depending on the structure of the control vector (Section 4.4.1), different step-sizes were used as follows:

Linear interpolation parameterization:

- Stepsize[p_1~p_{21}]=5; temperature profile
- Stepsize[p_{22}~p_{42}]=5; humidity profile
- Stepsize[p_{43}~p_{63}]=5; impingement velocity profile
- Stepsize[p_{64}]=10; cooking duration

For Fourier parameterization:

- Stepsize[p_1~p_{20}]=2; stepsize[p_{21}]=10; temperature profile
- Stepsize[p_{22}~p_{41}]=2; stepsize[p_{42}]=5; humidity profile
- Stepsize[p_{43}~p_{62}]=2; stepsize[p_{63}]=2; impingement velocity profile
- Stepsize[p_{64}]=10; cooking duration

The search direction was produced by using uniform random distribution of interval between -0.5 and 0.5.

4.6.4 ICRS/DS Based Strategy

The ICRS algorithm has three important heuristic parameters, k_1, k_2, and n_e. The k_1 parameter controls the size of the search step by changing the magnitude of the standard deviation vector. However, if the search fails to find improvement before the number of trial reaches the failure counter (F), the standard deviation vector is subsequently reduced by using k_2. The n_e integer parameter controls the rate of convergence by changing the failure counter value. Pan (1998) recommended default values of k_1=1/3, k_2=1/2, and n_e=4. However, the stepsize of search depends on the method of control profile parameterization. If the control parameters are highly sensitive,

then k_l needs to be reduced. A k_l value of 0.05 was used for the case of Fourier parameterization, because the Fourier coefficients are more sensitive than the discretized values in PLI parameterization. For the case of linear interpolation parameterization (PLI), the default value k_l=1/3 was used.

According to the original ICRS/DS algorithm, the failure counter (F) is n_e times twice the problem dimension. The problem dimension of this research is 64. Thus, the failure counter must be 512, which demands too much computation time. By using some trial runs, the failure counter was set to 100, because it was rare to reach 100 or more. For convergence criterion, Equation 3.19 was used with the tolerance value of 1.5×10^{-5}.

4.6.5 Benchmark Test for Optimization Algorithms

Even though the principles of GA, SA, and ICRS optimization algorithms are well established and proven, the developed computer codes must be validated for their effectiveness. Especially, in this application, a multi-variable problem is desirable, because the optimization problem of this study is multivariate. The Bezier[23] curve is a good tool to observe the behavior of all the variables in two-dimensional space (Venkataraman, 2002). A fifth-order Bezier curve-fitting problem was presented to each computer code. The fifth-order Bezier curve is:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} P_i (1-t)^{n-i} t^i, \quad t \in [0,1] \qquad \textbf{[4.34]}$$

where,

[23] Bézier curves were widely publicized in 1962 by the French engineer Pierre Bézier, who used them to design automobile bodies. The curves were developed in 1959 by Paul de Casteljau using de Casteljau's algorithm. Bézier curves are widely used in computer graphics to model smooth curves. (http://en.wikipedia.org/wiki/Bezier_curve)

n=degree

$\binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \cdots, n$ Bernstein basis polynomials

For n=5, the fifth order Bezier function becomes:

$$B(t) = P_0(1-t)^5 + 5P_1 t(1-t)^4 + 10P_2 t^2(1-t)^3 + 10P_3 t^3(1-t)^4 + 5P_4 t^4(1-t) + P_5 t^5$$

[4.35]

Equation [4.35] was used to minimize root mean squared error with the following curve:

$$f(x) = 1 + 0.25x + 2e^{-x}\cos 3x \qquad \textbf{[4.36]}$$

A total of eight design variables were used to fit the above curve. All the computer codes associated for this benchmark test were adopted from the works of Venkataraman (2002).

4.7 Case Studies

Global optimization techniques developed and tested in the above sections were applied to practical situations to validate their potential application to real, industrially-relevant processes. Even though the optimal control profiles are the "best" in terms of mathematical application of the theories, it is practically challenging to apply the theoretically "best" profile in the real life situation, given constraints of existing equipment and knowledge. Therefore, a series of case studies (single-stage, double-stage, and multi-zone) that are currently available, or at least practical with minimal changes of cooking system configuration, were solved and compared to each other.

4.7.1 Single-Stage Oven

The single-stage oven case represents an oven (JSO-IV) using constant control profiles during its process. The control variables were constant temperature, humidity, impingement velocity, and cooking duration. Because the oven uses a constant profile, the actual number of control variables can be reduced down to four instead of 64, which was used in the previous applications.

Three different models were used in this case study, FEM, DNNM, and SNNM_S. The previously developed FEM and DNNM models were used without alterations. However, the SNNM_S, was trained with the D3 training data group, because the static neural network model was specific for training data set. The FEM, GRNN, and SNNM_S were coupled with SA, GA, and ICRS/DS, respectively.

4.7.2 Double-Stage Oven

The double-stage oven case is considered as the second possible option that occurs in real commercial applications, by connecting two single-stage ovens in series. This is a relatively simple solution for the food processing industry. The double-stage system generates an equidistant, two-step constant control profile, which has more dynamics in control than does the single-stage oven. Although more capital cost is necessary to purchase an additional unit, increased yield and production rate may justify cost. In this case study, only the total yield of the system was considered and optimized.

The total number of control variables was seven, given two steps for each control profile and a cooking duration. FEM and DNNM were not altered, but the SNNM_D was trained with 2,187 data sets of data group D4, because the SNNM_D was a static model. The 2,187 training data sets were combinations of three different levels and the seven

control variables (Table 4.5). The FEM, DNNM, and SNNM_D were coupled with GA, SA, and ICRS/DS, respectively.

4.7.3 Multi-Zone Oven

This configuration may be a suggestion for the oven manufacturing industry. Among the three major control variables (temperature, humidity, and impingement velocity), varying impingement velocity is the easiest design change to implement in the existing system, because the local jet exit velocity can be controlled by the size of the jet exit or other flow control mechanisms. By observing this case study, the effect of a multi-zone concept was tested.

Control temperature and humidity remained constant throughout the process, and four equidistance zones were considered for different impingement velocities. Thus, the total number of control variables was seven, including the process duration. SNNM_M was trained with 2,187 data sets of data group D5, which were the combinations of three different levels for each variable according to Table 4.5.

4.7.4 An Example of Economic-Based Optimization

As the level of optimization is expanded from a single unit operation to an entire plant, the best control profile of the single unit may be reevaluated to improve the overall objective of the plant. Compared to unit operation optimization, economic-based optimization requires much more information related to management, such as product value, energy cost, labor cost, warehouse management, raw material prices, and so on. In addition, the objective can vary, such as minimizing energy consumption, maximizing processing rates, maximizing profit, or maximizing a practical quality attribute.

In this case study, capacity and geometric information for the oven was obtained from specifications of the JSO-IV (model 4044). For simplicity, raw material feeding rate, energy cost, and final product yield were considered as factors influencing the net profit of an entire system. Even though many of the values for these factors were "best guess", it was considered adequate to illustrate the concept of economic-based optimization. To make the problem as a single objective optimization, all the factors were converted into price ($) with the following equations. The economic-based optimization problem was defined as follow:

Objective function (f): maximizing profit [$]

f = *product value − feed cost − energy cost* :

- Energy cost ($/hr) = (steam cost) + (electricity cost) + (thermal cost)

$$= \alpha \cdot f_H(H) + \beta \cdot (f_V(V) + f_B(B)) + \gamma \cdot \kappa \cdot f_T(T)$$

- Product value ($/hr)= $\mu \cdot Y$,where $Y = f_Y(H,V,T,t) \cdot F$
- Feed cost (raw material: frozen beef patties)($/hr)= $\phi \cdot F$,where $F = 3600 \cdot \lambda \cdot B$

Variables:

- $f_Y(T,H,V,t)$ [%]: process model
- $f_H(H) = \theta \cdot H$ [kg/h]: steam consumption rate as a function of H
- $f_V(V) = \upsilon \cdot f \cdot A \cdot V$ [kW]: electrical demand as a function of V
- $f_B(B) = \dfrac{m \cdot B^2}{2 \cdot (L/B)} \cdot 10^{-3}$ [kW]: electrical demand for transporting patties

  (kinetic energy required to transport patties on conveyor belt was divided by residence time of a patty(L/B))

- $f_T(T) = \left(\frac{500{,}000}{150} \cdot (1.8T + 32) + 1{,}170{,}000\right) \cdot (1055.04)$ [J/h] (Fulton® thermal fluid boiler (capacity: 2.2 MMBtu/h) was modeled. A linear relationship between operating temperature and power was assumed)

- H: humidity set point [%MV]
- V: impingement velocity set point [m/s]
- T: temperature set point [°C]
- f_H: amount of steam as a function of humidity set point [kg]
- f_V: electricity as a function of impingement velocity set point [kW]
- f_T: amount of natural gas as a function of temperature set point [J/h]
- f_Y: cooking model output or yield [%]
- f_B: electricity as a function of belt speed [kW]
- Y: production rate [kg/h]
- F: feed rate [kg/h] as a function of process duration
- B: belt speed [m/s] (from (24 ft)/(60 s) to (24 ft)/(600 s)) as a function of process duration

Parameters:

- α=0.0092: steam cost [$/kg]
- β=0.30: electricity cost [$/kWh]
- γ=0.18: natural gas cost [$/kg]
- μ=7.71: product value [$/kg] (3.50 $/lbs)
- φ=4.19: raw patty price [$/kg] (1.90 $/lbs)

- $\lambda \approx 4$: loading capacity of patties per unit length of oven [kg/m] (258 patty (120×10mm) on cooking area assumed)
- $\theta \approx 5$: amount of steam per hour proportional to the process humidity (linear relationship was assumed from 90%-1,000 lbs/hr required) [kg/(h·%MV)]
- $\upsilon = 0.33$: electricity demand unit flow rate of air [kW/(m^3/h)]
- $\kappa = 2E\text{-}8$: kg of natural gas to produce 1 J [kg/J]
- f: fraction of nozzle open area (0.0683): total slot open area per total belt area or cooking zone area
- L=7.32: cooking zone length (24 ft) [m]
- w=1.85: width of conveyor belt [m]
- A=L×w=13.6: cooking zone area (146 ft^2) [m^2]
- m=64.6: total product weight on cooking area (43×6=258 patties×113.5 g=29.3 [kg]

Control variables:

- T: Process temperature [°C]
- H: Process humidity [%MV]
- V: Impingement exit velocity [m/s]
- t: Process duration [s]

Explicit constraints:

- Upper and lower bound of T, H, and V (Equation [4.26-28])
- Cooking duration can be expressed via belt speed:

$$B_{f,L}(=0.012) \le B \le B_{f,U}(=0.12) \ [m/s]$$

Implicit constraints:

- The constraints, such as microbial lethality, internal color change, and surface color, are imbedded in the cooking model implicitly.

Assumptions:

- JSO-IV was assumed to have no transitional (inlet or outlet) cooking/equilibration zone (*e.g.*, steam tunnel), and was therefore assumed in this entire project to be just a impingement zone.
- Only steady-state operation was considered.

The above objective function seems like a linear function, but it is nonlinear, because of the nonlinear cooking model. Thus, global optimization techniques were applied to the problem. Different cooking models (FEM and SNNM_S) were used for comparison. Even though the parameters of the above equations were estimated by guess, the effect of parameters was assessed through sensitivity analysis.

4.8 Programming and Computation Tools

A high level computing language, MATLAB® (The MathWorks, Inc., Natick, MA), was used for all the computer programming of the research, because of its convenience of handling matrix and vector formulations, which are the major operations of artificial neural networks, optimization algorithms, and finite element modeling. Also, neural network toolbox 4.0.1 in the MATLAB 7.0.0 provides neural networks, such as GRNN and FFNN, which were used in the computer codes.

High performance PC's (3.2 GHz, Intel Pentium 4 hyper threaded) were used to run the optimization algorithms using GRNN and FFNN. To run optimization algorithm

coupled with FEM, server computers (Sun Fire v880, 4 UltraSPARC-III 750MHz 64 bits CPU) were used.

By using those computers, approximately >20,000 FEM runs, > 20,000 ANN runs, and >100 optimization runs were calculated. In total, all of these runs required an estimate of >250 h of CPU time.

5 RESULTS AND DISCUSSION

5.1 Integrated Process Model Performance

The integrated process model was developed by adding quality prediction kinetics (Section 4.3.3 and 4.3.4) to the existing finite element model (Watkins, 2004) that was previously validated (Section 4.3.1). The performance of the integrated model was tested qualitatively with various control profiles, but not quantitatively, because of the lack of actual experimental data encompassing the complex output from the model. Even though the accuracy of the model was not measured with experimental values, a reasonable working model is sufficient to test the effectiveness of various optimization techniques and alternative models. Figure 5.1 and Figure 5.2 are example model predictions under constant control profiles and dynamic control profiles, respectively.

Although there were huge log reductions in some cases, the predicted log reduction of *Salmonella* and internal color change were used without truncation at the target log reductions (6.5 and 1.4, respectively) to increase the performance of neural networks. In some cases, instability of the prediction of yield and surface temperature were observed at the very early stage. However, the instability was minimized by decreasing the time step of model calculation from 1 to 0.5 second.

The surface color prediction in the moist process condition (Sec. 4.2.4) was well represented in Figure 5.1 and Figure 5.2, which represent high and low humidity process conditions, respectively. The surface color change in Figure 5.1 showed a very slow rate throughout the process, because the surface temperature could not reach the process dew point temperature (≈97.2 °C dew point, 90%MV). Under the ambient dew point temperature, condensation of moisture is predominant on the surface of a patty, which

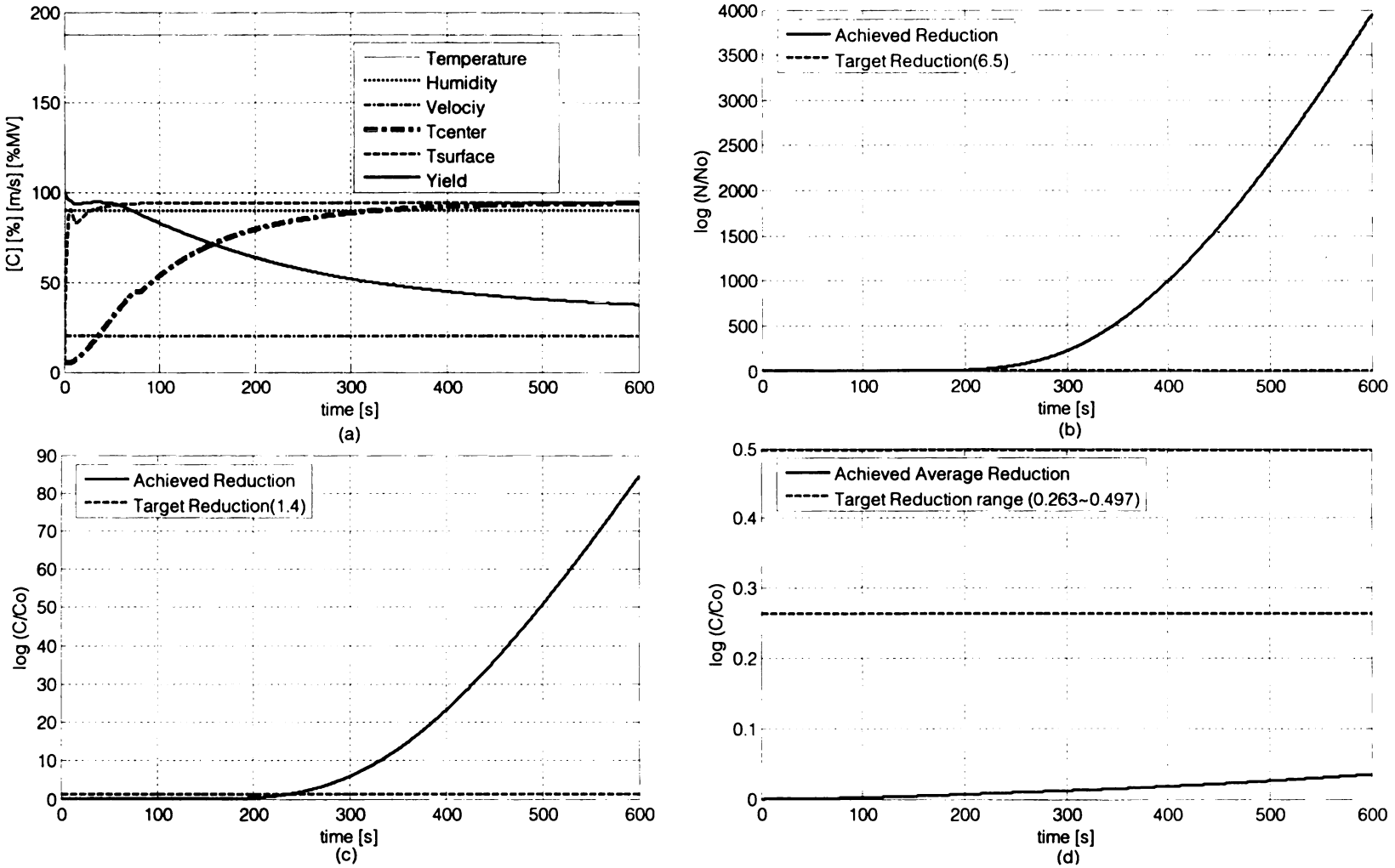


Figure 5.1 An example of the integrated process model performance under constant process condition: (a) Process conditions, yield, center, and surface temperature; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

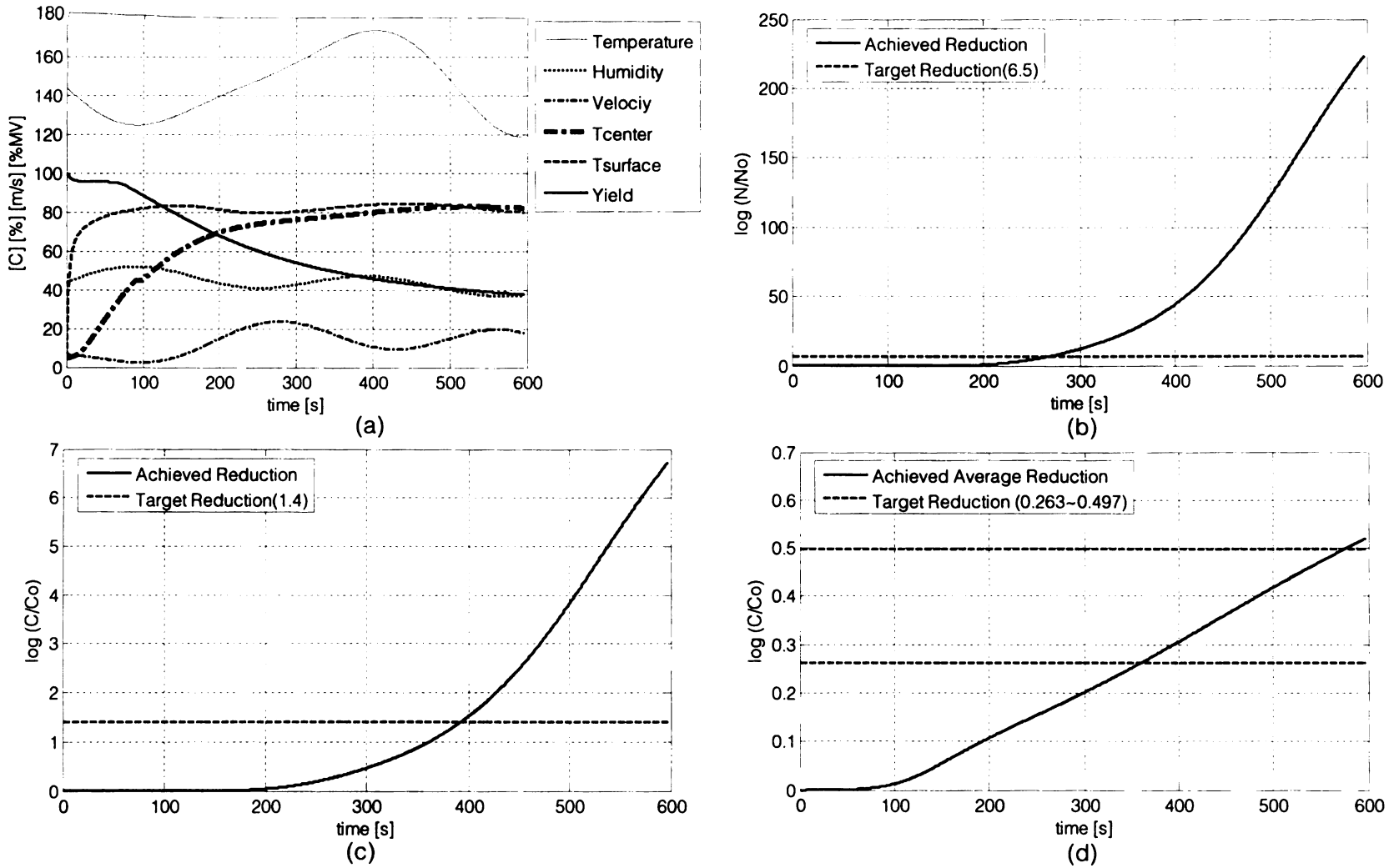


Figure 5.2 An example of the integrated process model performance under dynamic process condition: (a) Process conditions, yield, center, and surface temperature; (b) *Salmonella* inactivation: (c) Internal color change; (d) Surface color change.

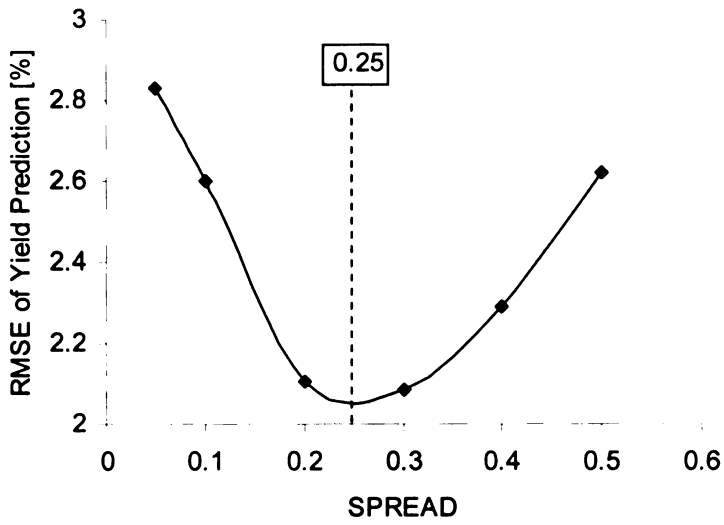
suppresses the browning reaction. However, the surface temperature in Figure 5.2 was much higher than the highest dew point temperature (≈81.7 °C dew point, 50%MV), which accelerated the simulated browning reaction on the surface. In Figure 5.2(d), surface color change remained within upper (0.497) and lower (0.263) bounds for about 200 seconds in the later stage.

5.2 Training and Optimizing Neural Networks

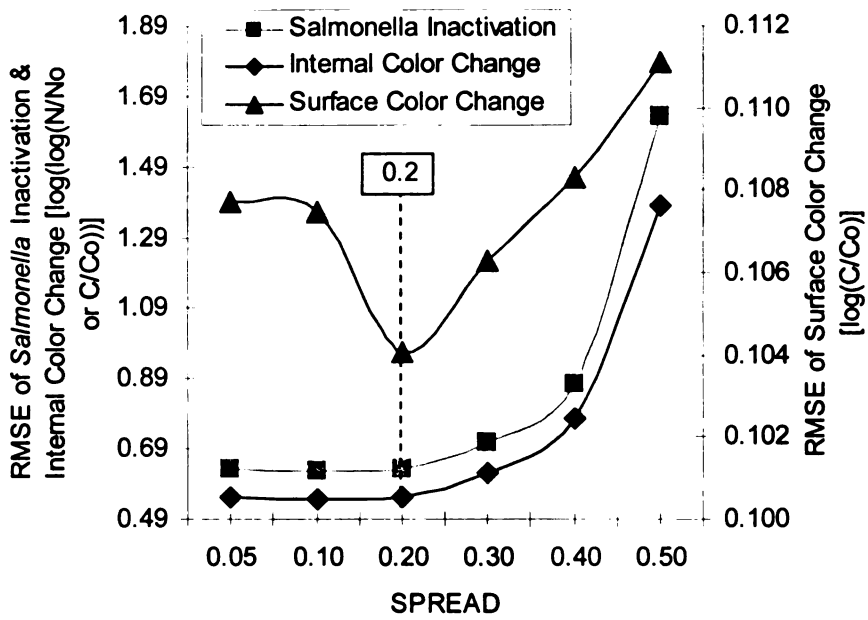
5.2.1 Dynamic Neural Network Model (DNNM)

The DNNM was developed by using four GRNN trained with a training data set (in data group D1) consisting of 53,442 states that were sampled out of the 910 results of the finite element model, according to the dynamic training methodology of the Section 4.4.3. The only network parameter of the GRNN was "spread", which determines the size of receptive field of the network. In this study, each "spread" was determined at a point minimizing RMSE of each GRNN (yield, *Salmonella* inactivation, internal color change, and surface color change) in DNNM. Firstly, the networks were trained with data group D1, and their prediction RMSE for each trial "spread" was computed by using 100 random data sets from data group D2 and plotted with respect to each trial "spread" (Figure 5.3).

The DNNM consisted of four different GRNN. For the yield prediction GRNN, 0.25 was determined as the optimal "spread" around the lowest RMSE. For the other constraint GRNN, 0.2 was to be a commonly optimal "spread." Thus, those optimal "spreads" were used to train each GRNN that was specific for each of the four outputs.



(a)



(b)

Figure 5.3 Optimal "spread" values at the lowest RSME were determined by trying "spread" for each GRNN in DNNM: (a) Yield prediction; (b) *Salmonella* inactivation, internal color change, and surface color change.

5.2.2 Static Neural Network Models (SNNM)

5.2.2.1 SNNM for random and single-stage process

The optimal number of neurons for each hidden layer needs to be determined for SNNM_R (SNNM for random process), because the model uses a feed-forward neural network. The optimal number of neurons of the two hidden layers was determined by simulating 9 different combinations. The combinations were designed with full factorial (5, 20, and 40 neurons). The RMSE for yield, *Salmonella* reduction, internal color change, and surface color change were calculated for each combination. However, the RMSE of each category could not be compared with each other, due to the scale of each category. Thus, to determine the optimal combination of neurons, the RMSE of each category was normalized with its possible maximum value (100, 10, 10, and 2) and averaged. Then the averages of the normalized RMSE were compared with each other.

The lowest average of the normalized RMSE (0.029) was observed with the combination of 40 and 20 neurons in the first and second hidden layer, respectively (Figure 5.4). Therefore, two hidden layers containing 40 and 20 neurons were used for the architecture of SNNM for the random process (SNNM_R). This architecture was also used for the training of SNNM for a single-stage process (SNNM_S), without validation, because the effect of the variation of the number of neurons was insignificant by observation.

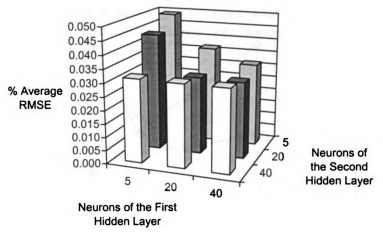


Figure 5.4 Average normalized RMSE of each combination for the number of neurons in each hidden layer of SNNM_R.

5.2.2.2 SNNM for double-stage and multi-zone process

The SNNM for double-stage (SNNM_D) and multi-zone processes (SNNM_M) was developed by using GRNN instead of FFNN, because of the poor performance of FFNN by trial and observations. Various values for "spread" were tested in terms of RMSE. Because the SNNM had a neural network to predict yield, *Salmonella* reduction, internal color change, and surface color all at the same time, the "spread" should be chosen by negotiating among RMSE of the outputs. The "spread" for SNNM_D was negotiated around 0.6, which gives more accuracy to constraints prediction than goal prediction (Figure 5.5). Also, 0.55 was chosen for the "spread" of GRNN_M by negotiating accuracies among outputs (Figure 5.6). Other values can be used as long as those values are not offset too much from a reasonable "spread."

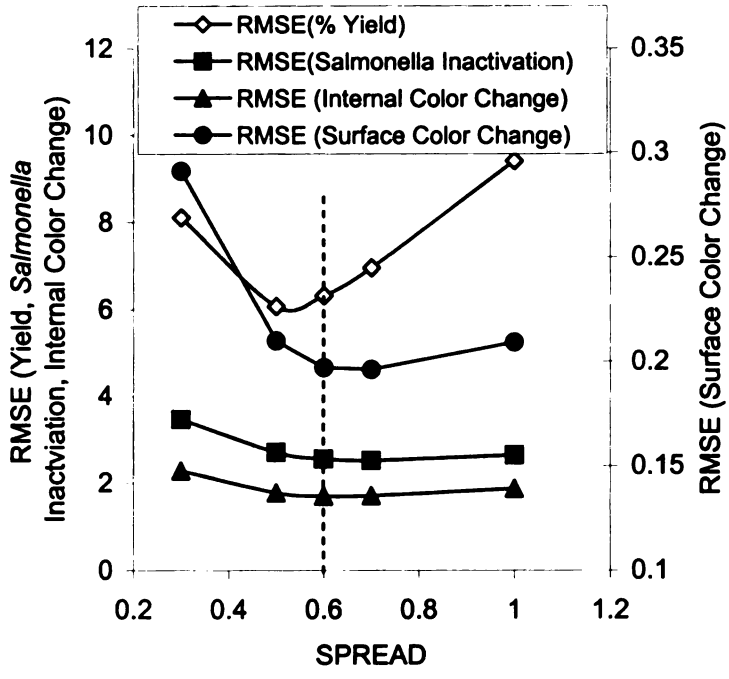


Figure 5.5 RMSE (yield, *Salmonella* inactivation, internal color change, and surface color change) of SNNM_D for different "spread" values.

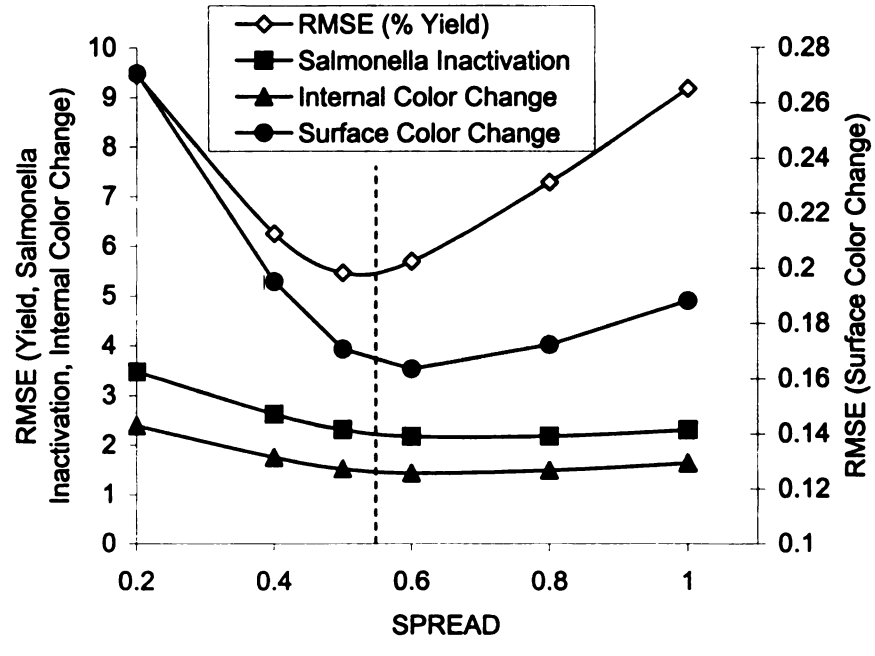


Figure 5.6 RMSE of SNNM_M for different "spread" values: (a) Yield prediction; (b) Constraints prediction.

5.3 Validation of the Trained Neural Networks

5.3.1 The Performance of DNNM

5.3.1.1 DNNM for Random Fourier Process

To increase the accuracy for yield, *Salmonella* inactivation, internal color change, and surface color change prediction, four DNNM's were trained by using GRNN. For validation of the networks, 1,000 conditions were generated randomly by varying the 20 coefficients of the Fourier series for each control vector (temperature, humidity, and impingement velocity) and cooking duration. Thereafter, the 1,000 conditions were applied to the finite element model, and 1,000 resultant data were obtained. The conditions at every 10 seconds and the initial state values were cast to the four individually trained DNNM, and the output vectors were compared with the result of the finite element model. The accuracy of the trained DNNM was calculated in terms of RMSE (Table 5.1).

Table 5.1 The performance of DNNM for a random data group (D2 of Table 4.4), in terms of RMSE.

	Yield [%]	*Salmonella* Inactivation (>0.813)** [log(log(N/N_0))]	Internal Color Change (>0.146)** [log(log(C/C_0))]	Surface Color Change (0.26~0.497) [log(C/C_0)]
RMSE	2.05	0.63	0.55	0.12
Number of Samples	1,000	1,000	1,000	999*

RMSE: Root mean squared error.

* One of the data in the 1,000 samples was eliminated due to overflow.

** log of critical limit for each factor (0.813=log(6.5), 0.146=log(1.4))

Figure 5.7 shows several example of the network prediction qualitatively. In Figure 5.8, the performance of the individual DNNM was represented graphically.

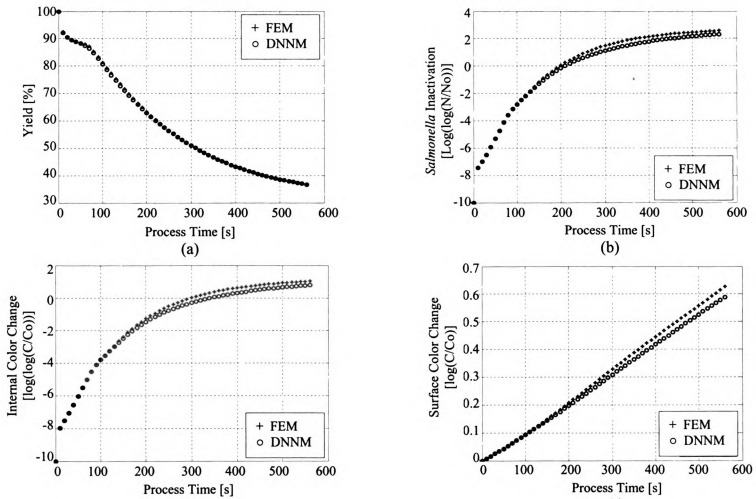
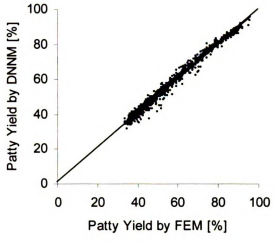
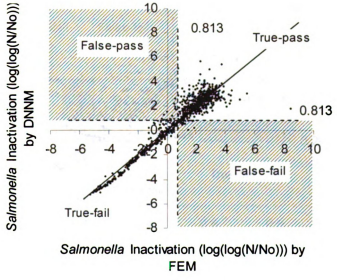


Figure 5.7 Example qualitative performance of the DNNM, with respect to: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

The slope (0.98) of the regression line of yield (Figure 5.8 (a)) prediction was close to 1, and the intercept (1.83) was also small, which means that the model represents the characteristics of the finite element model. Even though some points deviated more than 10% from the 1:1 line, the overall accuracy was fairly good (RMSE = 2.05).

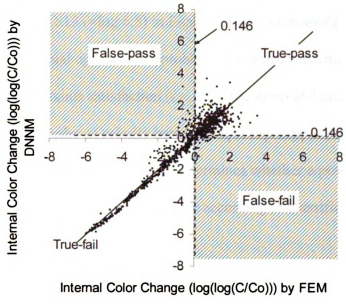


(a)

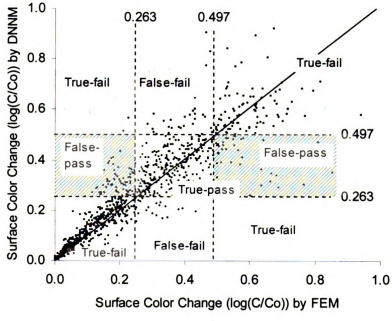


(b)

Figure 5.8 The performance of DNNM for random process as a predictor and classifier. (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change (some data in NE region was not plotted).



(c)



(d)

Figure 5.8 (cont'd).

The *Salmonella* inactivation results were divided into four regions, with a crosshairs at the value 0.813 (=log 6.5) in Figure 5.8 (b). *Salmonella* inactivation works as a constraint in the actual optimization problem. There are two critical aspects of the model prediction: constraints satisfaction (*i.e.,* classification) and goal prediction accuracy. If there is an order of priority between those two aspects, the constraints satisfaction must be placed first, because it determines whether a process "passes". Therefore, the prediction results were analyzed according to classification categories (Table 5.2).

The samples in the true-fail region (Figure 5.8 b) are under-cooked sample, in terms of microorganism inactivation. Again, the samples in the true-pass region passed the critical standard log reduction value. However, some points in the false-fail region were under-estimated samples, which mean safe estimation while sacrificing chance to be explored. In the false-pass region, there were about 1.7% of over-estimated samples out of the 1,000 points, which was an small number of false-passes. However, the false-pass region is the most dangerous one that must be avoided among the classification categories.

Table 5.2 Four possible classification categories of the constraints satisfaction.

	Neural Network Prediction	
FEM Prediction	Pass	Fail
Pass	True-pass	False-fail
Fail	False-pass*	True-fail

* Most dangerous prediction

Plot (c) in Figure 5.8 shows the result of internal color change prediction. With the same rationale as the *Salmonella* inactivation case, samples in the true-pass, true-fail,

and false-fail region are considered as safe classification. Only a small portion (53 points) of the 1,000 data points is located in the false-pass region, which is not desirable.

Data points in Figure 5.8 (d), were divided into 9 regions, because the surface color change constraint has lower (0.26) and upper critical limits of (0.497). The points in the true-fail, true-pass, and true-fail region were classified correctly and safely. However, about 11% of the 1000 points in the two false-pass regions were classified as a dangerous category.

Generally, the trained DNNM showed good prediction and classification performance for the random process, even though the DNNM was trained by using the results of constant processes. The false-pass region is inevitable for a model, but the region must be avoided when the optimization algorithms explore solution space with the model.

Among the 1,000 random data, 170 points that satisfied the safety and quality constraints were sorted out and superimposed on the yield plot (Figure 5.9).

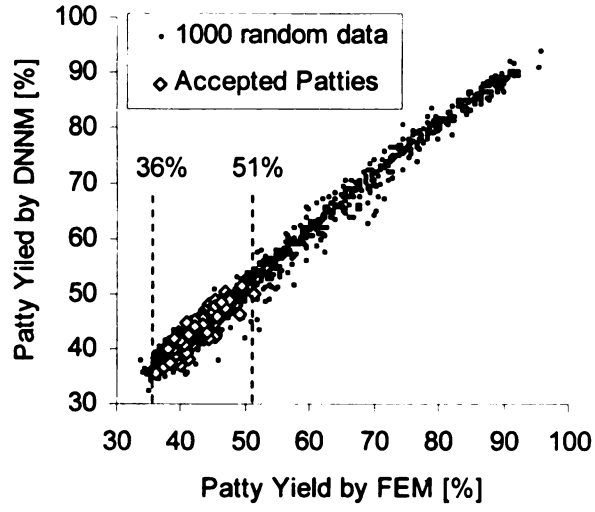


Figure 5.9 Accepted patties (170 patties) were superimposed on the 1,000 random data.

The accepted patties were located between 36% and 51% yield. This fact does not mean that the optimal solution may exist in this range. Rather, the optimal point may be in some region beyond 51%, but with a very low probability. However, GRNN based dynamic neural network model can search the region beyond 51%, because of the nature of the model, which is explained in detail in the Section 4.5.4. Also, the plot implies that optimization algorithms will progress fast in this range (36 - 51%), because of the high density of training data.

The performance of a neural network can be measured by two different criteria, depending on the characteristics of the network. Goal network predicting scalar, such as patty yield, can be estimated via the RMSE. However, constraint networks that determine "Yes" or "No", "Passed" or "Failed", or "True" or "False" (such as safety and quality) can be estimated by using classification rate. In Table 5.3, the classification rates of DNNM as a constraint network were divided in four categories. A critical value is the error of classification for false-pass category. The rate was 0.15 for DNNM network applied to random processes.

Table 5.3 Classification rate of DNNM for random processes.

	Classification Categories				
	Correct		Incorrect		
	True-pass*	True-fail	False-fail	False-pass	Total
NN prediction	170	647	71	112	1,000
(FEM results)	(241)	(759)	(241)	(759)	(1,000)
R_c**	0.71	0.85	0.29	0.15	

* Sample passed all constraints (Salmonella inactivation, internal color change, and surface color change)

** Rate of classification = DNNM prediction / FEM results

5.3.1.2 DNNM for Single-Stage, Double-Stage, and Multi-Zone Process

DNNM is a dynamic neural network model, which can be applied to any type of input patterns. Therefore, the performance of the DNNM must be general and robust. To measure the robustness, the performance of DNNM was measured for single-stage, double-stage, and multi-zone processes.

Randomly produced single-stage, double-stage, and multi-zone data were applied to the DNNM, and RMSE's for yield, *Salmonella* inactivation, internal color, and surface color prediction were calculated for each case (Table 5.4). The RMSE of the single-stage case showed good accuracy for goal prediction (yield, 1.67%) and constraint predictions (0.49 for *Salmonella* inactivation, 0.42 for internal color change, and 0.76 for surface color change). Such accuracy was possible, because the DNNM was trained with single-stage data. It is not strange that the accuracy for the multi-zone case was not much different from the accuracy of the single-stage case, because the multi-zone process has a similar input data pattern to the single-stage, except for a variable velocity profile. Generally, the accuracies of double-stage and multi-zone cases were in the reasonable range, which supports the robustness of the dynamic neural network model.

Table 5.4 The performance of DNNM for single-stage, double-stage, and multi-zone processes in terms of RMSE.

Process Type	Performance	Yield [%]	*Salmonella* Inactivation (>0.813)** [log(log(N/N$_0$))]	Internal Color Change (>0.146)** [log(log(C/C$_0$))]	Surface Color Change (0.26~0.497)** [log(C/C$_0$)]
Single-stage	RMSE (# data)	1.67 (998*)	0.49 (998*)	0.42 (998*)	0.76 (998*)
Double-stage	RMSE (# data)	3.22 (994*)	1.20 (994*)	1.04 (994*)	0.90 (990*)
Multi-zone Process	RMSE (# data)	1.91 (1,000)	0.57 (1,000)	0.50 (1,000)	2.09 (999*)

RMSE: Root mean squared error

* Some data in the 1,000 samples was eliminated due to computation overflow.

** Critical limits

Figure 5.10 shows validation plots of DNNM for the single-stage oven. All the predictions were very close to the 1:1 line. Other validation plots for double-stage (Appendix A: Figure A.1) and multi-zone process (Appendix A: Figure A.2) can be found in the appendix.

The performance of DNNM for various processes was also measured in terms of the rate of classification (Table 5.5). Classification rate for false-pass category for the single-stage case was very low (0.02) as expected. However, false-pass rates for double-stage process were high compared with the other processes. DNNM is a robust model but is not perfectly free from the characteristics of training data sets, which is a single-stage process. The double-stage process deviates more from a single-stage process than does the multi-zone process, and therefore shows the highest false-pass error for DNNM among other process types.

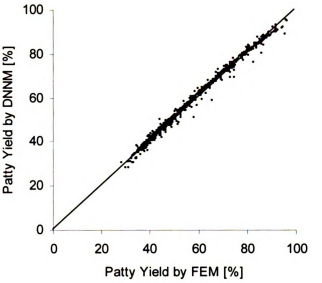
Table 5.5 Classification rate of DNNM for single-stage, double-stage, and multi-zone processes.

Process Type		Classification Categories				
		Correct		Incorrect		
		True-pass*	True-fail	False-fail	False-pass	Total
Single-stage	NN prediction (FEM results)	162 (200)	785 (800)	38 (200)	15 (800)	(1,000)
	R_c**	0.81	0.98	0.19	0.02	
Double-stage	NN prediction (FEM results)	80 (196)	682 (804)	116 (196)	122 (804)	(1,000)
	R_c	0.41	0.85	0.59	0.15	
Multi-zone Process	NN prediction (FEM results)	129 (200)	772 (799)	71 (200)	27 (799)	(999†)
	R_c	0.65	0.97	0.36	0.03	

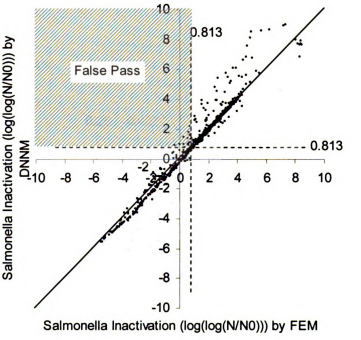
* Sample passed all constraints (*Salmonella* inactivation, internal color change, and surface color change)

** Rate of classification = DNNM prediction / FEM results

† Some data was eliminated due to computation overflow.

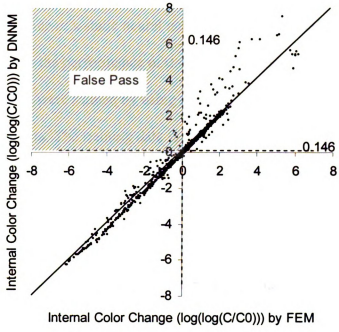


(a)

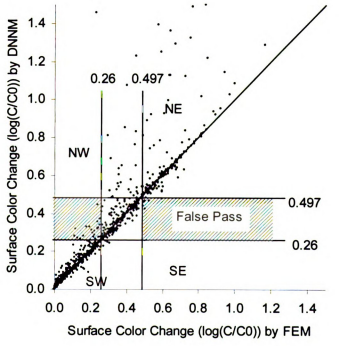


(b)

Figure 5.10 Goal and constraints prediction performance of the DNNM for single-stage process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.



(c)



(d)

Figure 5.10 (Cont'd)

5.3.2 The Performance of SNNM

The performance of a static neural network model is specific to the training data, which is in contrast to the performance of the DNNM trained by a single data group D1. Four different SNNM were trained by using random, single-stage, double-stage, and multi-zone process data. The training data for single, double, and multi-zone process were generated according to Section 4.4.2. Then the feed-forward neural network was used for all predictions (yield, *Salmonella* inactivation, internal color change, and surface color change). However, the feed-forward neural network was not trained well for double and multi-zone processes. Thus, a generalized regression neural network was used for double and multi-zone processes, and improved results were obtained.

The performances of the trained SNNM were validated by using test data groups in terms of RMSE (Table 5.6). The RMSE of yield for random and single-stage processes were 1.78 and 2.52, which were much smaller than the RMSE for double-stage and multi-zone process. For safety and quality constraints, the same was true. In addition, training result of SNNM for the double-stage and multi-zone process, in terms of RMSE and correlation coefficient, were poor (Figure A.4 and Figure A.5 in appendix A).

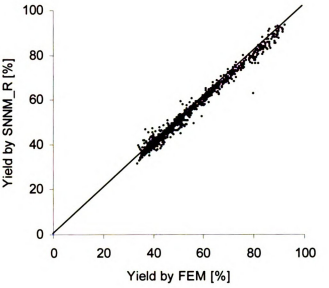
The performance of SNNM for 1,000 random processes was graphically represented in Figure 5.11. Compared to the performance of the DNNM (Table 5.1) for random processes, the SNNM were more accurate for yield, *Salmonella* reduction, and internal color change. Surface color change prediction accuracy was almost identical.

Table 5.6 The performance (RMSE) of SNNM for random, single-stage, double-stage, and multi-zone processes.

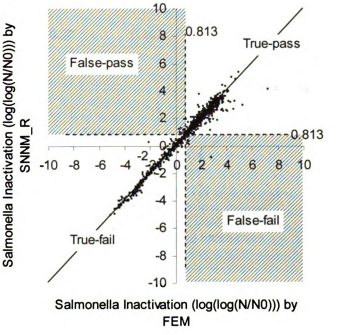
Process Type	Performance	Yield [%]	*Salmonella* Inactivation (>0.813) [log(log(N/N_0))]	Internal Color Change (>0.146) [log(log(C/C_0))]	Surface Color Change (0.26~0.497) [log(C/C_0)]
Random	RMSE	1.78	0.35	0.28	0.09
	N	1,000	1,000	1,000	999*
Single-stage	RMSE	2.52	1.38	0.92	0.15
	N	1,000	1,000	1,000	999*
Double-stage	RMSE	6.32	2.56	1.70	0.20
	N	1,000	1,000	1,000	1,000
Multi-Zone Process	RMSE	5.48	2.22	1.46	0.17
	N	1,000	1,000	1,000	999*

N: Number of data

** Some data in the 1,000 samples were eliminated due to computation overflow.

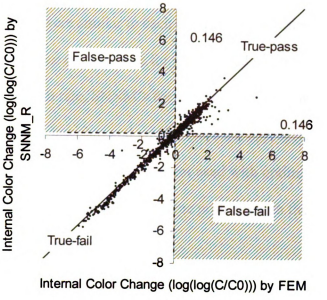


(a)

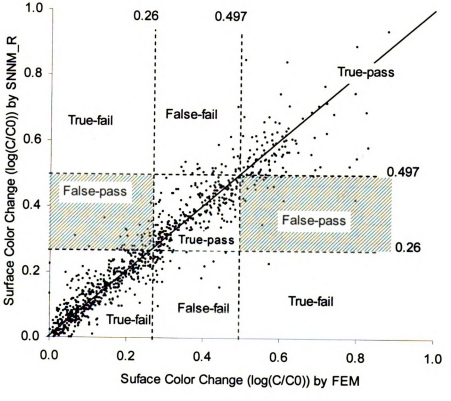


(b)

Figure 5.11 Goal and constraints prediction performance of the SNNM for random Fourier process condition: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.



(c)



(d)

Figure 5.11 (cont'd)

The performance of the SNNM as a classifier was also measured (Table 5.7). Although false-pass rates were similar to each other, false-fail rates of the double-stage and multi-zone process were double those for random and single-stage processes. Based on RMSE and classification rate, SNNM's for double-stage and multi-zone processes were a poor predictor and classifier, and are therefore not appropriate for further utilization. Therefore, those two SNNM were not used with optimization algorithms. The reason of such a poor performance could be some problem with the training data group or inappropriate neural network type.

Table 5.7 Classification rate of SNNM for random, single-stage, double-stage, and multi-zone process.

Process Type		Classification Categories				
		Correct		Incorrect		
		True-pass*	True-fail	False-fail	False-pass	Total
Random Process	NN prediction (FEM results)	190 (243)	702 (757)	53 (243)	55 (757)	(1,000)
	R_c	0.78	0.93	0.22	0.07	
Single-stage	NN prediction (FEM results)	146 (199)	739 (801)	53 (199)	62 (801)	(1,000)
	R_c	0.73	0.92	0.27	0.08	
Double-stage	NN prediction (FEM results)	90 (197)	729 (797)	107 (197)	70 (797)	(996†)
	R_c	0.46	0.91	0.54	0.09	
Multi-zone Process	NN prediction (FEM results)	99 (203)	733 (796)	104 (203)	63 (796)	(999†)
	R_c	0.49	0.92	0.51	0.08	

* Sample passed all constraints (*Salmonella* inactivation, internal color change, and surface color change)

R_c : Classification rate = NN prediction / FEM results.

† Some data was eliminated due to computation overflow.

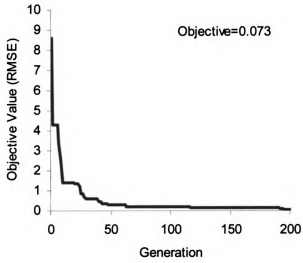
5.4 Benchmark Test for Optimization Algorithms

A Bezier curve fitting problem (Section 4.6.5) was solved with three optimization algorithms (GA, SA, and ICRS), and the results were compared (Table 5.8). Ten simulations were executed for each algorithm, and the average and variance were calculated. The lowest average RMSE was achieved by the ICRS algorithm. The three average RMSE were statistically compared, and the averages were not significantly different from each other (F-value=1.42; p-value=0.26). In other words, their performances were almost identical.

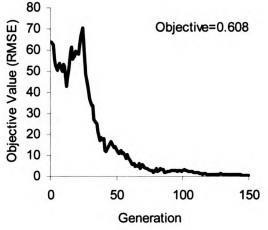
Table 5.8 Comparison of the convergence of three different optimization algorithms in terms of RMSE for the Bezier parametric curve fitting problem.

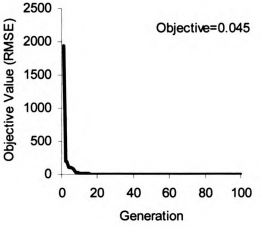
		RMSE	
Algorithms	Simulations	Average	Variance
GA	10	0.4804	0.1334
SA	10	0.2871	0.0586
ICRS	10	0.2580	0.1167

GA showed a gradual and moderate rate of convergence through out the total 200 generations (Figure 5.12 (a)). SA showed lots of exploration at the early stages but not many in the later iterations (Figure 5.12 (b)). The ICRS algorithm converged before the 20^{th} iteration (Figure 5.12 (c)). However, this does not mean that the ICRS is the fastest algorithm, because the execution time for one generation is heuristic, which is different from the other algorithms.



(a)





(b) (c)

Figure 5.12 Examples of objective function value (RMSE) for every generation of three different optimization algorithms in the Bezier parametric curve fitting problem. (a) GA; (b) SA; (c) ICRS.

Generally, all computer codes for the optimization algorithms successfully optimized the Bezier curve-fitting problem. However, the speed of the algorithms could not be compared, because the total execution time depends on the values of the algorithm parameters.

5.5 Optimal Conditions from the Various Strategies

5.5.1 Summary of the Various Optimization Strategies

To understand comprehensive characteristics of specific optimization strategies for meat patty cooking under moist air impingement conditions, three models, two control profile parameterization methods, and three optimization algorithms were combined (Table 5.9) and tested. However, the SNNM-PLI combinations (strategy #17, #18, and #19) were not tested, because of problem previously described (Section 4.6.1)

The highest maximum yield (73%) was achieved with the ICRS-FEM-PLI strategy (#3), and the second highest yield (70%) was achieved with SA-FEM-FS (strategy #6). However, neural network model based optimization strategies showed constraints satisfaction problems, meeting safety and quality constraints. This is an expected classification error, because those alternative models were trained using FEM results, which was considered as an experiment in this research. However, even though the FEM-based optimization strategies satisfied all the constraints, those might actually not be free from the constraint satisfaction problem, if the results were validated with actual field experiments.

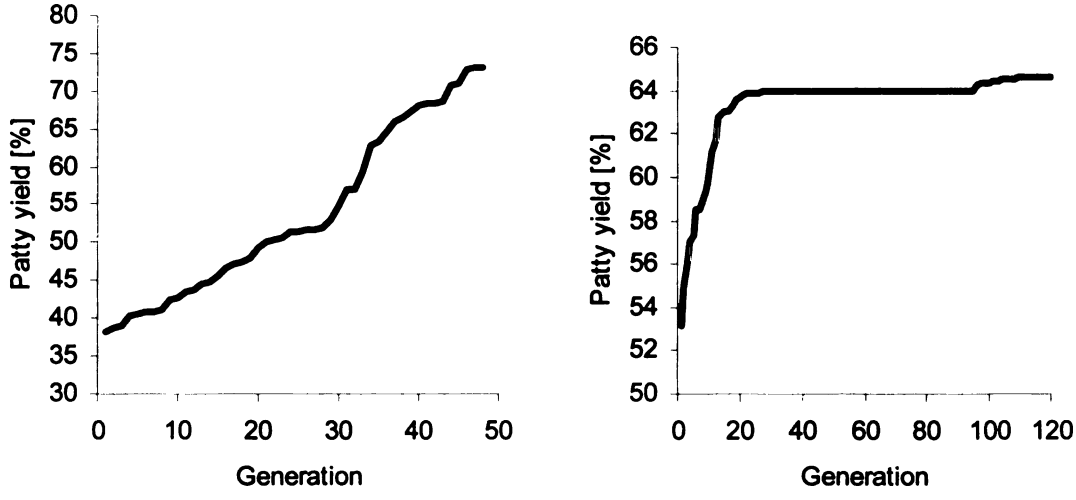
Table 5.9 Brief results of the various optimization strategies.

Model	CP[1]	Algorithm	Strategy	Max. Yield [%]	Constraints Satisfaction	Convergence and Optimal Profiles (Appendix A)
FEM	PLI	GA	1	56	S	Figure A.6 Figure A.7
		SA	2	64	S	Figure 5.15 Figure A.8
		ICRS	3	73	S	Figure 5.13 Figure 5.14
	FS	GA	5	60	S	Figure A.6 Figure A.7
		SA	6	70	S	Figure 5.15 Figure A.8
		ICRS	7	65	S	Figure 5.13 Figure 5.14
DNNM	PLI	GA	9	57	US	Figure A.9 Figure A.10
		SA	10	52	US	Figure A.11 Figure A.12
		ICRS	11	60	US	Figure A.13 Figure A.14
	FS	GA	13	69	US	Figure A.9 Figure A.10
		SA	14	49	S	Figure A.11 Figure A.12
		ICRS	15	55	US	Figure A.13 Figure A.14
SNNM_R	PLI	GA	17	N/A	N/A	N/A
		SA	18	N/A	N/A	N/A
		ICRS	19	N/A	N/A	N/A
	FS	GA	21	51	S	Figure A.15
		SA	22	53	US	Figure A.16
		ICRS	23	54	S	Figure A.17

[1] Control profile parameterization method

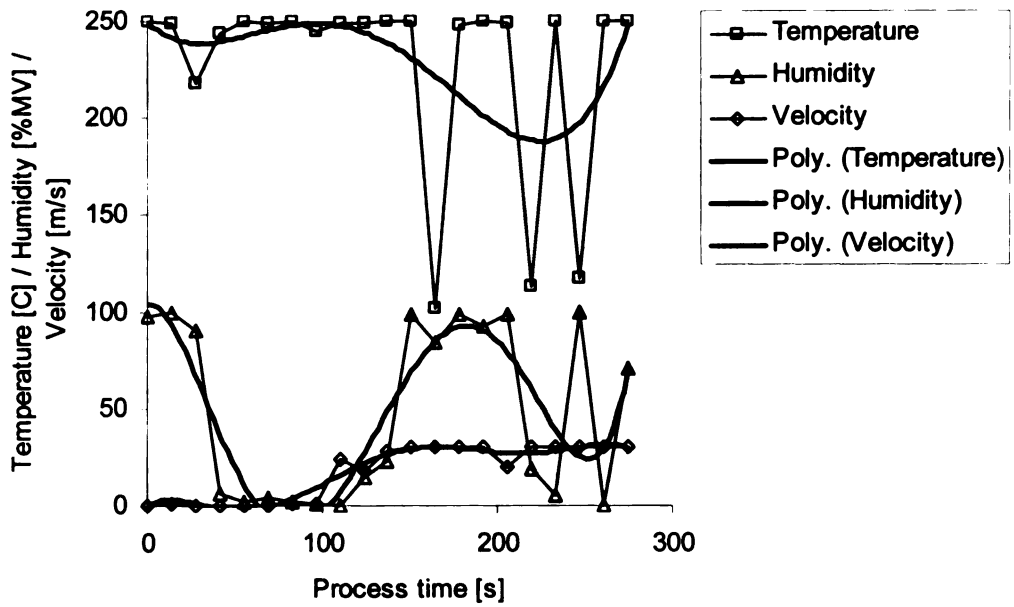
S: Satisfied; US: Unsatisfied

Generally speaking, FEM based approaches were the slowest approaches, and DNNM and SNNM-based approaches were much faster, while sacrificing accuracy. All the different strategies converged at the end of the optimization process (Figure 5.13, strategy #3). However, all the converged optimal control profiles were not consistent with each other, which implies that the solution domain might be highly multi-modal and constrained. In addition, the total number of control parameters was 64 in this study, which made it somewhat difficult to achieve uniform and consistent optimal yields and profiles. Even so, some strategies showed meaningful optimal control profiles, suggesting stepwise control policies in Figure 5.14 and Figure 5.15 . Polynomial regression curves were added to the PLI profiles to illustrate general trends in process conditions. Convergence and optimal control profiles of the other strategies can be found in Appendix A.

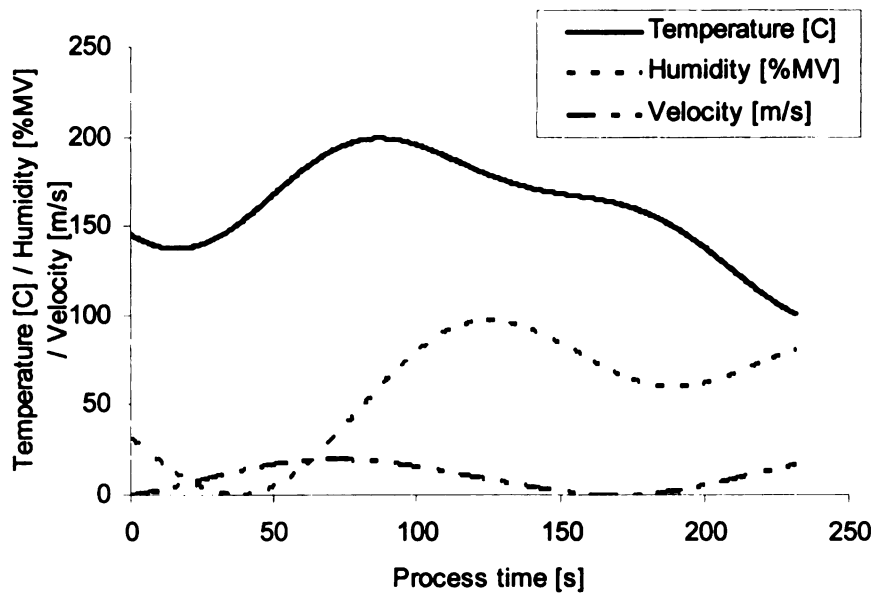


(a) (b)

Figure 5.13 ICRS-FEM optimization process and convergence (strategy #3): (a) PLI parameterization; (b) FS parameterization.

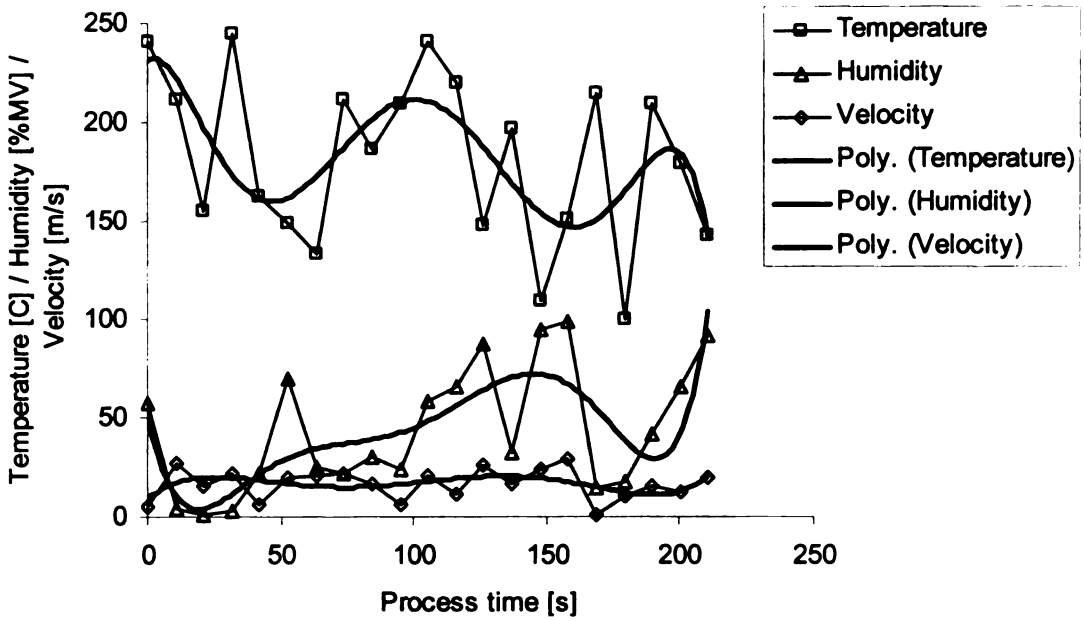


(a)

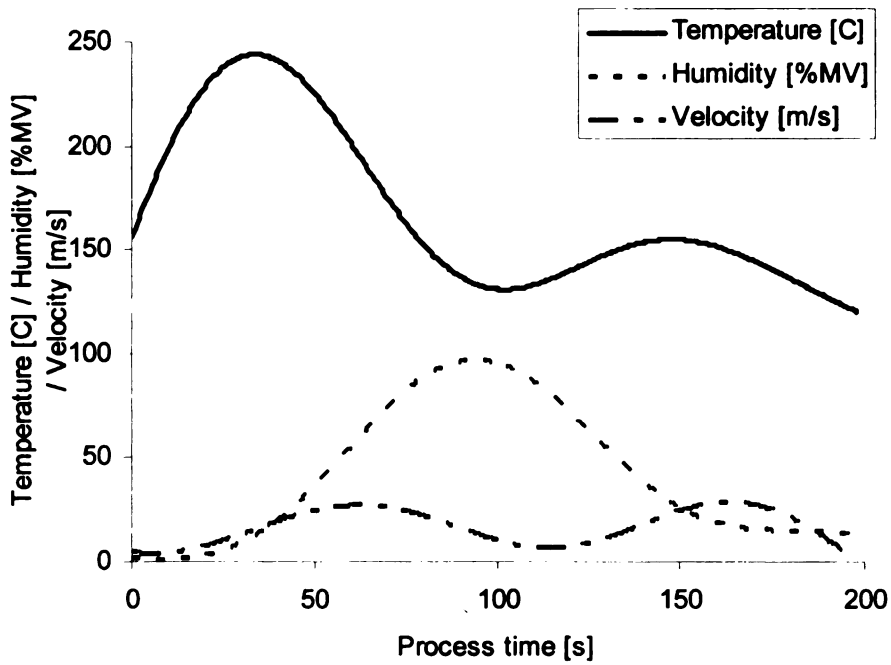


(b)

Figure 5.14 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by ICRS-FEM optimization strategy (strategy #3): (a) PLI parameterization; (b) FS parameterization.



(a)



(b)

Figure 5.15 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by SA-FEM optimization strategy (strategy #6): (a) PLI parameterization; (b) FS parameterization.

5.5.2 *Optimization Performance of the Alternative Models Coupled with Algorithms*

In this research, FEM was used as a replacement for actual experiments. Therefore, the results of DNNM and SNNM were compared with FEM results to test the performance of alternative models when those models were coupled with various optimization algorithms.

To measure the model performance via speed, the execution time of the model itself was measured by using the MATLAB profiler function and constant control profiles (600 seconds process duration). FEM, DNNM, and SNNM took 140, 16, and 0.22 seconds, respectively. Compared with FEM, therefore, DNNM and SNNM showed huge benefit in total execution time of algorithms, due to the iterative nature of the algorithms.

In addition to the execution speed, the model performance was measured via prediction accuracy, in terms of constraints satisfaction. Table 5.10 showed that most of the cases committed false-pass classification errors, except three cases (strategy #14, #21, and #23).

Table 5.10 Predicted goal and constraint values of DNNM and SNNM, compared with the result of FEM.

		Objective			Constraints						
	Strategy (Table 5.9)	Yield [%]			*Salmonella* log reduction (>0.813) [log(log(N/N$_0$))]		Internal color change (>0.146) [log(log(C/C$_0$))]		Surface color change (0.26~0.497) [log(C/C$_0$)]		S
Model		A	F	E [%]	A	F	A	F	A	F	
DNNM	9	57	44	29	1.44	4.01	0.15	2.19	0.27	>100*	
	10	52	48	9	1.48	1.76	0.15	0.29	0.26	0.17*	
	11	60	57	5	1.47	3.09	0.15	1.43	0.45	0.22*	
	13	69	65	6	1.71	-0.93*	0.15	-2.14*	0.39	0.15*	
	14	49	44	13	1.61	3.33	0.15	1.66	0.27	0.37	P
	15	55	50	10	1.7	1.41	0.19	-0.04*	0.30	0.27	
SNNM	21	51	47	10	1.64	2.37	0.15	0.80	0.26	0.32	P
	22	53	51	4	1.64	1.87	0.15	0.37	0.26	0.21*	
	23	54	48	13	1.65	2.78	0.15	1.14	0.26	0.41	P

A: alternative model prediction; F: finite element model result; S: constraints satisfaction status; P: passed all the three constraints; E (Prediction error) =100× |A-F| / F [%]; *: Unsatisfied constraint

Also, the prediction error for yield of those successful cases were 10 to 13%. Another interesting point was that there were more false-pass classification errors in surface color change prediction than the other two constraints, indicating that the surface color change is reacting sensitively to the control profiles.

5.5.3 Efficiencies of Various Strategies

Efficiency of an optimization can be measured by dividing total goal achievement by total execution time (Equation [5.1]).

$$\varepsilon_A = |Y_{\max} - Y_o| / t_{exe} \qquad \textbf{[5.1]}$$

However, comparing efficiencies of multiple strategies based on absolute fair conditions seemed impossible, because of the various parameters in the algorithms and a heuristic search algorithm. Although absolute comparison was difficult, total execution time and achieved maximum yield are provided in Table 5.11.

To measure combined performance of each strategy, averaged execution time per unit iteration was defined, which was calculated at the point where a strategy converged to 99% of the final converged value. Generally, the efficiencies of DNNM and SNNM were higher than the efficiency of FEM, due to fast convergence. However, the efficiency does not account for accuracy. The efficiencies in Table 5.11 do not provide an absolute fair comparison, but provide a rough reference. Even though a fair comparison was impossible, the performance of the optimization algorithms thereby can be viewed qualitatively.

Table 5.11 Total execution time and the achieved maximum yield for various optimization strategies.

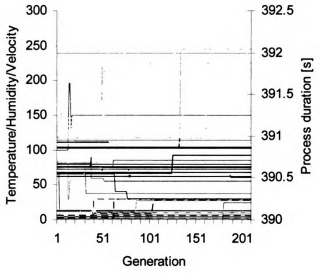
Model	CP	Algorithm	Strategy	$N_{G,99\%}$	t_G	t_{exe}	Yo [%]	Ymax [%]	D [%]	ε_A [% yield/s]
FEM	PLI	GA	1	86	1.33	114.38	48	56	8	0.07
		SA	2	95	10.94	1039.30	40	64	23	0.02
		ICRS	3	46	6.72	309.12	38	73	34	0.11
	FS	GA	5	165	0.58	95.70	53	60	6	0.07
		SA	6	127	4.9	622.30	40	70	29	0.05
		ICRS	7	96	6.03	578.88	53	65	11	0.02
DNNM	PLI	GA	9	554	0.08	44.32	52	57	5	0.11
		SA	10	130	0.48	62.40	43	52	9	0.14
		ICRS	11	231	0.23	53.13	48	60	12	0.22
	FS	GA	13	215	0.05	10.75	51	69	17	1.60
		SA	14	147	0.12	17.64	38	49	11	0.62
		ICRS	15	7	0.08	0.56	38	55	16	28.91
SNNM	FS	GA	21	23	0.0039	0.09	50	51	1	11.59
		SA	22	N/A	0.00044	N/A	N/A	N/A	N/A	N/A
		ICRS	23	26	0.0034	0.09	48	54	5	59.39

CP: Control profile parameterization; $N_{G,99\%}$: Iteration number achieving 99% of the final objective value; t_G: Averaged execution time per iteration; t_{exe}: total execution time; Yo: Initial objective value; Ymax: Final objective value; D: |Yo-Ymax|; ε_A: Optimization efficiency [% yield/s]

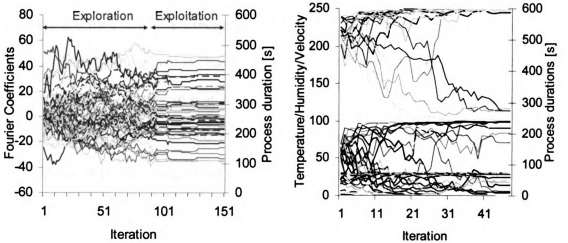
Strategy #1~7 were run on Sun Fire v880 (4 UltraSPARC-III 750MHz 64 bits CPU), and 9~23 were run on PC (3.2 GHz, Intel Pentium 4 hyper threaded). Thus, to compare with FEM results, time must be multiplied by about 4.5.

Global optimization has exploration and exploitation features in its algorithm. Figure 5.16 shows how the control variables converge throughout the whole optimization process in GA, SA, and ICRS. In GA (Figure 5.16, (a)), there is no distinct point dividing exploration and exploitation mode, because the algorithm has the two features in every generation. However, in SA, there is a distinct section of exploration, which is over 50% of the entire optimization process, which indicates the algorithm was effective for searching many possibilities at the early stages of optimization. Following exploration, the exploitation process gradually starts as the process goes to the later stage of optimization. However, the exhaustive exploration at the early stage could be negative to the efficiency of SA algorithm, especially when the process model is slow. Compared

with SA, ICRS showed no distinctive section of exploration and exploitation; rather, it was gradual. Therefore, ICRS could avoid exhaustive searching at the early stage by transitioning smoothly from exploration to exploitation within a small number of iteration.



(a)



(b) (c)

Figure 5.16 Convergence history of each control parameters shows exploration and exploitation features of the optimization algorithms: (a) GA-FEM-PLI (strategy #1); (b) SA-FEM-FS (strategy #6); (c) ICRS-FEM-PLI (strategy #3).

5.5.4 Constraints Satisfaction Problem

To analyze the search pattern of a DNNM or SNNM based optimization strategy that is committing false-pass classification errors frequently, the search pathways were validated with FEM results. A typical example is presented in the Figure 5.17. The performance index or goal (yield) moved along the 1:1 prediction line, even though the accuracy was not high (Figure 5.17 (a)). However, the search algorithm traveled to the false-pass region at the later stage of optimization for all three constraints. This strange behavior was likely due to erroneous moves that had small slope in the validation plot. However, if the pathways of the constraints are guided to follow the 1:1 line, the possibility of the erroneous move can be reduced.

Therefore, a directional constraints satisfaction (DCS) algorithm was developed and implemented in the ICRS algorithm. The DCS algorithm guides the progress of constraints in one direction, which is determined at the starting point, until no progress is observed, and then turns the direction into a direction of more progress. This alternate process is kept until the goal is converged. Figure 5.18 shows an example of a DCS implemented ICRS algorithm search behavior. In the figures, all three constraints moved along the 1:1 line without deviatory moves, even though a constraint fell into a false-pass region. Another positive side effect is that the pathway of the goal was also straight, which means fast convergence. However, the goal achieved was relatively lower than the algorithm without DCS. This drawback might be possible, because the search algorithm was initially constrained in one direction while sacrificing explorations chances.

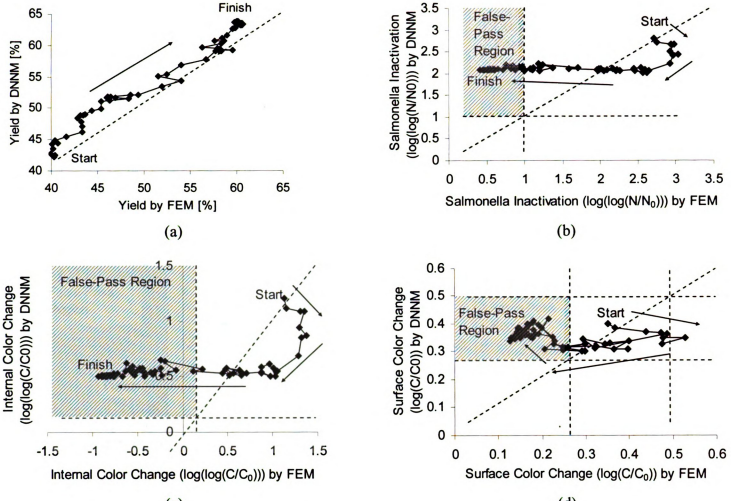


Figure 5.17 A typical search path committing false-pass classification errors (ICRS-DNNM-FS): (a) Yield; (b) *Salmonella* reduction; (c) Internal color change; (d) Surface color change.

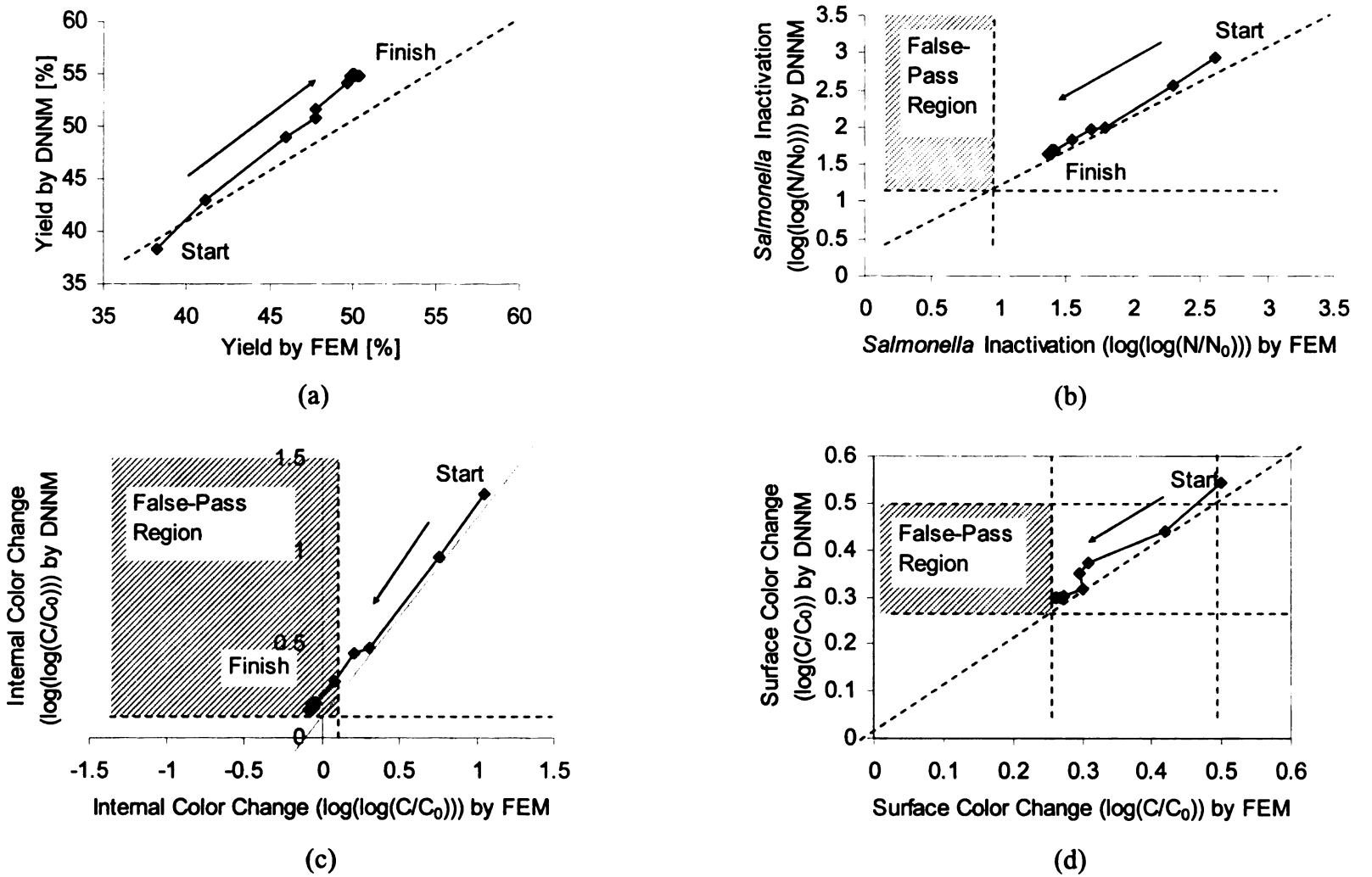


Figure 5.18 Directional constraint satisfaction (DCS) algorithm prevents search from drifting toward false-pass region (ICRS-DNNM-FS): (a) Yield; (b) *Salmonella* reduction; (c) Internal color change; (d) Surface color change.

Another strategy to avoid false-pass classification errors is using under-estimated conditions as a starting point of the optimization algorithm. For strategy #23, a highly under-estimated point was intentionally used as a starting point, which was successful in avoiding false-pass classification error throughout the several simulations.

5.5.5 Sensitivity of the Optimal Control Profiles

Monte Carlo simulation was applied to the optimal solutions to check the sensitivity of the answer by using Equation [5.2]:

$$\tilde{p}_i = p_i + \sigma \cdot \Gamma \qquad \textbf{[5.2]}$$

By using control parameter (p_i) as an average, a normal distribution (Γ), and a practical standard deviation (σ), a perturbed control parameter ($\tilde{p}_i$) was generated for 64 control parameters. The standard deviations were set as follows: 1 for PLI parameters, which means actual perturbation of ±1 (°C, %MV, m/s, and s) and 0.01 for FS parameters, which perturbs control profiles with almost the same magnitude of the PLI case, except process duration (=1). A total of 100 simulations were executed for the optimal results of strategy #3 and #6, because these were the first and the second best results, in terms of patty yield.

For strategy #3 (FEM-PLI-ICRS), 37 simulations showed computational overflow, and the other 63 simulations gave proper predictions (Table 5.12). However, all the 64 simulations were rejected due to the constraints satisfaction test. This is a reasonable result, because the search algorithm locates a final solution with very small stepsize moves in all directions, which result in a solution very close to the constraint boundaries. If this high sensitivity of the optimal solution is also true to actual experiments, the

adoption of the optimal solution must be careful, because of the variance of the actual control profiles. Therefore, a sub-optimal solution of strategy #3 was checked for sensitivity. The sub-optimal solution was arbitrarily chosen at 45th iteration (48 iteration total). Even though the simulation showed almost the same number of computational overflows, the risk of failure was lowered from 100% to 85% (Table 5.12), and the expected average yield was 64.51±2.02 (95% confidence) among 62 non-failed simulations, compared with 71% for the optimum. The Monte Carlo simulation was also applied to strategy #6, with standard deviation 0.01 for Fourier coefficients and 1 for process duration. This case showed less computational overflows, much lower risk of failure (49%), and higher expected average yield of 68.36±0.62 (95% confidence) than the case of strategy #3. Therefore, Fourier parameterization can be considered to have more advantages in real application of the optimal solution in terms of risk of failure.

Table 5.12 Sensitivity of the optimal solutions for FEM_ICRS_PLI and FEM_SA_FS was obtained by using Monte Carlo simulation.

Strategy	Maximum Yield [%]	Number of Failures N_f	Total Number of Samples N_t**	Risk of Failure (N_f/ N_t) [%]	Yield Confidence Interval for N_t (95% confidence)
#3	73	63	63	100	N/A
	71	53	62	85	64.51±2.02*
#6	70	39	79	49	68.36±0.62

* Among nine true-pass cases, an outlier (20% yield) was excluded

** Among 100 simulations, some simulations failed to predict results due to computational over flow.

The simulation results of strategy #6 were represented by using histograms (Figure 5.19) to see more aspects of the simulations. Figure 5.19 (a) showed that a point of 10% deviation from the optimal solution was rare but possible. In contrast, there was also a possibility to have a little more improved result. The constraints for *Salmonella* reduction and internal color change had a small possibility of rejection. However, the surface color change constraint showed more vulnerability to rejection.

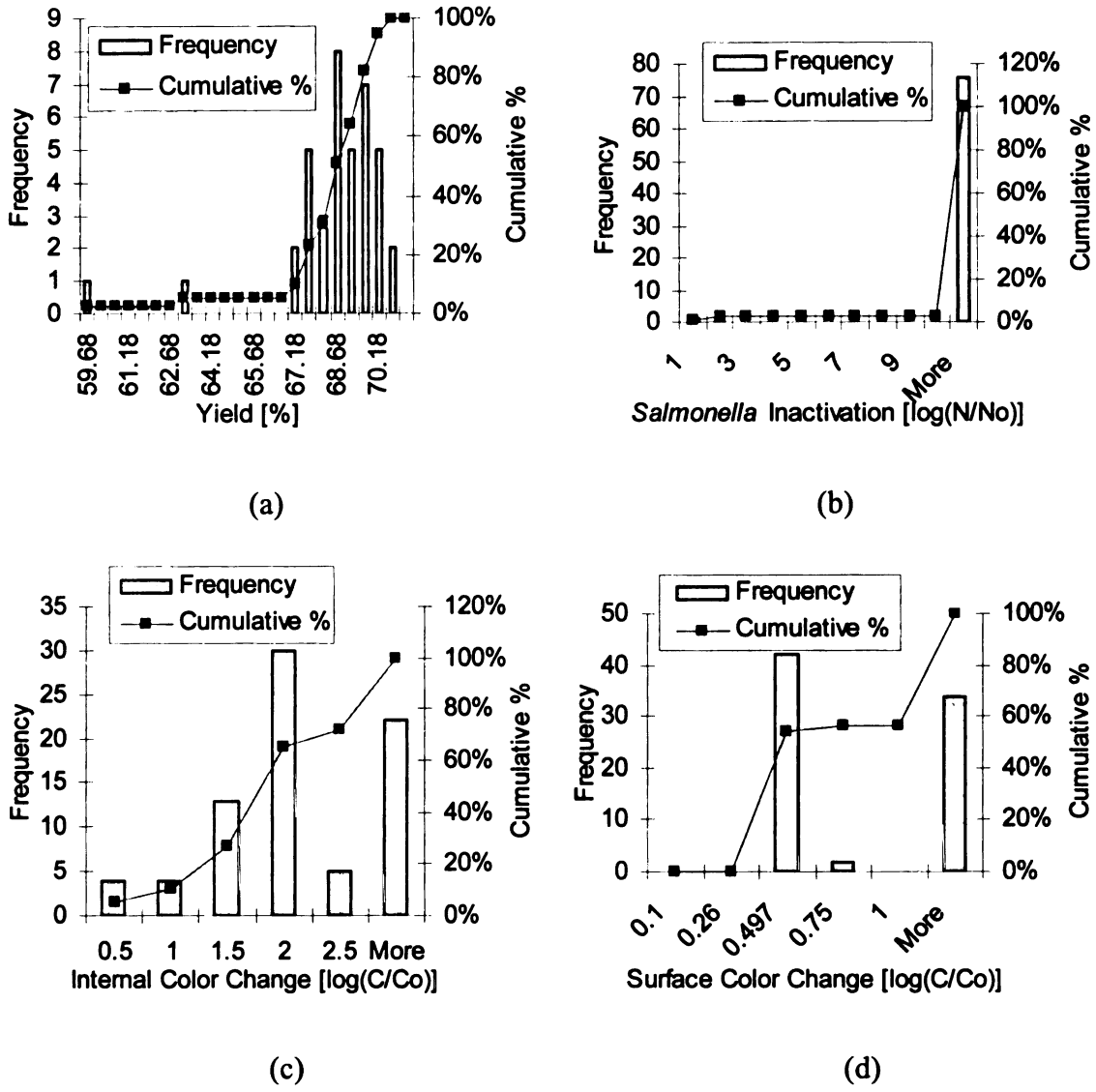


Figure 5.19 Histograms of objectives and constraints were plotted from the population of Monte Carlo simulation for the strategy #6: (a) Yield distribution of passed simulations; (b) *Salmonella* inactivation distribution; (c) Internal color change distribution; (d) Surface color change distribution.

5.6 Case Studies

5.6.1 Single-Stage Process

The ICRS algorithm was applied to find optimal constant profiles for a single-stage oven process. Two independent trials, having different starting conditions, were made for strategy #25, and the results were summarized in Table 5.13.

Table 5.13 The results of the single-stage optimization process by using ICRS-FEM.

	Objective	Constraints			Control Variables			
Trial	Maximum Yield [%]	*Salmonella* Inactivation (>6.5) [log(N/N$_0$)]	Internal Color Change (>1.4) [log(C/C$_0$)]	Surface Color Change (0.26~0.497) (0.11~0.87)# [log(C/C$_0$)]	T [°C]	H [%MV]	V [m/s]	Pt [s]
1	68*	53.24*	1.54*	0.39*	102	97	28	212
1**	57	959.01	21.08	0.27	104	95	28	306
2	57	49.14	1.40	0.31	248	10†	29	222
3#	57	49.14	1.40	0.31#	248	10	29	222
4#	67	47.58	1.40	0.14#	100	89	29	216
5‡	67	47.71	1.41	0.26	101	87	10	230

* The values are meaningless due to computational failure.
** 17th iteration of trial 1 (21 iteration total).
† Lower bound was 10% MV that is achievable in practical application.
# Surface color change constraint was relaxed. (0.11~0.87).
‡ A reduced *D*-value (=233 s which is the ½ of the original *D*-value) was used for surface color change kinetics.

At the first trial, 68% yield was achieved with low temperature (close to 100 °C) and high humidity (close to 100 %MV) (LTHH) condition. However, the algorithm took a wrong path at the 17th iteration and showed abrupt improvement of the yield, because of failure of the predictive model at certain conditions (Figure 5.20). Even so, the optimization process already showed a converging trend at the 13th iteration, and the result of the 17th iteration was considered as the final converged answer. Therefore, the maximum yield was 57% at the 17th iteration, with low temperature and high humidity

condition, which could be a generally accepted operating policy in commercial application. In addition, the optimal profiles at the17th and the 21st iteration were very close to each other, except for total process duration.

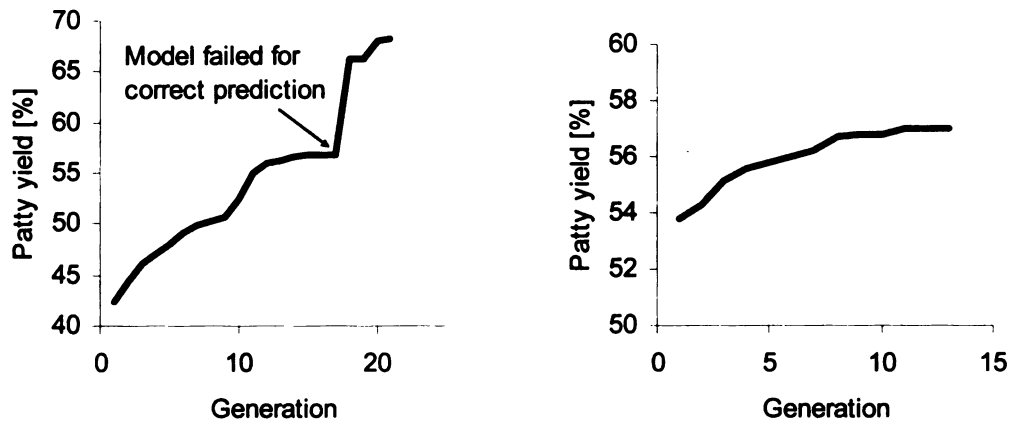


Figure 5.20 Optimization processes for single-stage oven: (a) Process for low temperature and high humidity optimal profile; (b) Process for high temperature and low humidity profile.

The second trial (Table 5.13) showed 57% yield with high temperature (close to 250 °C) and low humidity (≈10 %MV), which is opposite to the conditions of the 1st trial. The HTLH (high temperature and low humidity) condition processed the patty minimally in terms of *Salmonella* reduction and internal color change, which was the advantage, compared with the case of the LTHH. In addition, the total process time is also shorter than the duration of the LTHH.

Therefore, those results suggest that the HTLH condition is more favorable for single-stage cooking, according to the facts found in this optimization. However, this is opposite to the conventional operating concept in industry, which favors LTHH (or HTHH) to achieve high yield. The surface color change factor incorporated into process model might be the cause of this opposite result, because HTLH condition is advantageous to surface browning.

In addition, the effects of a relaxed constraint and a kinetic parameter change were studied via trial 3, 4, and 5 in Table 5.13. Trial 3 and 4 were conducted with relaxed surface color constraints (0.11~0.87), which showed two extreme cases, HTLH and LTHH. These patterns were similar with the patterns of trial 1 and 2 in Table 5.13, except improved yield (66%) and decreased process time (216 s), which were possible because the surface color change could go beyond the previous lower bound (0.26) and resulted in 0.14. However, the results of trial 3 were identical with trial 2, which showed that the increased range of surface color constraint had no effect on this HTLH pattern. For trial 5, the *D*-value of surface color change was decreased by half of the original value (7.75 min), which increased reaction rate. LTHH profile was found as an optimal solution, because the decreased *D*-value was close to the *D*-value of *Salmonella* inactivation.

Neural network model based optimization strategies were also tested, and the results were summarized in **Table 5.14**. The yield prediction accuracies of SNNM were better the DNNM, even though the model performance of DNNM was better than SNNM. The third trial (#27-3 in **Table 5.14**) was very accurate (2% error) and passed all the constraints.

Table 5.14 The accuracy of ICRS-DNNM (strategy #26) and ICRS-SNNM (strategy #27) were validated by using FEM results.

		Objective			Constraints										
		Yield [%]			Salmonella Inactivation (>0.813) [$\log(\log(N/N_0))$]		Internal Color Change (>0.146) [$\log(\log(C/C_0))$]		Surface Color Change (0.26~0.497) [$\log(C/C_0)$]			Control Variables			
Model	Strategy	A[1]	F[2]	Error [%]	A	F	A	F	A	F	Constraints Satisfaction	Temperature [°C]	Humidity [%MV]	Velocity [m/s]	Process Time [s]
DNNM	26-1	50	42	19	1.48	2.75	0.15	1.14	0.26	0.28	Pass	244	73	23	391
	26-2	60	57	5	1.49	3.13	0.15	1.45	0.48	0.25	Fail	105	99	9	328
	26-3	55	44	25	1.48	2.92	0.15	1.26	0.49	0.48	Pass	247	18	24	346
SNNM	27-1*	55	49	12	1.97	2.26	0.34	0.71	0.26	0.31	Pass	250	60	1	416
	27-2*	54	55	1	1.77	1.65	0.27	0.16	0.26	0.24	Fail	100	54	7	326
	27-3	55	54	2	2.42	2.79	0.95	1.17	0.26	0.26	Pass	101	76	1	465

[1] Alternative model (neural network)
[2] Finite element model
* DSC (directional satisfaction of constraints) algorithm was used.

The optimal control profiles found by DNNM and SNNM showed similar tendency with FEM based optimization. Temperature profile patterns were polarized into upper and lower limits, while humidity profile patterns were in the mid range between the upper and lower limits. To avoid frequent false-pass classification errors, the DCS algorithm was applied to the two trials of SNNM based optimization (#27-1 and #27-2). Even though one of the trials failed to satisfy surface color constraints, the predicted value (0.26) was close to the result (0.24) of FEM.

The optimal solutions for the single-stage oven were compared with the previous actual experiment (Watkins, 2004). The actual experiment was for the single stage moist-air impingement oven, and the conditions consisted of temperature (121~232 °C), humidity (50~88 %MV), impingement velocity (11.4~21.8 m/s), and cooking duration (180~660 s). Among 27 different conditions, 19 conditions were verified as fully-cooked[24] cases, and the yield was ranged from 58% to 73%. Considering the differences in the constitutions of the sample patties (65.7 % water and 10% fat), patty dimensions, the process conditions, and the number of quality constraints, the optimal solutions of this study can be thought to be in a reasonable range.

[24] According to FSIS, required lethalities (6.5-log10 or 7-log10 reduction of *Salmonella*) are achieved instantly when the internal temperature of a cooked meat product reaches 71.1°C or above.

5.6.2 Double-Stage Process

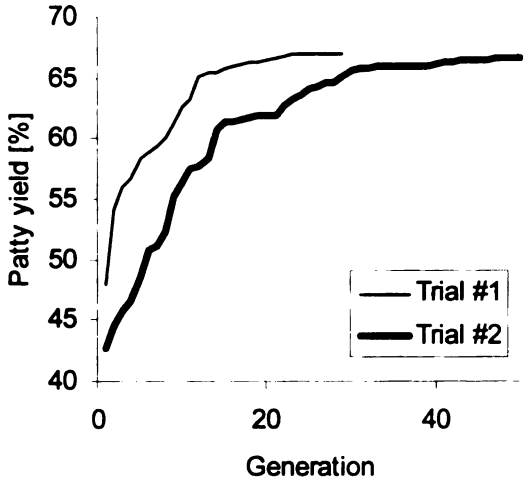
The double-stage process is a simple and practical solution to give dynamics to the process, via serial arrangement of two single-stage ovens. Two trials with ICRS-FEM strategy (strategy #29) were conducted, and the results are summarized in the Table 5.15. The two trials showed consistent results in every aspect. The yields achieved by double-stage process were 67%, which were about 10% point higher than the yields of single-stage process. The advantage is not just improved yield, but also the shorter process duration, which is almost half that of the single-stage process. This is a very encouraging result for both food processors and oven manufacturers. For food processors, high yield and short process time increase profits and productivity, which can justify the capital cost of two ovens. In addition, the optimal process satisfies multiple safety and quality constraints within a single process.

In the previous section, the results from strategies #3 and #7 already suggested that a 2- or 3-step process might be optimal. Thus, that indication was further supported by the results of the double-stage process optimization. In Figure 5.21, temperature and velocity profiles were not significantly different in each step. However, humidity showed two significantly different conditions, dry process (≈0 %MV) and wet process (≈100 %MV). This implies that humidity plays major role in the double-stage process. Also, the pattern of humidity condition (low to high humidity) was opposite to industrial convention (high to low humidity). This interesting result might be caused by the surface color change factor in the integrated FEM.

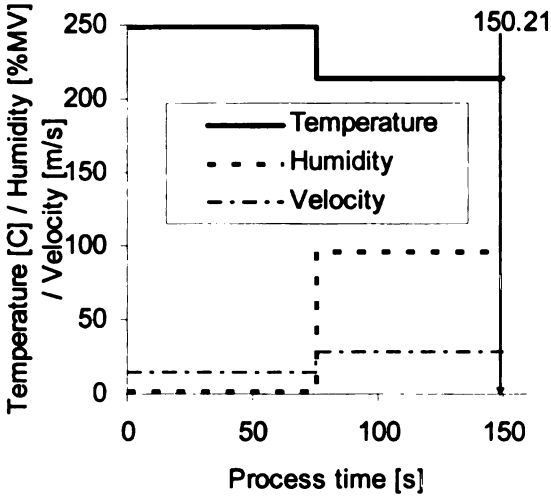
Table 5.15 Maximum yield and optimal control variables were found by using ICRS-FEM for double-stage process.

			Goal	Constraints			Control Variables						
							Temperature		Humidity		Velocity		Process Time
Model	Strategy	Trials	Yield [%]	Salmonella Inactivation (>6.5) [log(N/N$_0$)]	Internal Color Change (>1.4) [log(C/C$_0$)]	Surface Color Change (0.26~0.497) (0.11~0.87)* [log(C/C$_0$)]	T1 [°C]	T2 [°C]	H1 [%MV]	H2 [%MV]	V1 [m/s]	V2 [m/s]	Pt [s]
FEM	29	1	67	56.87	1.40	0.49	248	214	2	96	14	28	150
		2	67	55.93	1.40	0.41	237	227	4	87	28	27	156
		3*	67	56.09	1.40	0.42	243	221	4	96	29	27	155
		4*	66	47.13	1.40	0.11	101	102	97	86	4	27	250

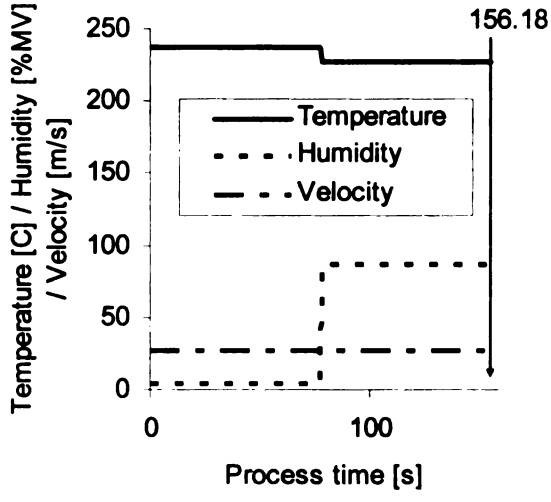
* Surface color change constraint was relaxed. (0.11~0.87)



(a)



(b)



(c)

Figure 5.21 Optimization processes and optimal control profiles for double-stage oven obtained by ICRS-FEM strategy: (a) Optimization process; (b) Optimal control profile of trial #1; (c) Optimal control profile of trial #2.

The effects of relaxed constraint were also demonstrated via trial 3 and 4 in Table 5.15. Trial 3 showed almost identical results with the previous trials (trial 1 and 2). However, trial 4 suggested LTHH profile because of the relaxed surface color change constraint. The yield was almost same with the other trials but the process time was longer than the other trials. Thus, the high temperature and low-high humidity profile seems more desirable in case of double-stage oven process.

Neural network model based optimizations were also conducted using DNNM. Although the ICRS-DNNM found optimal solutions, they fell into false-pass classification errors. Also, SNNM application was not considered, due to low model accuracy.

5.6.3 Multi-Zone Process

A multi-zone process was investigated by using the ICRS algorithm coupled with FEM (strategy #33) and DNNM (strategy #34). In Table 5.16, maximum yield achieved by the ICRS-FEM strategy was 65%, which is about 8% point higher than the maximum yield of the single-stage process.

The process time of strategy #33 was also about half the process time of the single-stage process (trial 1 in Table 5.13) even in HTLH conditions. This result is also opposite to industrial convention, which would suggest that LTHH achieves high yield. In LTHH profile, internal temperature of the patty can be easily increased to inactivate *Salmonella* and to achieve internal color change, but the high humidity also hampers nonenzymatic surface browning. However, in HTLH profile, the low impingement velocity in the ¾ portion of the entire process was minimizing yield loss until the last portion of the process, in which the impingement velocity was increased to meet the

surface color constraint. These internal reactions (*Salmonella* inactivation and internal color change) and surface reaction (surface color change) took place one after the other in the LTHH profile, which resulted in longer cooking duration. However, these internal and surface reactions took place simultaneously in the HTLH profile, which seemed to shorten the process.

In this comparison, only ICRS-DNNM strategy was tested, because the model performance of SNNM for multi-zone process was poor (Figure A.5). Although the maximum value achieved by ICRS-DNNM was 61%, the value was over-estimated by about 5% points, compared with the FEM validation result. Figure 5.22 shows stepwise velocity profiles for strategy #33 and #34. ICRS-DNNM found LTHH profile as an optimal control profile, while ICRS-FEM suggests HTLH control profiles.

When the surface color change constraint was relaxed, which ranged from 0.11 to 0.87, two trials (33-1 and 33-2 in Table 5.16) showed improved yields and HTLH type profiles. Trial 33-2 was quite impressive, in that the process achieved better results in yield and process time than the case of the double-stage oven. The improved yield (69%) was achieved just by varying impingement velocity. However, the ultimate dry conditions (~0%MV) of all the trials of strategy 33 must be considered in the practical sense. A practical lower bound for humidity was considered necessary.

Table 5.16 Maximum yield and optimal control variables were found by using ICRS-FEM and ICRS-DNNM for multi-zone process.

		Goal	Constraints			Control Variables						
					Surface Color	Temperature	Humidity	Velocity				Process Time
Strategy	Model	Yield [%]	*Salmonella* Inactivation (>6.5) [$(\log(N/N_0)$]	Internal Color Change (>1.4) [$\log(C/C_0)$]	Change (0.26~0.497) (0.11~0.87)* [$\log(C/C_0)$]	T [°C]	H [%MV]	V1 [m/s]	V2 [m/s]	V3 [m/s]	V4 [m/s]	t_f [s]
33	FEM	65	57.45	1.44	0.49	240	0	7	8	8	24	161
33-1*		66	63.81	1.57	0.50	229	0	4	6	18	23	169
33-2*		69	57.76	1.40	0.59	247	0	3	13	27	7	152
34	DNNM	61	29.76	1.40	0.42	105	97	29	6	2	9	328
	FEM**	55	3.26	1.57	0.35	105	97	29	6	2	9	328

* Surface color change constraint was relaxed. (0.11~0.87)

** Strategy #34 was validated with FEM.

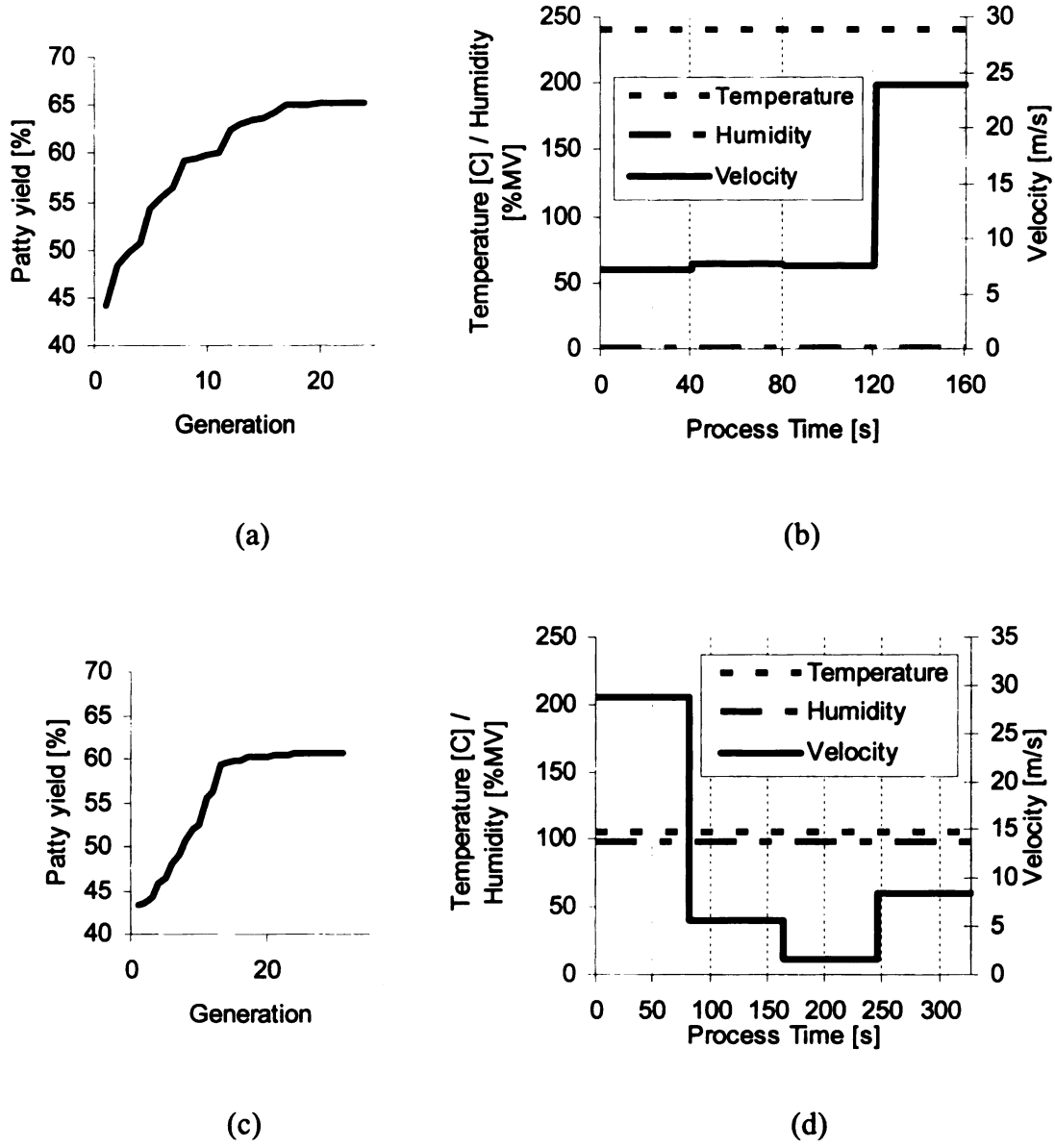


Figure 5.22 Optimization process and optimal control profile were obtained by using ICRS-FEM and ICRS-DNNM for multi-zone process: (a) Optimization process of strategy #33; (b) Optimal control profiles of strategy #33; (c) Optimization process of strategy #34; (d) Optimal control profiles of strategy #34.

5.6.4 Economic-Based Optimization

Based on the single-stage process model, a profit model was developed according to the equations in Section 4.7.4. Then, the ICRS algorithm was used to search for a maximum profit, rather than simply a maximum yield. All the control variables, such as temperature, humidity, velocity, process duration, and yield, converged to 243 °C, 2 %MV, 30 m/s, 154 s, and 64%, respectively (Figure 5.23 (a)). The optimal profile suggested HTLH policy, which is also consistent with the second trial of the single-stage optimization. The maximum yield reached 64%, which is much higher than the yield of the single-stage process, even though the optimization was based on the same predictive model. This is possible, because the objective function value is not yield, but profit, which might affect the solution space feature. Therefore, the same algorithm could reach a much higher objective value. The profit started from negative value and turned to gain after passing around 55% yield (Figure 5.23 (b)). Plot (c) of Figure 5.23 shows that the energy cost is fluctuating as the control variables changes. However, the impact of the energy cost was much smaller than the feed cost or product value. The product value was proportional to the yield, and the feed cost was inversely proportional to the process duration.

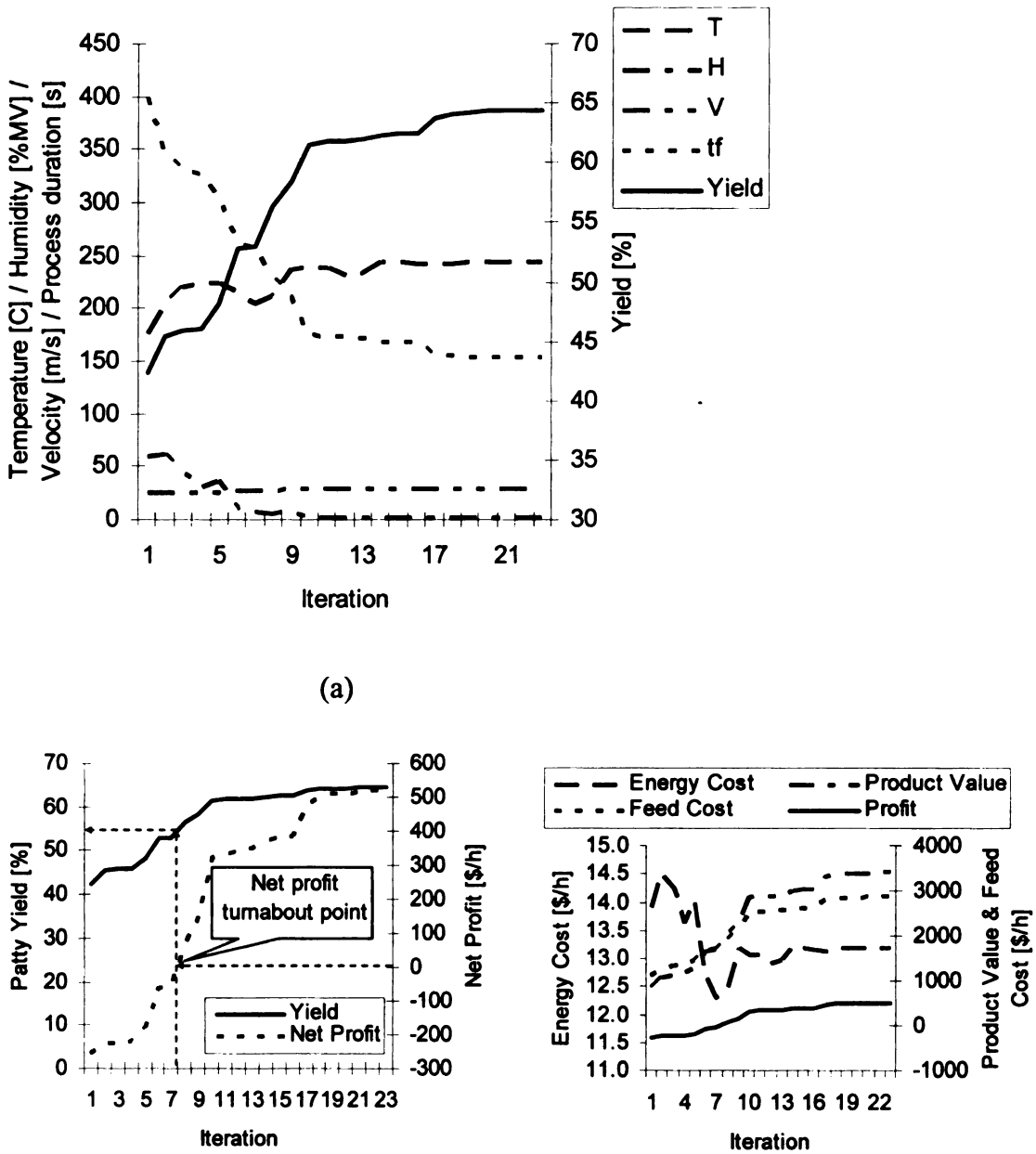


Figure 5.23 Optimization processes and optimal control variables for maximum profit of single-stage oven were found by using ICRS-FEM strategy: (a) Convergence history; (b) Comparison of net profit and patty yield; (c) Various cost history.

In addition, the sensitivity of the optimal solution was investigated by using Monte Carlo simulation (Equation [5.2]). As in the previous cases, the optimal solution was also highly sensitive to the constraints. All the 100 simulations with standard deviation of 2 °C, 2 %MV, 0.1 m/s, and 1 s (for temperature, humidity, velocity, and process duration) failed to satisfy the constraints (Table 5.17). However, in Table 5.17, a sub-optimal solution (241°C, 3 %MV, 29 m/s, and 175 s at 10^{th} iteration) showed 47% risk of failure, which was much lower than the risk of failure of the optimal solution. Even though the sub-optimal solution was more practical for application, there was 200 $/h difference with the optimal solution, which was estimated $720,000 difference annually (based on 12 h operation/day for 300 day/yr.).

Table 5.17 Sensitivity of the optimal solutions was obtained by using Monte Carlo simulation for economic-based optimization problem.

						Confidence Interval (95% confidence)	
Strategy	Optimal Profit [$/h]	Yield @ Optimal Profit [%]	Number of Fails N_f	Total Simulations N_t	Risk of Failure (N_f/N_t) [%]	Yield [%]	Profit [$/h]
#37	518	64	100	100	100%	N/A	N/A
	318	61	47	100	47%	61.6±0.12	324.98±5.49

6 CONCLUSIONS

6.1 General Conclusions

Global optimization algorithms, integrated process models, and control profile parameterization methods were combined and utilized to find optimal control profiles maximizing patty yield while satisfying microbial safety and quality constraints. Based on the results of the previous chapter, the following conclusions were drawn:

1. Artificial neural networks were developed and substituted as alternative process models for finite element model. Dynamic neural network models (DNNM) showed robust performance for random, single-stage, double-stage, and multi-zone processes. However, static neural network models (SNNM) demonstrated good performance just for random and single-stage processes. The DNNM and SNNM were much faster than the FEM (about 9× and 636×, respectively). Although the accuracy of DNNM and SNNM were lower than FEM, those neural network models were viable alternatives to FEM, due to improved execution time.
2. Various optimization strategies were designed with combinations of models, algorithms, and parameterization methods. The highest yield (73%) was obtained by strategy #3 (ICRS-FEM-PLI), which converged to control profiles with apparent step changes in the control variables. The strategies using neural network models found optimal solutions 10~1,000 times faster than did FEM. However, most of them committed false-pass classification errors or showed low accuracy for yield prediction, even though the DCS algorithm helped the model not commit false-pass classification errors. Compared to the other algorithms, ICRS was the most recommendable algorithm, because it was easy to set the algorithm

parameters, and ICRS showed well-balanced exploration and exploitation features. In addition, ICRS-related strategies achieved significant improvement of the objective value within relatively few iterations. Even though the highest yield was achieved by a strategy using PLI, the solutions obtained by PLI were much more sensitive than the solutions by FS. Therefore, FS is recommended for practical usefulness of the obtained solution. Even though the ideal and optimal profiles are impossible to generate practically, those ideal profiles suggest what type of general process might be a desirable strategy.

3. Single-stage, double-stage, and multi-zone processes were studied by using three different models and the ICRS algorithm. The maximum yield (67%) was achieved in the double-stage process, and the control profiles showed similar control patterns with the results from strategy #3 (FEM-PLI-ICRS) applied to a continuously varying process. Case studies showed many possible variations of the single-stage process to achieve improved solutions. In addition to the optimization for yield, a simple economic-based example was illustrated from the viewpoint of net profit for a single-stage process. Maximum profit was achieved at 64% yield, which was different from the single-stage case with a yield objective.
4. Monte Carlo simulation showed that the sensitivity of optimal solutions was very critical for the usefulness of the solution. Small perturbations of the optimal control profiles could result in failure to satisfy safety and quality constraints. Instead of using the optimal solution, however, slightly sub-optimal solutions

were less sensitive to the perturbations and therefore less susceptible to fail (for constraints).

This study was designed to build a foundation for multivariate nonlinear dynamic process optimization, such as meat patty cooking under moist air impingement environments. This type of problem is very difficult and complex from the viewpoint of product and process. In this study, various factors, such as product yield, microbial safety, and colors, were investigated to identify the best optimization strategy for this type of process. Finally, many valuable aspects related to developing the model, selecting optimization algorithms, interpreting optimal results, and handling various pitfalls were carefully studied, and knowledge was established. However, a direct application of the resultant numbers of this study to a problem requires discretion by users, because the resultant numbers, such as maximum yield and the status of constraint satisfaction, can be significantly affected by the properties related to various calculations, such as yield, safety, and quality predictions. Even so, the knowledge of process optimization strategy in this study remains sound for the applications to various complex food processing operations, such as frying, drying, extrusion, baking, and retorting.

6.2 Suggestions for Future Research

Although the results of this project are instructive and potentially valuable, further research and improvements are still necessary. The following suggestions are made for future research.

1. The process model plays a very important role in optimization. The FEM, DNNM, and SNNM models must be validated with actual experiments. For this purpose,

the integrated FEM needs to be fine-tuned with validated predictive model parameters of quality.

2. Optimal solutions found throughout this research need to be validated experimentally. Even though some optimal profiles are impractical to generate, single or double-stage results can be validated by experiment.
3. Methods or algorithms need to be developed to avoid false-pass classification errors when alternative models are used. A neural network trained with the relationship between goal and constraints might help optimization algorithms maneuver the solution space without committing false-pass classification error.
4. Even though the alternative models did not show satisfactory performance locating optimal solutions, they can be used to acquire prior knowledge for the target model. For example, sensitivity analysis can be used along with alternative models to reduce control variables, test the effect of parameters, determine the approximate location of optimal solution, etc.
5. Effective control profile parameterization methods need to be devised to address the sensitivity issues for optimal solutions. Confidence interval embedded control profiles can reduce the risk of failure involved in the high sensitivity of the optimal control profile. Also, such a high sensitivity problem needs to be validated by experiment.
6. The finite element model used in this study had some degree of error with physical experiment. In this study, how the source (finite element model) error affects the optimal solutions was not studied. However, as long as the finite element model represents the general characteristics of the solution space, the

general conclusions seem to remain sound, even though some numeric value might change. Even so, the effect of the source error on the optimal solution should be studied.

7. A new efficient global optimization algorithm having advantages from GA, SA, and ICRS can be developed and applied to this problem.
8. Typical installations of a single-stage oven have an inlet and outlet tunnel (often with saturated condition) in addition to the major cooking zone. These actual process conditions need to be implemented in the control profiles to reflect these systems and to obtain results that are directly practical and interesting to the industry.

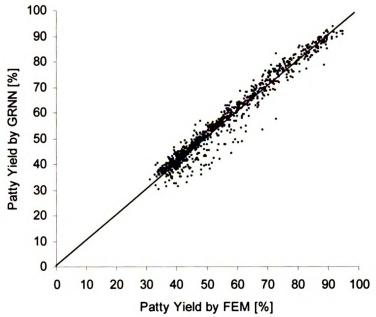
APPENDICES

APPENDIX A

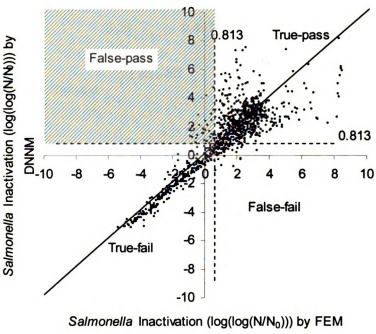
Tables and Figures of Optimization Processes

Polynomial regression lines were added to smooth piecewise control profiles so that general trend could be observed in the following plots:

Figure A.7 (a), Figure A.10 (a), Figure A.12 (a), and Figure A.14 (a)

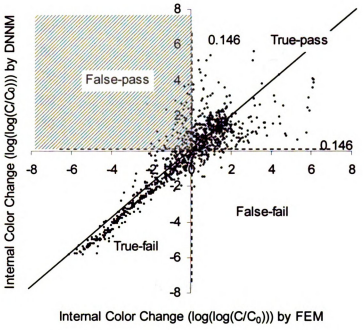


(a)

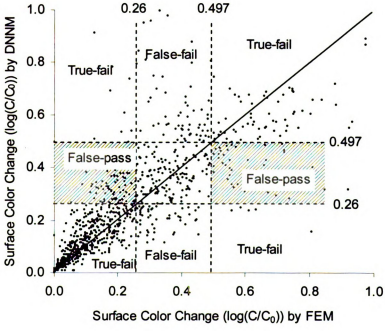


(b)

Figure A.1 The performance of DNNM for double-stage process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

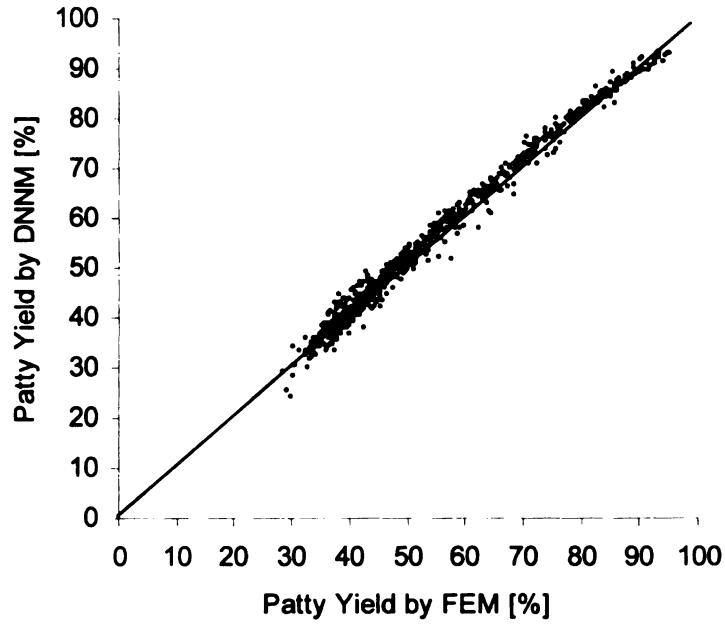


(c)

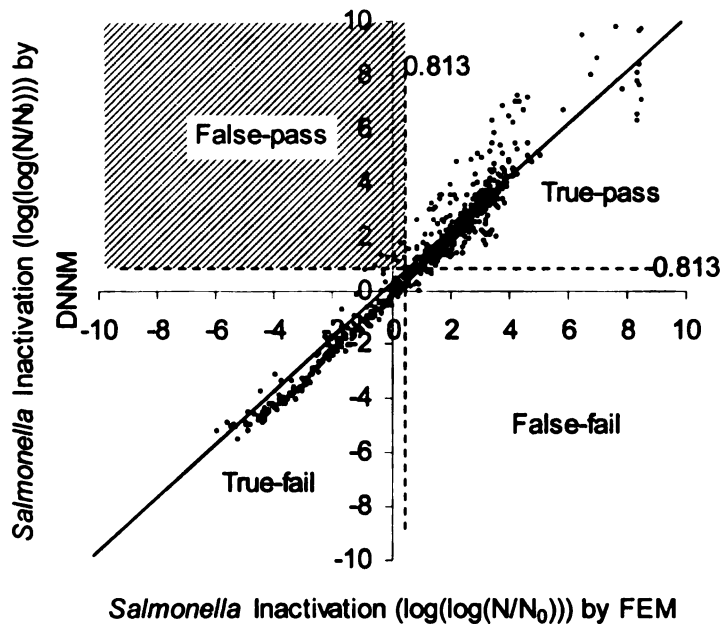


(d)

Figure A.1 (cont'd)

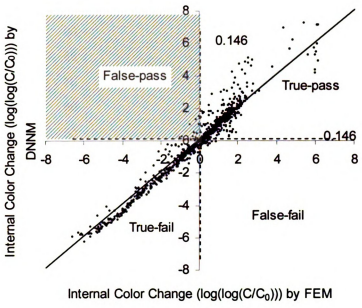


(a)

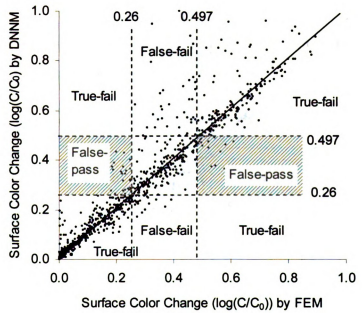


(b)

Figure A.2 The performance of the DNNM for multi-zone process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

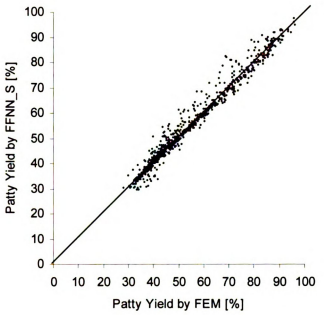


(c)

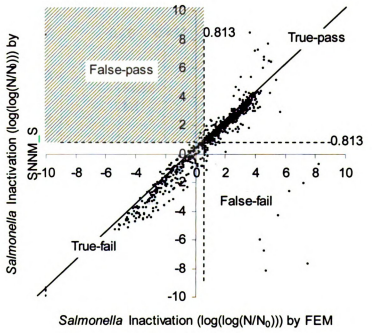


(d)

Figure A.2 (cont'd)

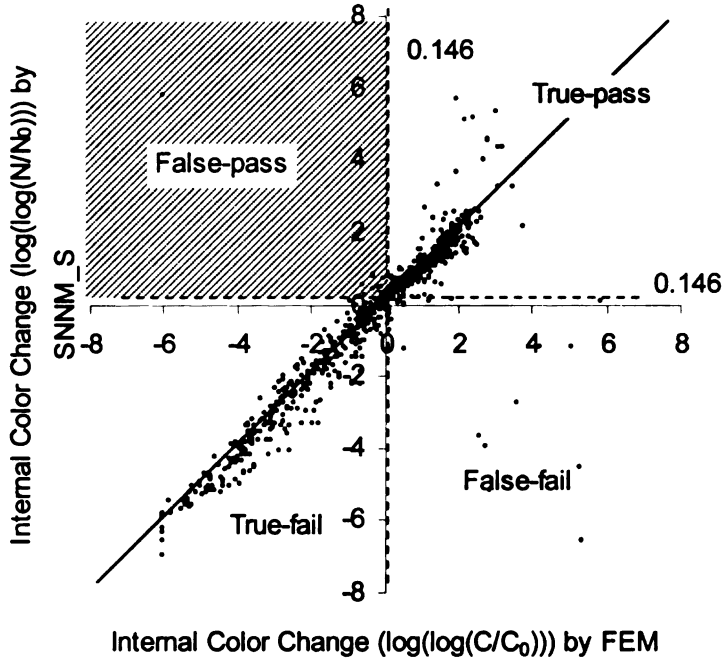


(a)

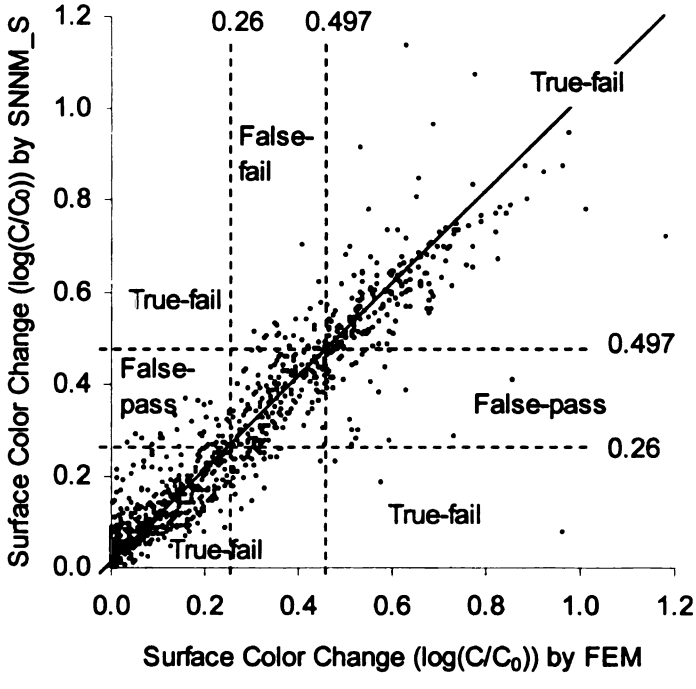


(b)

Figure A.3 The performance of the SNNM for single-stage process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

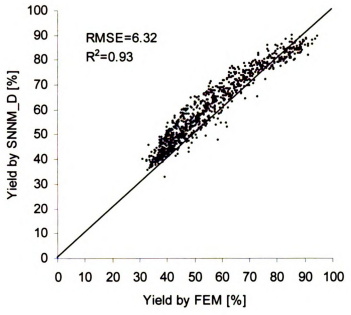


(c)

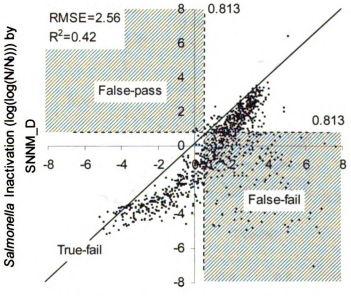


(d)

Figure A.3 (cont'd)

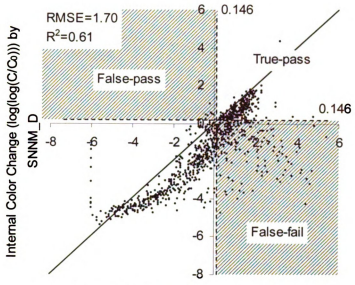


(a)

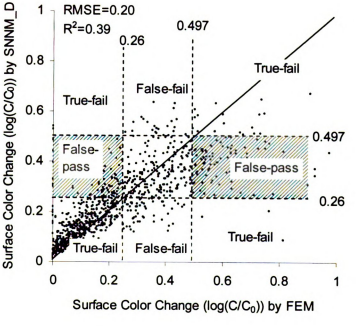


(b)

Figure A.4 The performance of the SNNM for double-stage process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.

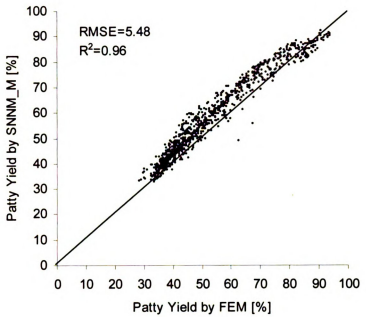


(c)

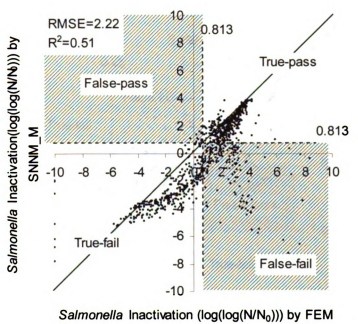


(d)

Figure A.4 (cont'd)

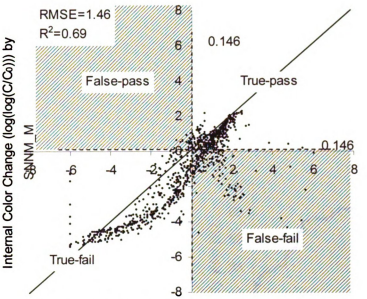


(a)

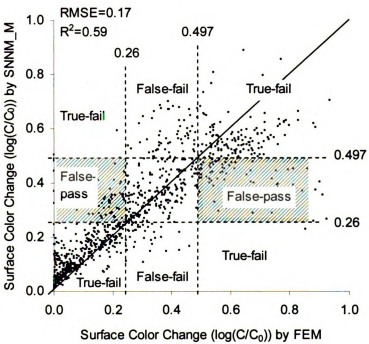


(b)

Figure A.5 The performance of the SNNM for multi-zone process: (a) Yield; (b) *Salmonella* inactivation; (c) Internal color change; (d) Surface color change.



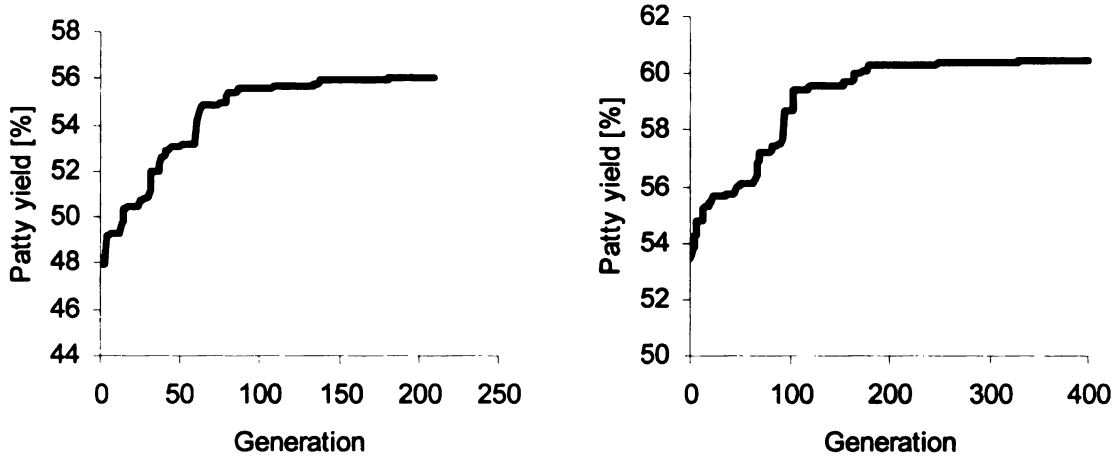
(c)



(d)

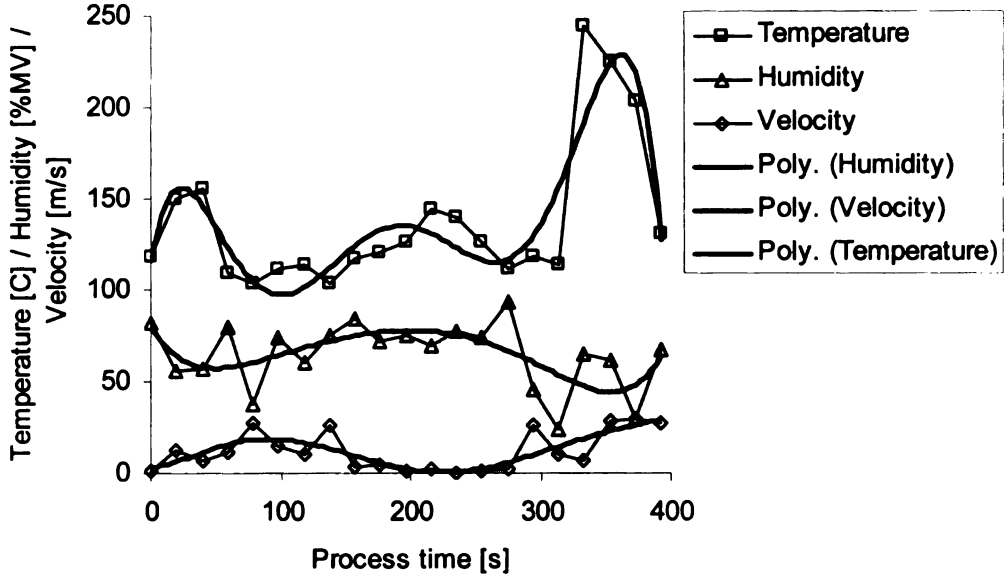
Figure A.5 (cont'd)

Strategy #1 & #5

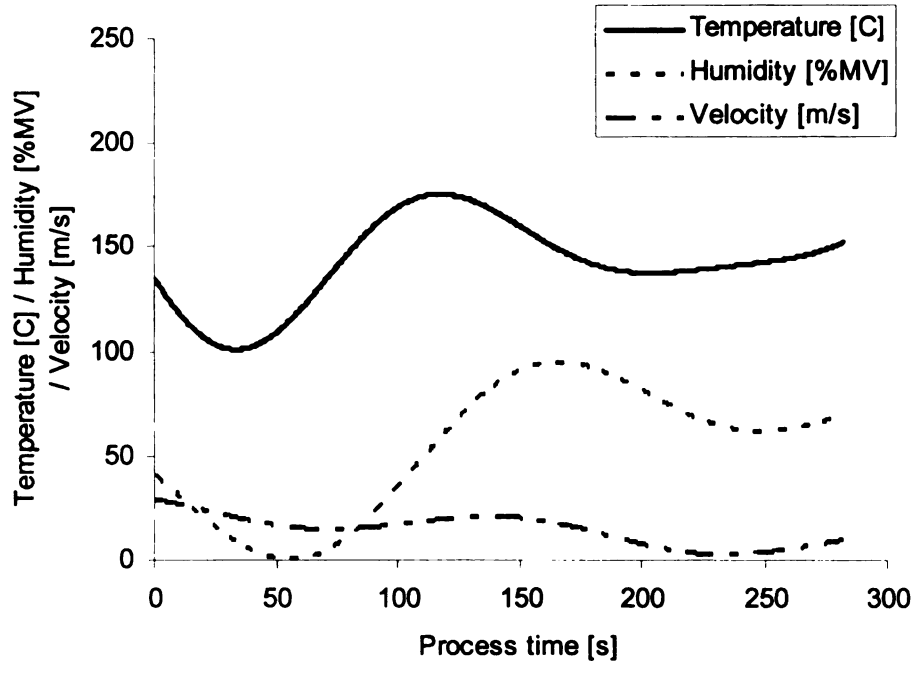


(a) (b)

Figure A.6 GA-FEM optimization process and convergence: (a) PLI parameterization; (b) FS parameterization.

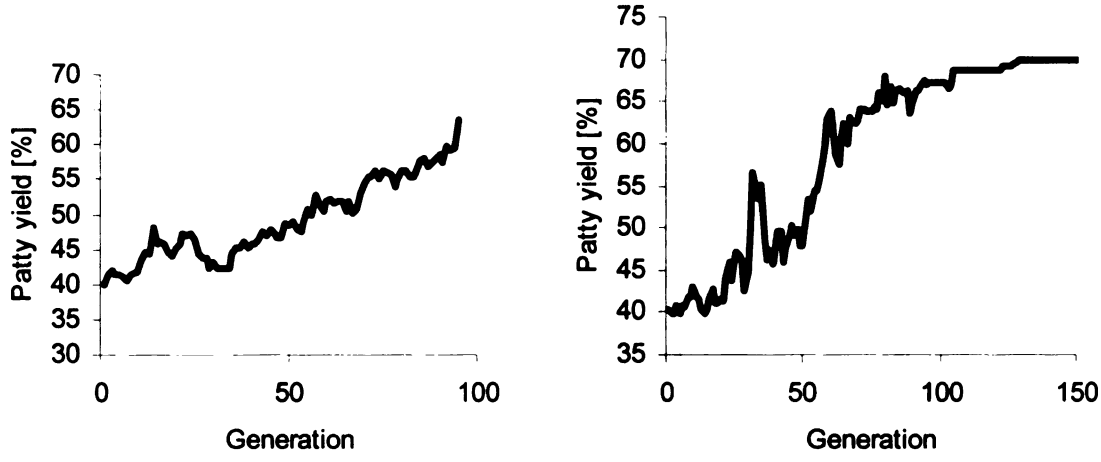


(a)



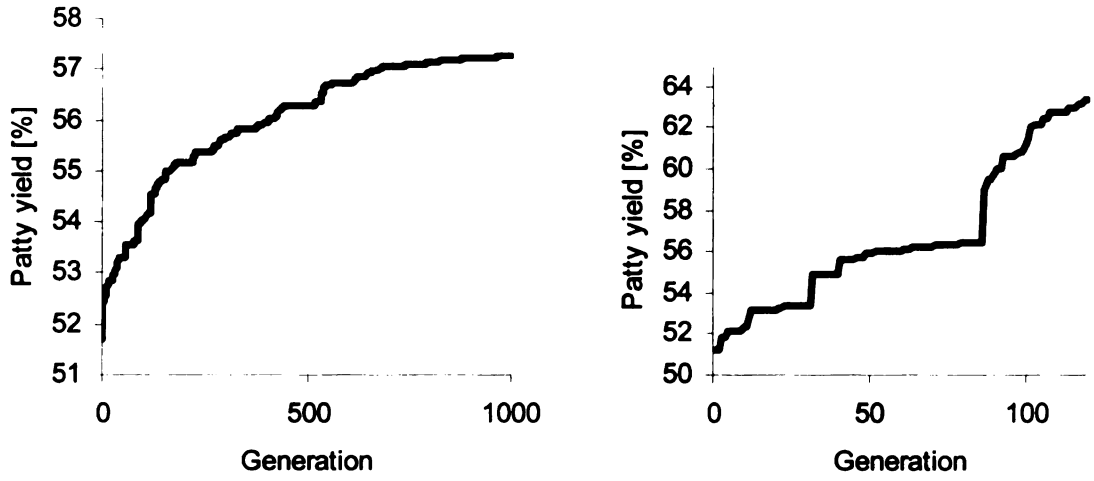
(b)

Figure A.7 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by GA-FEM optimization strategy: (a) PLI parameterization; (b) FS parameterization.



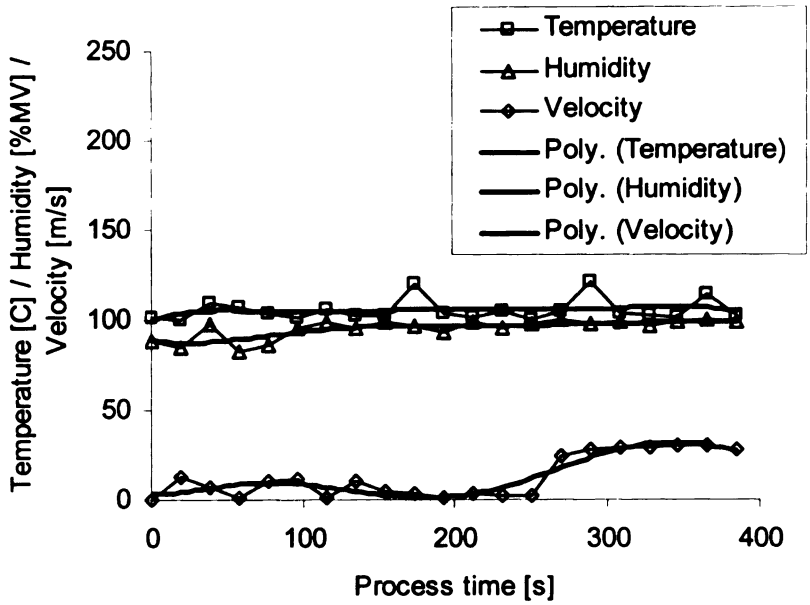
(a) (b)

Figure A.8 SA-FEM optimization process and convergence: (a) PLI parameterization; (b) FS parameterization.

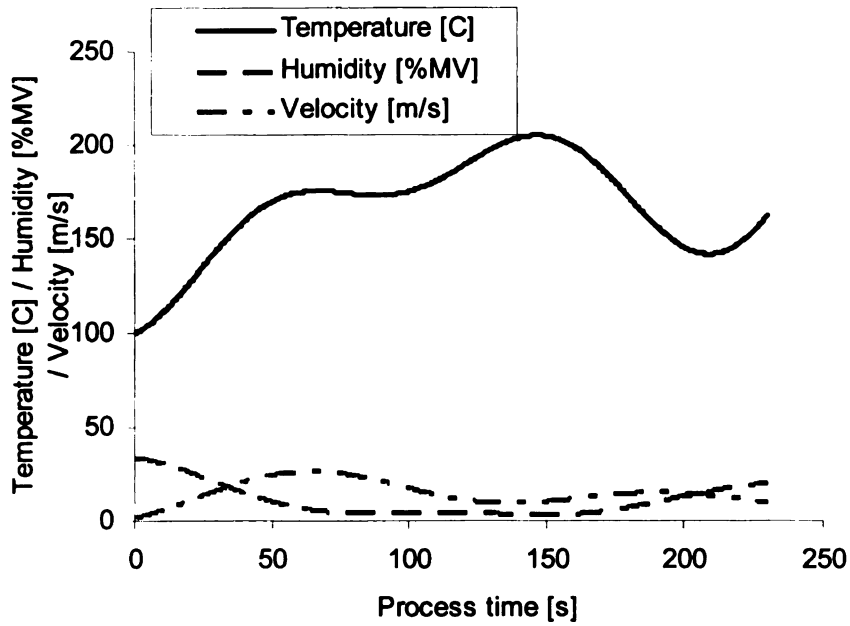


(a) (b)

Figure A.9 GA-DNNM optimization process and convergence: (a) PLI parameterization; (b) FS parameterization.

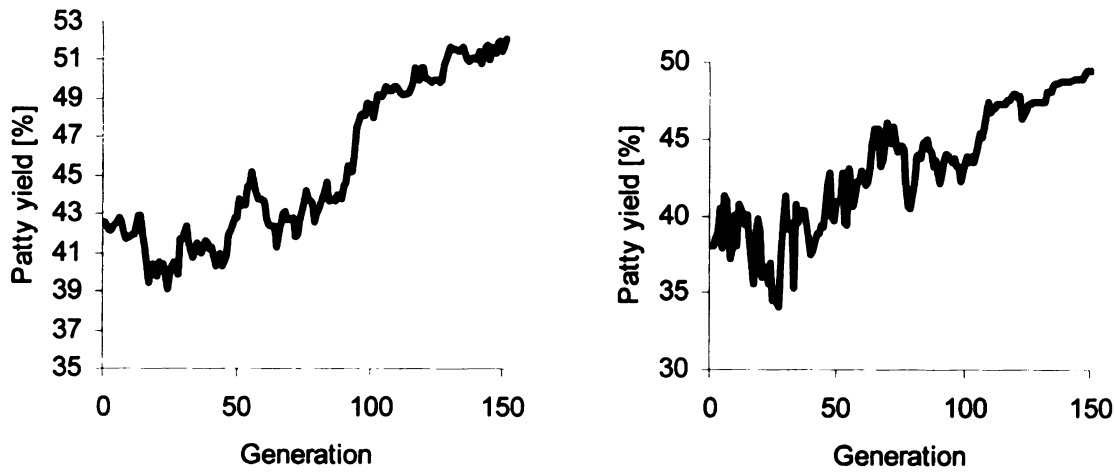


(a)



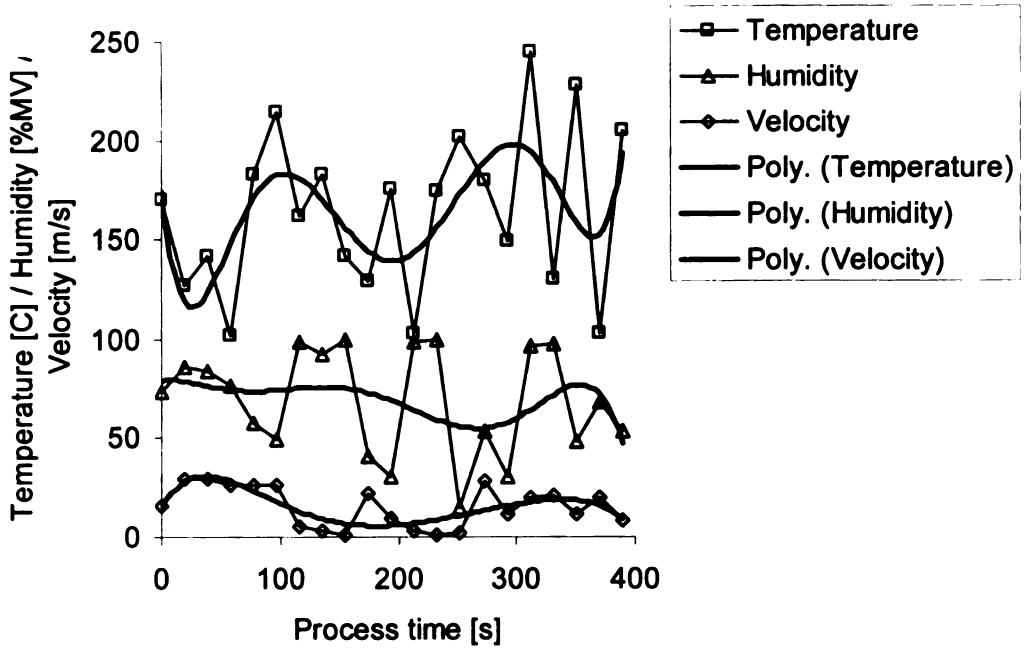
(b)

Figure A.10 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by GA-DNNM optimization strategy: (a) PLI parameterization; (b) FS parameterization.

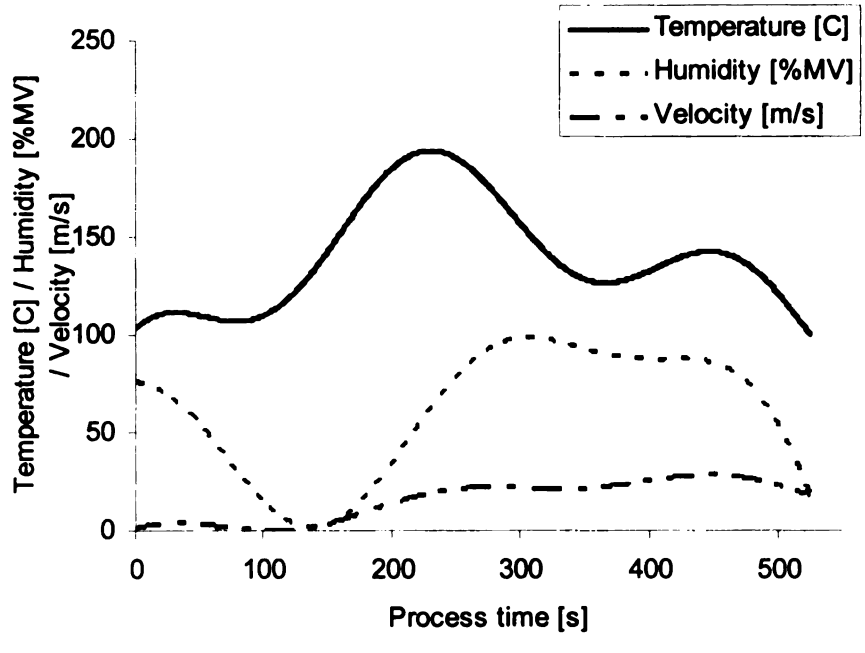


(a) (b)

Figure A.11 SA-DNNM optimization process and convergence: (a) PLI parameterization; (b) FS parameterization.

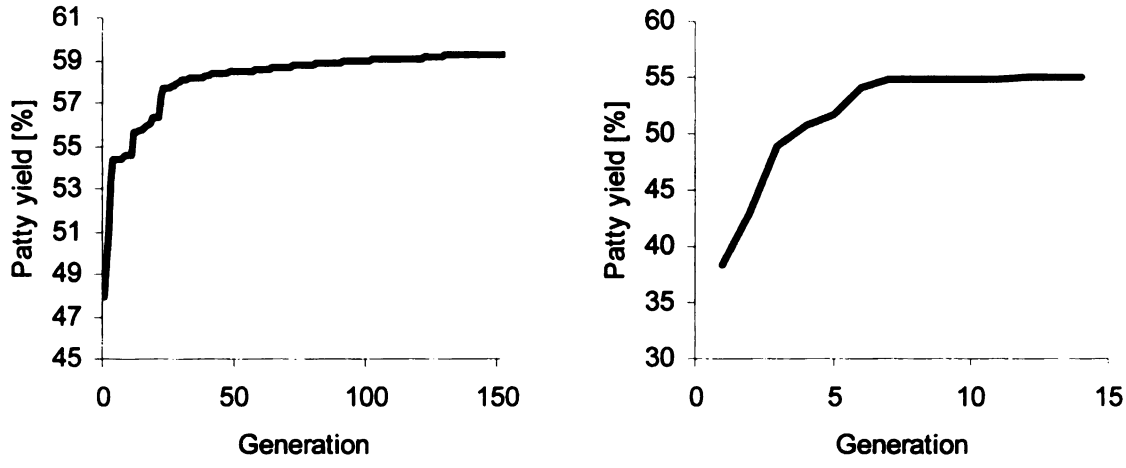


(a)



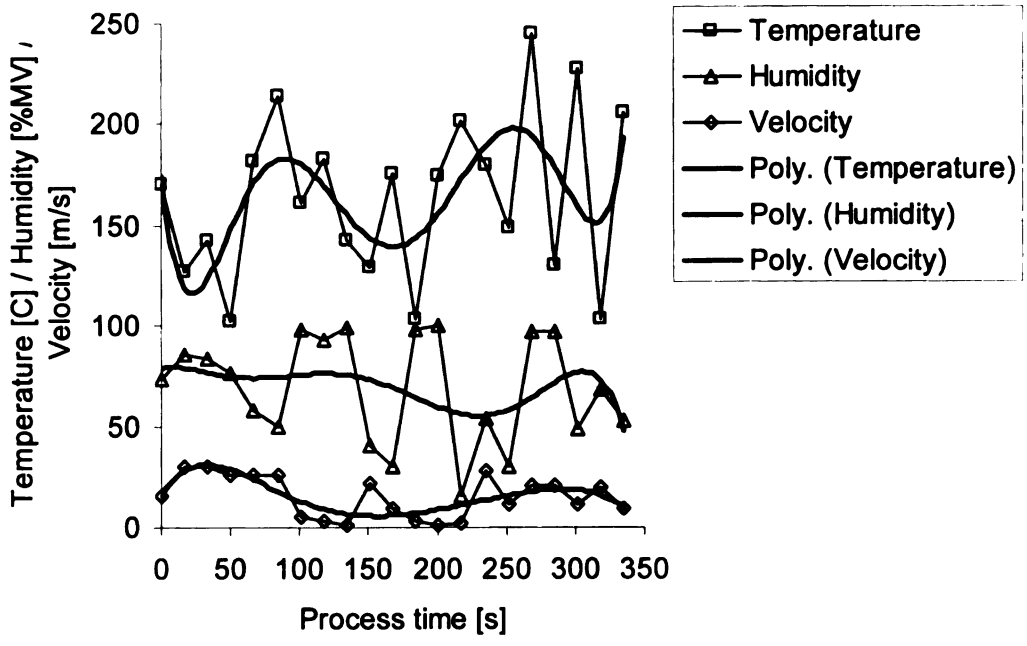
(b)

Figure A.12 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by SA-DNNM optimization strategy: (a) PLI parameterization; (b) FS parameterization.

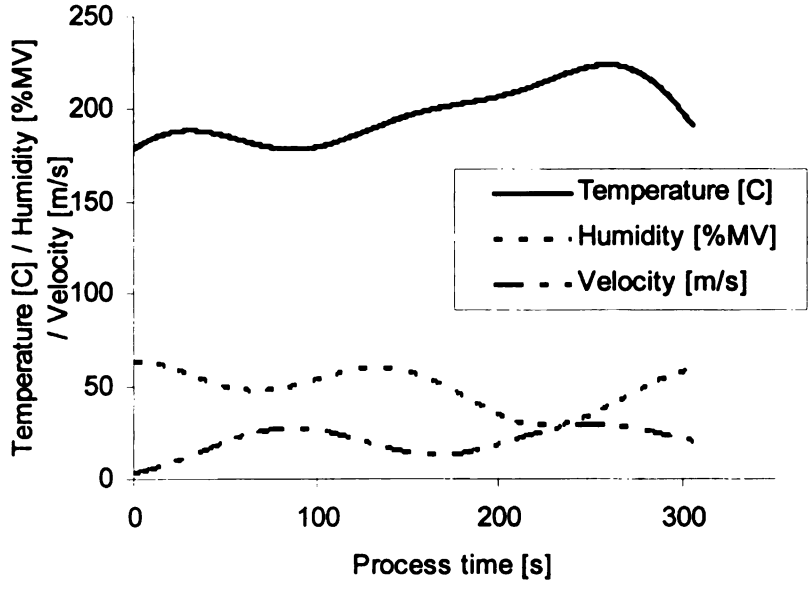


(a) (b)

Figure A.13 ICRS-DNNM optimization process and convergence: (a) PLI parameterization; (b) FS parameterization.



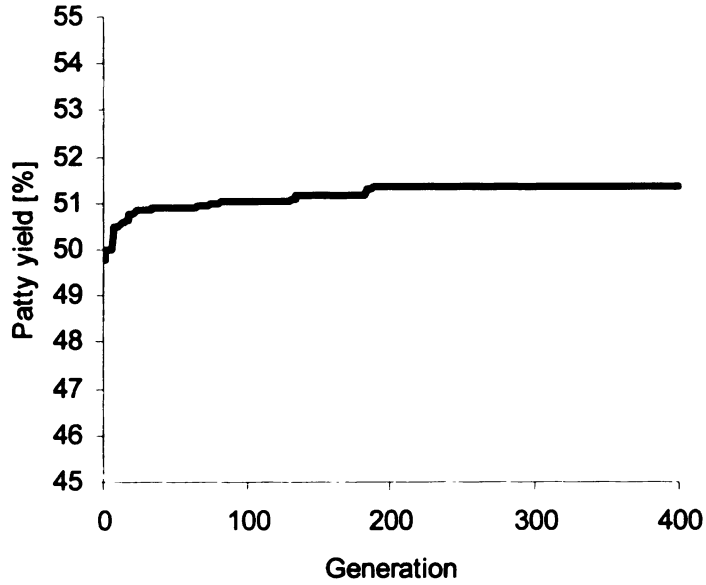
(a)



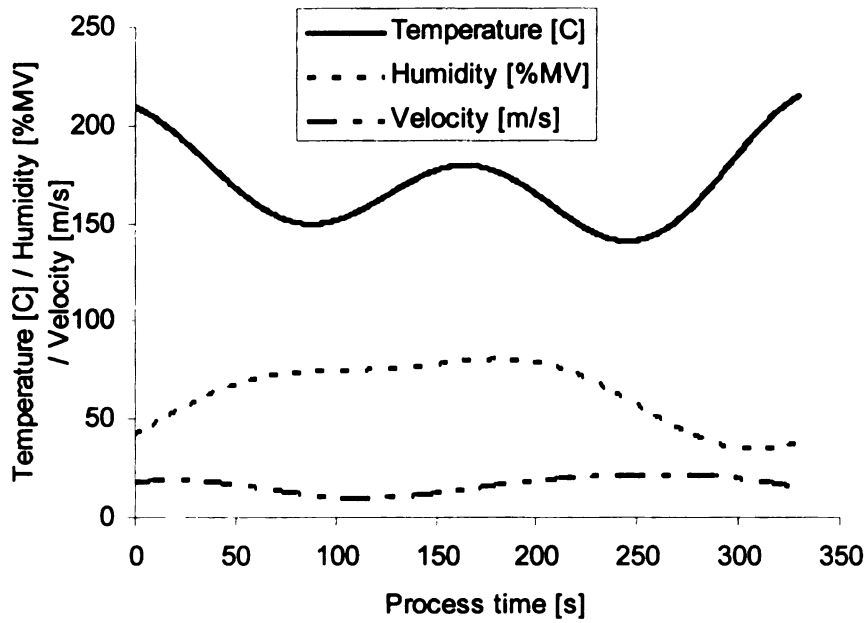
(b)

Figure A.14 Optimal control profile of temperature, humidity, impingement velocity, and cooking duration found by ICRS-DNNM optimization strategy: (a) PLI parameterization; (b) FS parameterization.

Strategy #21



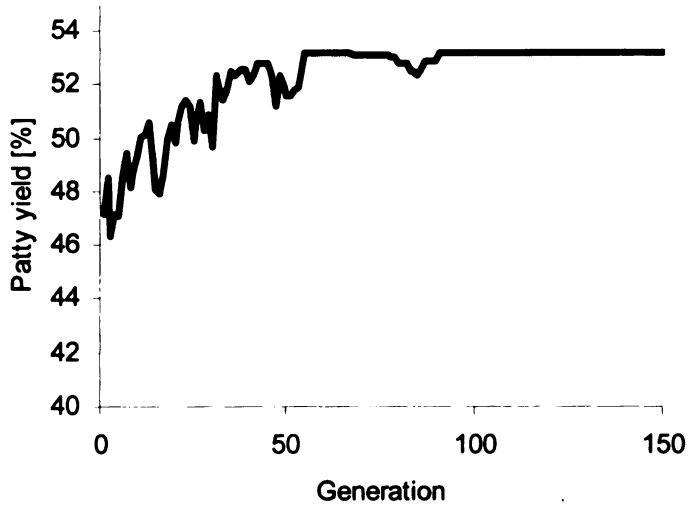
(a)



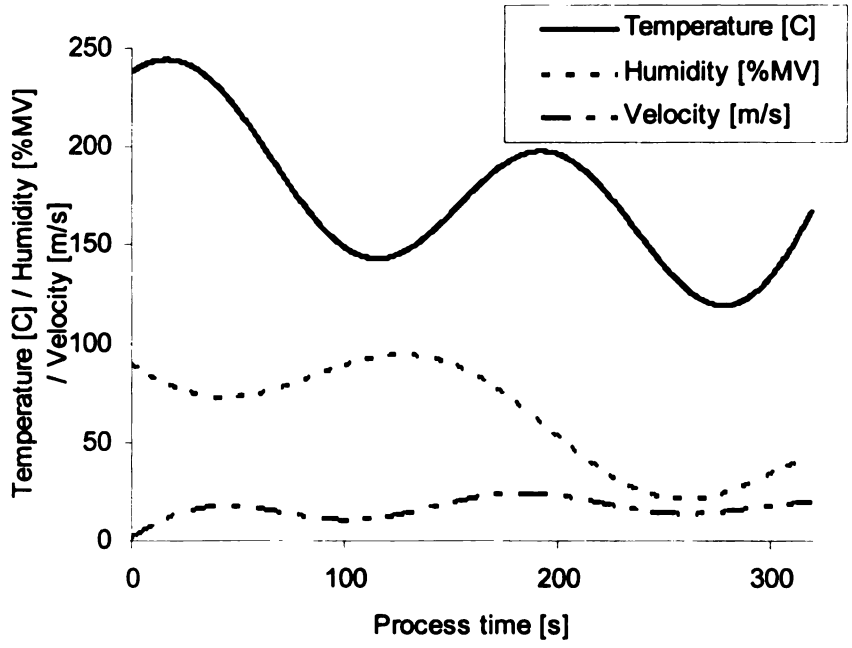
(b)

Figure A.15 Optimization process and optimal control profiles found by GA-SNNM_R optimization strategy and Fourier series parameterization: (a) Convergence history; (b) Optimal control profile of temperature, humidity, impingement velocity, and cooking duration.

Strategy #22



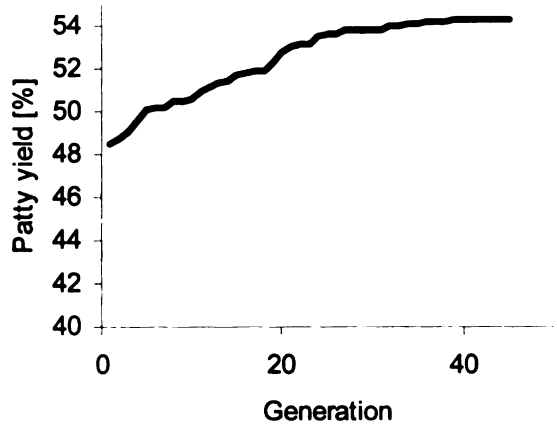
(a)



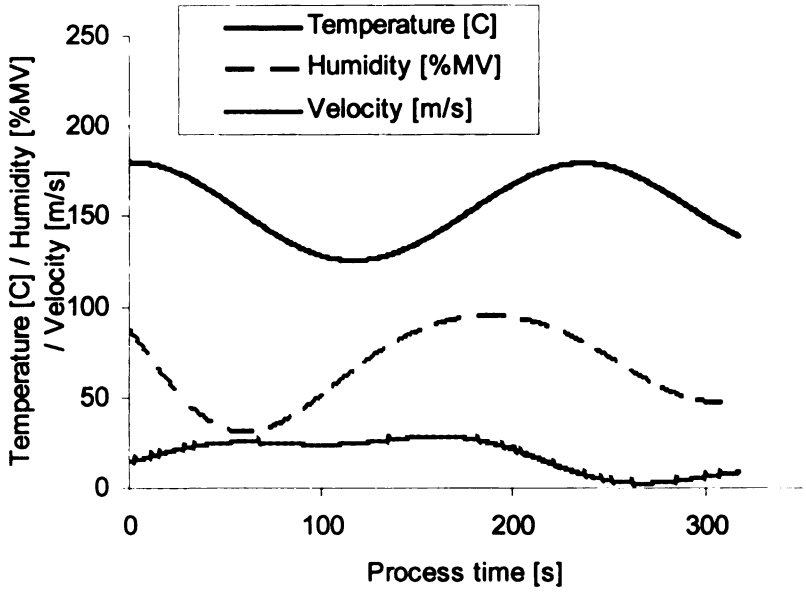
(b)

Figure A.16 Optimization process and optimal control profiles found by SA-SNNM_R optimization strategy and Fourier series parameterization: (a) Convergence history; (b) Optimal control profile of temperature, humidity, impingement velocity, and cooking duration.

Strategy #23

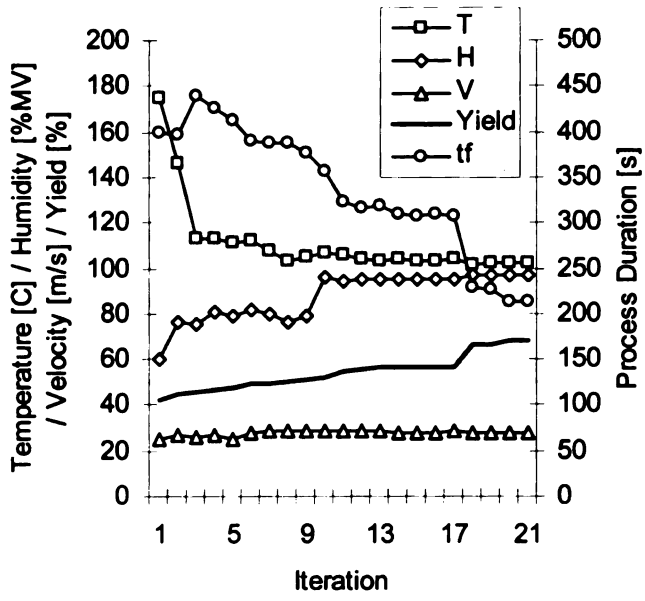


(a)

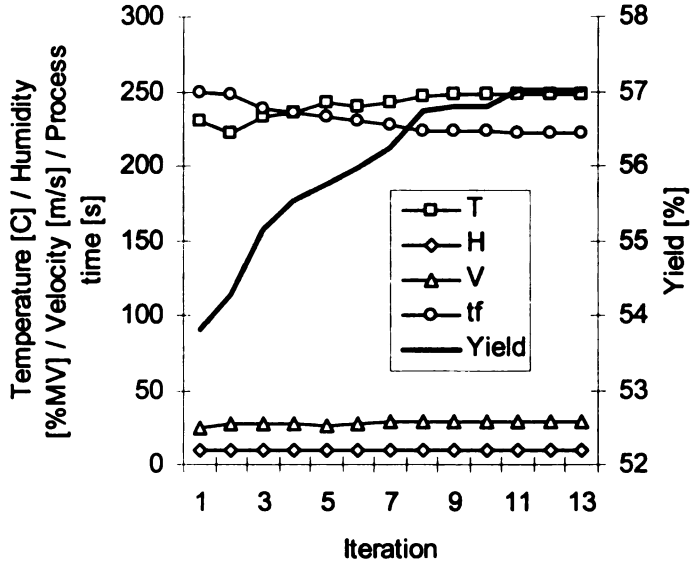


(b)

Figure A.17 Optimization process and optimal control profiles found by ICRS-SNNM_R optimization strategy and Fourier series parameterization: (a) Convergence history; (b) Optimal control profile of temperature, humidity, impingement velocity, and cooking duration.

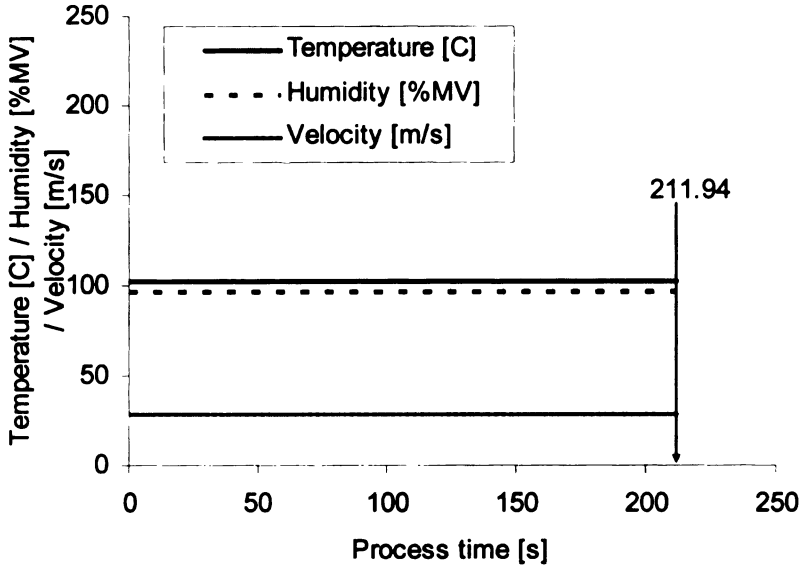


(a)

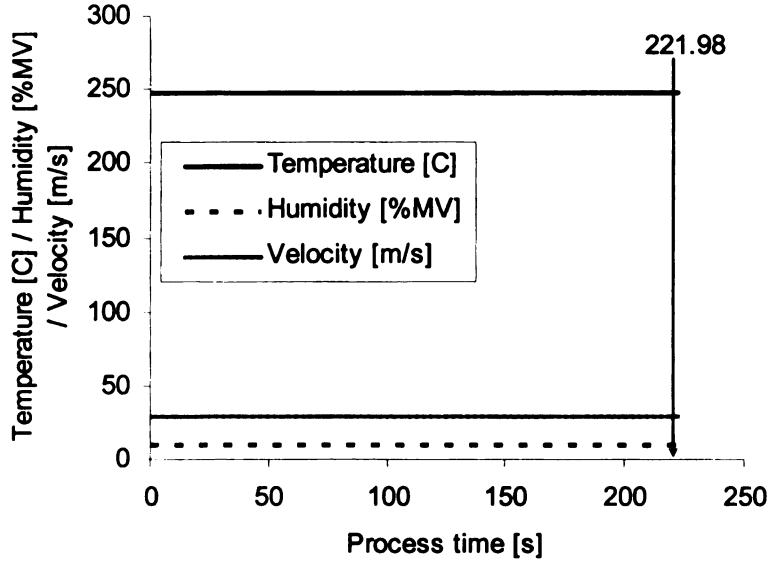


(b)

Figure A.18 Two different optimization processes and optimal control profiles found by single-stage ICRS-FEM optimization strategy: (a) Convergence history of case I (LTHH); (b) Convergence history of case II (HTLH); (c) Optimal control profile of temperature, humidity, impingement velocity, and cooking duration of case I; (d) Optimal control profile of temperature, humidity, impingement velocity, and cooking duration of case II.



(c)



(d)

Figure A.18 (cont'd)

APPENDIX B

Computer Codes

User Guide

In this study, various optimization strategies and case studies were coded via Matlab. Even though there are many computer codes for each cases, some typical codes were represented in the appendix. Many other codes were simple variations of those typical codes. Also, some sub-programs were commonly used in the other applications with slight variations.

The appendix has codes related to two neural network model development (DNNM and SNNM), three GO algorithms (GA, SA, and ICRS), and two parameterization methods (PLI and FS). However, the entire FEM code was not included, except some quality prediction sub-programs, because the large portion of the work was done by Watkins (2004).

MATLAB codes for case studies have the same frameworks with other strategies. The only difference is the sub-program generating random profiles, which has specific number of control variables for each applications, 1-stage (4 control variables), 2-stage (7 control variables), and multi-zone process (7 control variables). Therefore, those codes were not included in this appendix.

Dynamic Neural Network Model (DNNM)

Train_DNNM.m has following two sub programs.

- ***Generate_SeqData_Train.m***
- ***Generate_SeqData_Test.m***

```
% Train_DNNM.m
% This program trains a dynamic neural network model as an alternative
to
% a finite element model by using a nonlinear system identification
method.

% ==============================
%    TRAINING PHASE
% ==============================

clear all
fprintf('Network Training Started...\n')
fprintf('>Training Phase:\n')

counter=0;

Nh=1; % Global training matrix head number
Nt=0; % Global training matrix tail number

m=2;        % length of past time series
mg=3*m+1+1; % total number of rows of training matrix
mk=3*m+1;   % row # of target output value at global training matrix

% Constructing global trainging data set
for j=1:1
    start =[1   209]; % start file #
    finish=[910 328]; % finish file #
    fprintf('Total number of training data = %d\n',finish(j))
    for i=start(j):finish(j)

        % clear interim variables
        clear  a dP1 dP2 dP3 dP4 dT1 dT2 dT3 dT4 P1 P2 P3 P4 T1 T2 T3
T4
        path=j;
        switch path
            case 1

fid3=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\Train\FE Solution
Files\Constant\FEsolution_',num2str(i),'.dat']);
            case 2

fid3=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\Test\FE Solution
Files\FEsolution_',num2str(i),'.dat']);
        end

        % Reading individual data files
        a=fscanf(fid3,'%f %f %f %f %f %f %e %e %e %e %f %f',[12,
inf]); % [time temp steam velocity yield VavgM SamonellaLR DI DE4 DEavg
Tc Ts]
        a=a';
        fclose(fid3);

        % Data Correction: Abnormal data trend will be truncated
```

```
        % The total row# of data file is 1201. If the file has abnormal
data, fscanf terminates its reading right at the beginning of the
abnomality. In this case the total number of row of the file will be
less than 1200
        [rn cn]=size(a);
        if rn < 1200
            a=a(1:rn-20,:);
        end

        % Converting concurrent data to sequential data format
        [P1, T1]=Generate_SeqData_Train(a,5); % 5 Yield
        [P2, T2]=Generate_SeqData_Train(a,7); % 7 Salmonella Reduction
        [P3, T3]=Generate_SeqData_Train(a,8); % 8 DI (Doneness via
internal color change
        [P4, T4]=Generate_SeqData_Train(a,10);% 10 DEavg (Averaged
Doneness via surface color change

        [rn1,cn1]=size(P1(mg,:));
        [rn2,cn2]=size(P2(mg,:));
        [rn3,cn3]=size(P3(mg,:));
        [rn4,cn4]=size(P4(mg,:));

        % Log10(P2 & P3, T2 & T3)for scaling
        P2=prelog('p',P2,m); T2=prelog('t',T2,m);
        P3=prelog('p',P3,m); T3=prelog('t',T3,m);


        % Calculating difference of target values in an interval
        for h=1:cn3
            if h==1
                dT1(1,h)=P1(mk,h)-T1(1,h);
                dT2(1,h)=P2(mk,h)-T2(1,h);
                dT3(1,h)=P3(mk,h)-T3(1,h);
                dT4(1,h)=T4(1,h)-0;
            else
                dT1(1,h)=T1(1,h-1)-T1(1,h);
                dT2(1,h)=T2(1,h-1)-T2(1,h);
                dT3(1,h)=T3(1,h-1)-T3(1,h);
                dT4(1,h)=T4(1,h)-T4(1,h-1);
            end
        end

        % Replacing the target value with the difference
        T1(1,1:cn3)=dT1;
        T2(1,1:cn3)=dT2;
        T3(1,1:cn3)=dT3;
        T4(1,1:cn3)=dT4;

        % Creating a global training file from individual file
        Nt=Nh+cn1;

        SP1(1:mg,Nh:Nt-1)=P1(1:mg,:); ST1(1,Nh:Nt-1)=T1(1,:); % Yield
        SP2(1:mg,Nh:Nt-1)=P2(1:mg,:); ST2(1,Nh:Nt-1)=T2(1,:); %
Salmonella LR
        SP3(1:mg,Nh:Nt-1)=P3(1:mg,:); ST3(1,Nh:Nt-1)=T3(1,:); % DI
        SP4(1:mg,Nh:Nt-1)=P4(1:mg,:); ST4(1,Nh:Nt-1)=T4(1,:); % DEavg
```

```
        Nh=Nt;
    end
end % End of constructing a global training data set
fprintf('End of constructing a global training data set\n')

% Normalization
[SP1n,minSP1,maxSP1] = premnmx(SP1);
[SP2n,minSP2,maxSP2] = premnmx(SP2);
[SP3n,minSP3,maxSP3] = premnmx(SP3);
[SP4n,minSP4,maxSP4] = premnmx(SP4);

[ST1n,minST1,maxST1] = premnmx(ST1);
[ST2n,minST2,maxST2] = premnmx(ST2);
[ST3n,minST3,maxST3] = premnmx(ST3);
[ST4n,minST4,maxST4] = premnmx(ST4);


% Training networks for each output by using Generalized Regression NN
net1=newgrnn(SP1n, ST1n, 0.5);   % Yield
net2=newgrnn(SP2n, ST2n, 0.22);  % Salmonella
net3=newgrnn(SP3n, ST3n, 0.22);  % DI
net4=newgrnn(SP4n, ST4n, 0.20);  % DE

fprintf('\n>End of Training\n')

% Saving trained networks
save('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net1','net1');
save('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net2','net2');
save('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net3','net3');
save('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net4','net4');

% Saving normalization parameters
%    SP1      SP2     SP3     SP4     ST1     ST2     ST3     ST4
% min max  min max
% row 1~8
% row 9
Outputs(1:8,1)=minSP1; Outputs(1:8,3)=minSP2; Outputs(1:8,5)=minSP3;
Outputs(1:8,7)=minSP4;
Outputs(1:8,2)=maxSP1; Outputs(1:8,4)=maxSP2; Outputs(1:8,6)=maxSP3;
Outputs(1:8,8)=maxSP4;
Outputs(9,1)=minST1; Outputs(9,3)=minST2; Outputs(9,5)=minST3;
Outputs(9,7)=minST4;
Outputs(9,2)=maxST1; Outputs(9,4)=maxST2; Outputs(9,6)=maxST3;
Outputs(9,8)=maxST4;

fid=
fopen('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\normal_para.dat'
,'wt');
fprintf(fid,'%10.8f %10.8f %10.8f %10.8f %10.8f %10.8f %10.8f %10.8f
\n',Outputs');
fclose(fid);


% Retrieving stored networks and parameters
clear net
load('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net1','net1');
load('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net2','net2');
```

```
load('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net3','net3');
load('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\net4','net4');

fid=fopen('C:\MATLAB7\work\Dissertation\NN\TrainedNetWorks\normal_para.
dat');
a=fscanf(fid,'%f %f %f %f %f %f %f',[8, inf]); % [time temp steam
velocity yield VavgM SamonellaLR DI DE4 DEavg Tc Ts]
a=a';
fclose(fid);

minSP1=a(1:8,1); minSP2=a(1:8,3); minSP3=a(1:8,5); minSP4=a(1:8,7);
maxSP1=a(1:8,2); maxSP2=a(1:8,4); maxSP3=a(1:8,6); maxSP4=a(1:8,8);
minST1=a(9,1); minST2=a(9,3); minST3=a(9,5); minST4=a(9,7);
maxST1=a(9,2); maxST2=a(9,4); maxST3=a(9,6); maxST4=a(9,8);

m=2;        % length of past time series
mg=3*m+1+1; % total number of rows of training matrix
mk=3*m+1;   % row # of target output value at training matrix


%==============================
%   SIMULATION PHASE
%==============================
fprintf('>Simulation Phase\n')

clear Outputs
clear Y_pred L_pred DI_pred DE_pred t
clear Y_meas L_meas DI_meas DE_meas

for counter=10:10 % 13
    clear a b Cond ittc1 ittc2 ittc3 ittc4
    clear itt1 itt2 itt3 itt4
    clear dPP1 dPP2 dPP3 dPP4 dTT1 dTT2 dTT3 dTT4
    clear PP1 PP2 PP3 PP4 TT1 TT2 TT3 TT4

    % Reading result files
    fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\Test\FE Solution
Files\FEsolution_',num2str(counter),'.dat']);
    a=fscanf(fid,'%f %f %f %f %f %f %e %e %e %e %f %f',[12, inf]); %
[time temp steam velocity yield VavgM SamonellaLR DI DE4 DEavg Tc Ts]
    a=a';
    fclose(fid);

    % Reading condition files
    fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\Test\Condition
Files\Condition_',num2str(counter),'.dat']);
    b=fscanf(fid,'%f %f %f %f',[4, inf]); % [time temp steam velocity
yield VavgM SamonellaLR DI DE4 DEavg Tc Ts]
    b=b';
    fclose(fid);

    % Reconstructing testing matrix
    [PP1, TT1]=Generate_SeqData_Test(a,5);  % Yield
    [PP2, TT2]=Generate_SeqData_Test(a,7);  % Salmonella LR
    [PP3, TT3]=Generate_SeqData_Test(a,8);  % DI
    [PP4, TT4]=Generate_SeqData_Test(a,10); % DEavg
```

```
% log10(ipp2, ipp3)scaling
PP2=prelog('p',PP2,m);
PP3=prelog('p',PP3,m);

% Recording measured data for comparison
measuredTT1=PP1(mk,:);
measuredTT2=PP2(mk,:);
measuredTT3=PP3(mk,:);
measuredTT4=PP4(mk,:);


% Sampling contol profile from condition file
Cond=Generate_SeqData_Cond(b);
[rt1,ct1]=size(PP1);

% Assigning initial values
% Yield
ipp1(1:mg,1)=Cond(1:mg,1); % copy
ipp1(mk,1)=100;  % 100 %
itt1(1:m-1,1)=0; % initial difference is zero

% Salmonella reduction
ipp2(1:mg,1)=Cond(1:mg,1);
ipp2(mk,1)=-10; % ~0
itt2(1:m-1,1)=0;

% Internal doneness via internal color change
ipp3(1:mg,1)=Cond(1:mg,1);
ipp3(mk,1)=-10; % ~0
itt3(1:m-1,1)=0;

% External doneness via surface color chane
ipp4(1:mg,1)=Cond(1:mg,1);
ipp4(mk,1)=0; % 0
itt4(1:m-1,1)=0;


% Initial simulation

% Scaling input data by using MinMax value at the trining phase
ipp1n = tramnmx(ipp1,minSP1,maxSP1);
ipp2n = tramnmx(ipp2,minSP2,maxSP2);
ipp3n = tramnmx(ipp3,minSP3,maxSP3);
ipp4n = tramnmx(ipp4,minSP4,maxSP4);

% Simulation
it1=sim(net1,ipp1n);
it2=sim(net2,ipp2n);
it3=sim(net3,ipp3n);
it4=sim(net4,ipp4n);

% Denormalization
it1 = postmnmx(it1,minST1,maxST1);
it2 = postmnmx(it2,minST2,maxST2);
it3 = postmnmx(it3,minST3,maxST3);
it4 = postmnmx(it4,minST4,maxST4);
```

```
    ittc1(1,1)=it1(1,1);
    ittc2(1,1)=it2(1,1);
    ittc3(1,1)=it3(1,1);
    ittc4(1,1)=it4(1,1);

    %-----------------------------------------------
    % Beginning of time stepping simulation
    %-----------------------------------------------
    for j=2:ct1
        i=j;

        % Constructing input data set
        ipp1(1:mk-1,1)=Cond(1:mk-1,i); % copying conditions
        ipp1(mg,1)=Cond(mg,i); % copying time sequence
        itt1(1,1)=it1(1,1);

        ipp2(1:mk-1,1)=Cond(1:mk-1,i);
        ipp2(mg,1)=Cond(mg,i);
        itt2(1,1)=it2(1,1);

        ipp3(1:mk-1,1)=Cond(1:mk-1,i);
        ipp3(mg,1)=Cond(mg,i);
        itt3(1,1)=it3(1,1);

        ipp4(1:mk-1,1)=Cond(1:mk-1,i);
        ipp4(mg,1)=Cond(mg,i);
        itt4(1,1)=it4(1,1);

        % replacing past time series of output
        ipp1(mk,1)=ipp1(mk,1)-itt1(1,1); % value(t=n)=value(t=n-1)-
difference
        ipp2(mk,1)=ipp2(mk,1)-itt2(1,1);
        ipp3(mk,1)=ipp3(mk,1)-itt3(1,1);
        ipp4(mk,1)=ipp4(mk,1)-itt4(1,1);

        % Normalization
        ipp1n = tramnmx(ipp1,minSP1,maxSP1);
        ipp2n = tramnmx(ipp2,minSP2,maxSP2);
        ipp3n = tramnmx(ipp3,minSP3,maxSP3);
        ipp4n = tramnmx(ipp4,minSP4,maxSP4);

        % Simulation
        it1=sim(net1,ipp1n);
        it2=sim(net2,ipp2n);
        it3=sim(net3,ipp3n);
        it4=sim(net4,ipp4n);

        % Denormalization
        it1 = postmnmx(it1,minST1,maxST1);
        it2 = postmnmx(it2,minST2,maxST2);
        it3 = postmnmx(it3,minST3,maxST3);
        it4 = postmnmx(it4,minST4,maxST4);

        % record simulation result; difference at each time
        ittc1(1,i)=it1(1,1);
        ittc2(1,i)=it2(1,1);
        ittc3(1,i)=it3(1,1);
```

```
    ittc4(1,i)=it4(1,1);

    % shift down past time series of output
    itt1(2:m-1,1)=itt1(1:m-2,1);
    itt2(2:m-1,1)=itt2(1:m-2,1);
    itt3(2:m-1,1)=itt3(1:m-2,1);
    itt4(2:m-1,1)=itt4(1:m-2,1);

end % End of time stepping simulation


% copying time series to t
t=PP1(mg,1:i);
t=t';

% Calculating cumulative results

[b c]=size(ittc1);
ittc1m(1,1)=100;
ittc1m(1,2:c+1)=100-cumsum(ittc1);
TT1=measuredTT1;

[b c]=size(ittc2);
ittc2m(1,1)=-10;
ittc2m(1,2:c+1)=-10-cumsum(ittc2);
TT2=measuredTT2;

[b c]=size(ittc3);
ittc3m(1,1)=-10;
ittc3m(1,2:c+1)=-10-cumsum(ittc3);
TT3=measuredTT3;

[b c]=size(ittc4);
ittc4m(1,1)=0;
ittc4m(1,2:c+1)=0+cumsum(ittc4);
TT4=measuredTT4;

% Transposing data for graphical representation
Tp1=TT1'; ittc1=ittc1m';
Tp2=TT2'; ittc2=ittc2m';
Tp3=TT3'; ittc3=ittc3m';
Tp4=TT4'; ittc4=ittc4m';

% Storing result of each simulation
Y_pred(counter,1)=ittc1(ct1,1);
Y_meas(counter,1)=Tp1(ct1,1);

L_pred(counter,1)=ittc2(ct1,1);
L_meas(counter,1)=Tp2(ct1,1);

DI_pred(counter,1)=ittc3(ct1,1);
DI_meas(counter,1)=Tp3(ct1,1);

DE_pred(counter,1)=ittc4(ct1,1);
DE_meas(counter,1)=Tp4(ct1,1);
```

```
    % Graphical Validation
    % Tp:FEM prediction  ittc:Network prediction
    figure
    h=plot(t,Tp1(1:i,1),'k+',t,ittc1(1:i,1),'ko');
    %title('Yield');
    xlabel('Process Time [s]');
    ylabel('Yield [%]');
    legend('FEM','GRNN',4);
    set(h,'MarkerSize',5,'LineWidth',1.5);
    grid on

    figure
    h=plot(t,Tp2(1:i,1),'k+',t,ittc2(1:i,1),'ko');
    %title('Salmonella LR');

    xlabel('Process Time [s]');
    ylabel('Log(Log Reduction of Salmonella)');
    legend('FEM','GRNN',4);
    set(h,'MarkerSize',5,'LineWidth',1.5);
    grid on

    figure
    h=plot(t,Tp3(1:i,1),'k+',t,ittc3(1:i,1),'ko');
    %title('DI');

    xlabel('Process Time [s]');
    ylabel('Log(Log Reduction of Internal color change)');
    legend('FEM','GRNN',4);
    set(h,'MarkerSize',5,'LineWidth',1.5);
    grid on

    figure
    h=plot(t,Tp4(1:i,1),'k+',t,ittc4(1:i,1),'ko');
    %title('DEavg');

    xlabel('Process Time [s]');
    ylabel('Log(Log Reduction of Crust Color Change)');
    legend('FEM','GRNN',4);
    set(h,'MarkerSize',5,'LineWidth',1.5);
    grid on

    fprintf('%d\n',counter)
end

% Exporting final results

Outputs(:,1)=Y_meas(:,1);
Outputs(:,2)=Y_pred(:,1);
Outputs(:,3)=L_meas(:,1);
Outputs(:,4)=L_pred(:,1);
Outputs(:,5)=DI_meas(:,1);
Outputs(:,6)=DI_pred(:,1);
Outputs(:,7)=DE_meas(:,1);
Outputs(:,8)=DE_pred(:,1);
```

```
fid=
fopen(['C:\MATLAB7\work\Dissertation\NN\Validation\result_final_opt.dat
'],'wt');
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs');
fclose(fid);
```

```
% Generate_SeqData_Train.m
% Reconstruct P(input) & Tg(output) with original data to train DNNM

function [rv1, rv2]=Generate_SeqData_Train(a,result)

% Input & output matrix structure
% [input output]
% input=[Td Sd Vd / Tm Sm Vm / Y L Tc Ts / Yp Lp Tcp Tsp / t ]
% output=[Yf Lf Tcf Tsf]

[r c]=size(a); % [#rows #columns]

m=2; % number of past time series
i=0;
dt=10; % [sec]

for mi=(dt*2+1):(2*dt):(r-0);% (2*dt) due to the time interval of data
set is 0.5 s
i=i+1;

    %Conditions @ t(n+1)
    %Process Temperature
    for n=1:m
        if (mi-2*dt*n)<1
            T(n)=25; % Ambient Temp. 25C
        else
            T(n)=a(mi-2*dt*n,2);
        end
    end

    %Process Humidity
    for n=1:m
        if (mi-2*dt*n)<1
            H(n)=5; % Ambient Humidity. 30%
        else
            H(n)=a(mi-2*dt*n,3);
        end
    end

    %Jet Velocity
    for n=1:m
        if (mi-2*dt*n)<1
            V(n)=0; % Ambient Jet. 0 m/s
        else
            V(n)=a(mi-2*dt*n,4);
        end
    end


    % Time data
    if (mi-2*dt*1)<= 0
        t=((mi-2*dt*1)+1)/2-1;
    else
```

```
        t=a(mi-2*dt*1,1); %a(mi-2*n,1);
    end



    %State @ t(n)

    %Output
    for n=1:m-1
        if (mi-2*dt*n)< 1 % initial state
            switch result
                case 5
                    Yp(n)=100; %initial yield
                case 7
                    Yp(n)=0;   %initial Salmonella LR
                case 8
                    Yp(n)=0;   %initial DI
                case 10
                    Yp(n)=0;   %initial DEavg
            end
        else

            dumy=a(mi-2*dt*n,result);
            Yp(n)=dumy; % 5: row# contains output variable

        end
    end


dumy=a(mi,result); % 5: row# contains output variable
Yf=dumy;


% Constructing input and output set
k=i; %k=i-m; %(m-1);

% Input data set
P(1:m,k)=T(1:m);              % Temperature series
P(m+1:2*m,k)=H(1:m);          % Humidity series
P(2*m+1:3*m,k)=V(1:m);        % Velocity series
P(3*m+1:4*m-1,k)=Yp(1:m-1);   % Output series
P(4*m,k)=t;                   % Linear Data series
% P(4*m:5*m-1,k)=t(1:m);

% T matrix
Tg(1,k)=Yf; % State @ t(n+1)

end

% Return variables
rv1=P;
rv2=Tg;
```

```
% Generate_SeqData_Test.m
% Reconstruct G(input)with test data to simulate DNNM

function rv1=Generate_SeqData_Test(a)

[r c]=size(a); % [#rows #columns]

m=2; % number of past time series
i=0;
dt=10; % [sec]

% Tailoring original condition data set
T1=a(1:dt*2:r,2); [i j]=size(T1); T1(i+1,1)=a(r,2);
H1=a(1:dt*2:r,3); [i j]=size(H1); H1(i+1,1)=a(r,3);
V1=a(1:dt*2:r,4); [i j]=size(V1); V1(i+1,1)=a(r,4);
t=a(1:dt*2:r,1);  [i j]=size(t); t(i+1,1)=a(r,1);

% Transposing
T1=T1'; H1=H1'; V1=V1'; t=t';

[nr nc]=size(T1);

% Assigning intial values
T2(1,1)=25; T2(1,2:nc)=T1(1,1:nc-1);
H2(1,1)=25; H2(1,2:nc)=H1(1,1:nc-1);
V2(1,1)=25; V2(1,2:nc)=V1(1,1:nc-1);

% Merging into a single matrix
G(1,:)=T1;
G(2,:)=T2;
G(3,:)=H1;
G(4,:)=H2;
G(5,:)=V1;
G(6,:)=V2;
G(8,:)=t;

% Return value
rv1=G;
```

Static Neural Network Model (SNNM)

```
% Train_SNNM.m
% This program is to train SNNM with random profile

clear all

%================================================================
%    TRAINING
%================================================================

% Reading files
fprintf('***************\n');
fprintf('Training Phase\n')

ct=0; % counter

for i=1:1000

    % Reading Fourier parameters
    fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\D2\Condition
Para\Parameter_',num2str(i),'.dat']);
    h=fscanf(fid,'%f',[1, inf]); % [temp(1-21) steam(22-42)
velocity(43-63) time(64) yield SamonellaLR DI DEavg RMSE(T) RMSE(H)
RMSE(V) ]
    h=h';
    fclose(fid);
    % Reading results
    fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\D2\FE Solution
Files\FEsolution_',num2str(i),'.dat']);
    g=fscanf(fid,'%f %f %f %f %f %f %e %e %e %e %f %f',[12, inf]); %
[temp(1-21) steam(22-42) velocity(43-63) time(64) yield SamonellaLR DI
DEavg RMSE(T) RMSE(H) RMSE(V) ]
    g=g';
    fclose(fid);
    [rn cn]=size(g);


        % Rearranging data structure
        ct=ct+1;
        P(1:64,ct)=h(1:64,1); % input training

        % Yield
        T(1,ct)=g(rn,5); % output training

        % lethality
        if g(rn,7)<0.0000000001
            T(2,ct)=-10;
        else
            T(2,ct)=log10(g(rn,7)); % output training
        end

        % DI (interanal color change)
        if g(rn,8)<0.000001
            T(3,ct)=-6;
```

```
        else
            T(3,ct)=log10(g(rn,8)); % output training
        end

        % DE (surface color change)
        T(4,ct)=g(rn,10);

end

% Normalization
[Pn,minP,maxP,Tn,minT,maxT] = premnmx(P,T);


% Network training: feed-forward neural network
net = newff([minP maxP],[40 20 4],{'tansig' 'tansig'
'purelin'},'traincgp'); %40 20
net.trainParam.epochs = 1500;
net.trainParam.show=10;
net = train(net,Pn,Tn);

fprintf('\n>End of Training\n')

% Saving trained network & parameters
save('C:\MATLAB7\work\Dissertation\NN\SNNM_R\net','net');

Outputs(1:64,1)=minP;
Outputs(1:64,2)=maxP;
Outputs(65:68,1)=minT;
Outputs(65:68,2)=maxT;

fid= fopen('C:\MATLAB7\work\Dissertation\NN\SNNM_R\net_para.dat','wt');
fprintf(fid,'%10.8f %10.8f\n',Outputs');
fclose(fid);


%=============================================================
%    SIMULATION
%=============================================================

% Testing
ct=0;
clear h g Pt Tt Outputs
nf=0;

fprintf('----------------\n')
fprintf('Simulation Phase\n')

% Retrieving stored network and parameters
clear net
load('C:\MATLAB7\work\Dissertation\NN\SNNM_R\net','net');

fid=fopen('C:\MATLAB7\work\Dissertation\NN\SNNM_R\net_para.dat');
a=fscanf(fid,'%f %f',[2, inf]); % [time temp steam velocity yield VavgM
SamonellaLR DI DE4 DEavg Tc Ts]
a=a';
fclose(fid);
```

```
minP=a(1:64,1);
maxP=a(1:64,2);
minT=a(65:68,1);
maxT=a(65:68,2);

for i=1:1000
nf=nf+1;

    % Reading Fourier parameters

fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\D6\Para\Parameter_',
num2str(i),'.dat']);
    h=fscanf(fid,'%f',[1, inf]); % [temp(1-21) steam(22-42)
velocity(43-63) time(64) yield SamonellaLR DI DEavg RMSE(T) RMSE(H)
RMSE(V) ]
    h=h';
    fclose(fid);
    % Reading results

fid=fopen(['C:\MATLAB7\work\Dissertation\Data_Pool\D6\Test_Fourier',num
2str(i),'.dat']);
    g=fscanf(fid,'%f %f %f %f %f %f %e %e %e %e %f %f',[12, inf]); %
[temp(1-21) steam(22-42) velocity(43-63) time(64) yield SamonellaLR DI
DEavg RMSE(T) RMSE(H) RMSE(V) ]
    g=g';
    fclose(fid);
    [rn cn]=size(g);


        % Rearranging data structure
        ct=ct+1;
        Pt(1:64,ct)=h(1:64,1); % input training

        % Yield
        Tt(1,ct)=g(rn,5); % output training

        % lethality
        if g(rn,7)<0.0000000001
            Tt(2,ct)=-10;
        else
            Tt(2,ct)=log10(g(rn,7)); % output training
        end

        % DI
        if g(rn,8)<0.000001
            Tt(3,ct)=-6;
        else
            Tt(3,ct)=log10(g(rn,8)); % output training
        end

        % DE
        Tt(4,ct)=g(rn,10);


end

% Normalization
```

```
[Ptn] = tramnmx(Pt,minP,maxP);

% simulation
Tsimn=sim(net,Ptn);
Tsim=postmnmx(Tsimn,minT,maxT);


%===============================================================
% Graphical validation
%===============================================================
figure
plot(Tt(1,:),Tsim(1,:),'.r');
title('Yield')
grid on
RMSE_Yield=RMSE(Tt(1,:)',Tsim(1,:)',0);


figure
plot(Tt(2,:),Tsim(2,:),'.r');
title('Salmonella LR')
grid on
RMSE_LR=RMSE(Tt(2,:)',Tsim(2,:)',0);

figure
plot(Tt(3,:),Tsim(3,:),'.r');
title('DI')
grid on
RMSE_DI=RMSE(Tt(3,:)',Tsim(3,:)',0);

figure
plot(Tt(4,:),Tsim(4,:),'.r');
title('DEavg')
grid on
RMSE_DEavg=RMSE(Tt(4,:)',Tsim(4,:)',0);

fprintf('%6.4f\n%6.4f\n%6.4f\n%6.4f\n',RMSE_Yield,RMSE_LR,RMSE_DI,RMSE_
DEavg);


%===============================================================
% Exporting
%===============================================================

% Changing variable name
F1(:,1)=Tt(1,:)'; F1(:,2)=Tsim(1,:)';
F2(:,1)=Tt(2,:)'; F2(:,2)=Tsim(2,:)';
F3(:,1)=Tt(3,:)'; F3(:,2)=Tsim(3,:)';
F4(:,1)=Tt(4,:)'; F4(:,2)=Tsim(4,:)';

% Exporting final results
Outputs(:,1)=F1(:,1);
Outputs(:,2)=F1(:,2);
Outputs(:,3)=F2(:,1);
Outputs(:,4)=F2(:,2);
Outputs(:,5)=F3(:,1);
Outputs(:,6)=F3(:,2);
Outputs(:,7)=F4(:,1);
```

```
Outputs(:,8)=F4(:,2);

Outputs=Outputs';

fid= fopen(['C:\MATLAB7\work\Dissertation\NN\SNNM_R\RMSE.dat'],'wt');
fprintf(fid,'RMSE(Y)= %10.4f\n',RMSE_Yield);
fprintf(fid,'RMSE(L)= %10.4f\n',RMSE_LR);
fprintf(fid,'RMSE(DI)= %10.4f\n',RMSE_DI);
fprintf(fid,'RMSE(DE)= %10.4f\n',RMSE_DEavg);
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs);
fclose(fid);

clear Outputs

%=========================================================================
% Classification & Exporting
%=========================================================================

% Initial values
c1=0;c2=0;c3=0;c4=0;
c5=0;c6=0;c7=0;c8=0;
Outputs_aa=[0; 0; 0; 0; 0; 0; 0; 0;];
Outputs_rr=[0; 0; 0; 0; 0; 0; 0; 0;];
Outputs_typeI=[0; 0; 0; 0; 0; 0; 0; 0;];
Outputs_typeII=[0; 0; 0; 0; 0; 0; 0; 0;];
index_aa=0; index_rr=0; index_typeI=0; index_typeII=0;

for i=1:1000

    % Rate of the accepted for the accepted
    if
(F2(i,1)>=0.813)&&(F3(i,1)>=0.146)&&(F4(i,1)>=0.26)&&(F4(i,1)<=0.497)%
accpeted patties among the measured
        c1=c1+1;
        if
(F2(i,2)>=0.813)&&(F3(i,2)>=0.146)&&(F4(i,2)>=0.26)&&(F4(i,2)<=0.497)
            c2=c2+1;
            Outputs_aa(c2,1)=F1(i,1);
            Outputs_aa(c2,2)=F1(i,2);
            Outputs_aa(c2,3)=F2(i,1);
            Outputs_aa(c2,4)=F2(i,2);
            Outputs_aa(c2,5)=F3(i,1);
            Outputs_aa(c2,6)=F3(i,2);
            Outputs_aa(c2,7)=F4(i,1);
            Outputs_aa(c2,8)=F4(i,2);
            index_aa(c2,1)=i;
        end
    end


    % Rate of the rejected for the rejected
    if
(F2(i,1)<0.813)||(F3(i,1)<0.146)||(F4(i,1)<0.26)||(F4(i,1)>0.497)%
rejected patties among the measured
        c3=c3+1;
```

```
        if
(F2(i,2)<0.813)||(F3(i,2)<0.146)||(F4(i,2)<0.26)||(F4(i,2)>0.497)
            c4=c4+1;
            Outputs_rr(c4,1)=F1(i,1);
            Outputs_rr(c4,2)=F1(i,2);
            Outputs_rr(c4,3)=F2(i,1);
            Outputs_rr(c4,4)=F2(i,2);
            Outputs_rr(c4,5)=F3(i,1);
            Outputs_rr(c4,6)=F3(i,2);
            Outputs_rr(c4,7)=F4(i,1);
            Outputs_rr(c4,8)=F4(i,2);
            index_rr(c4,1)=i;
        end
    end


    % Rate of Type I error
    if
(F2(i,1)>=0.813)&&(F3(i,1)>=0.146)&&(F4(i,1)>=0.26)&&(F4(i,1)<=0.497)%
accpeted patties among the measured
        c5=c5+1;
        if
(F2(i,2)<0.813)||(F3(i,2)<0.146)||(F4(i,2)<0.26)||(F4(i,2)>0.497)
            c6=c6+1;
            Outputs_typeI(c6,1)=F1(i,1);
            Outputs_typeI(c6,2)=F1(i,2);
            Outputs_typeI(c6,3)=F2(i,1);
            Outputs_typeI(c6,4)=F2(i,2);
            Outputs_typeI(c6,5)=F3(i,1);
            Outputs_typeI(c6,6)=F3(i,2);
            Outputs_typeI(c6,7)=F4(i,1);
            Outputs_typeI(c6,8)=F4(i,2);
            index_typeI(c6,1)=i;
        end
    end


    % Rate of False-pass classification error
    if
(F2(i,1)<0.813)||(F3(i,1)<0.146)||(F4(i,1)<0.26)||(F4(i,1)>0.497)%
rejected patties among the measured
        c7=c7+1;
        if
(F2(i,2)>=0.813)&&(F3(i,2)>=0.146)&&(F4(i,2)>=0.26)&&(F4(i,2)<=0.497)
            c8=c8+1;
            Outputs_typeII(c8,1)=F1(i,1);
            Outputs_typeII(c8,2)=F1(i,2);
            Outputs_typeII(c8,3)=F2(i,1);
            Outputs_typeII(c8,4)=F2(i,2);
            Outputs_typeII(c8,5)=F3(i,1);
            Outputs_typeII(c8,6)=F3(i,2);
            Outputs_typeII(c8,7)=F4(i,1);
            Outputs_typeII(c8,8)=F4(i,2);
            index_typeII(c8,1)=i;
        end
    end
```

```
end

R_aa=c2/c1;
R_rr=c4/c3;
R_typeI=c6/c5;
R_typeII=c8/c7;


% Exporting Classification Results
fid= fopen(['C:\MATLAB7\work\Dissertation\NN\SNNM_R\DA_aa.dat'],'wt');
fprintf(fid,'R_aa= %10.4f=%d/%d\n',R_aa,c2,c1);
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs_aa');
fprintf(fid,'%d\n',index_aa');
fclose(fid);

fid= fopen(['C:\MATLAB7\work\Dissertation\NN\SNNM_R\DA_rr.dat'],'wt');
fprintf(fid,'R_rr= %10.4f=%d/%d\n',R_rr,c4,c3);
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs_rr');
fprintf(fid,'%d\n',index_rr');
fclose(fid);

fid=
fopen(['C:\MATLAB7\work\Dissertation\NN\SNNM_R\DA_typeI.dat'],'wt');
fprintf(fid,'R_typeI= %10.4f=%d/%d\n',R_typeI,c6,c5);
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs_typeI');
fprintf(fid,'%d\n',index_typeI');
fclose(fid);

fid=
fopen(['C:\MATLAB7\work\Dissertation\NN\SNNM_R\DA_typeII.dat'],'wt');
fprintf(fid,'R_typeII= %10.4f=%d/%d\n',R_typeII,c8,c7);
fprintf(fid,'%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f
\n',Outputs_typeII');
fprintf(fid,'%d\n',index_typeII');
fclose(fid);
```

Genetic Algorithm (GA)

This case was coupled with FEM and PLI (piecewise linear interpolation) parameterizaiton method.

GA_FEM_PLI.m has following sub-programs. Some sub-programs for genetic operations (crossover and mutation) were adopted from the work of Venkataraman (2002) and modified.

- SimpleCrossover.m
- ArithmeticCrossover.m
- Mutation_PLI.m
- Populator_PLI.m
- CheckRange_PLI.m
- GenProfile_PLI.m
- fun_FEM_PLI.m
    - FunGenFEM_PLI.m
    - Model_FEM (Finite element model was not included. Refer the model developed by Watkins (2004).

✓ fun_DNNM_FS.m :

This sub-program can be included when the GA is coupled with DNNM and Fourier series parameterization method while maintaing framworks of GA_FEM_PLI.m.

- FourierFunGen.m (included DNNM package)
- Generate_SeqData_Test.m (included in DNNM package)

```
% GA_FEM_PLI.m
% Genetic algorithm coupled with FEM and piecewise linear interpolation
% parameterization (PLI) were used.

clear all
global Fbest_trace Xbest_trace Mu_count

% GA parameters
fprintf('GA Optimization Started!\n')

nDes=64;     % # of design parameters [duration temp moisture velocity]
nG=200;      % total number of  generation
nPi=10;      % # of initial population
nP=4;        % # of population for each generation
nSC=3;           % number of simple crossover - each yields 2 children
nAC=3;       % number of arithmetic crossover - each yields 2 children
nIM=4;           % number of immigrants
nMU=2;       % number of mutation


% Generating initial population
fprintf('Generating initial population\n')

% Getting initial population information from a file
for i=1:nPi


fid=fopen(['/home/jeongsa1/Matlab/1/IP/IDsmooth_',num2str(i),'.dat']);
    g=fscanf(fid,'%g',[1, inf]); % [temp(1-21) steam(22-42)
velocity(43-63) time(64) yield SamonellaLR DI DEavg RMSE(T) RMSE(H)
RMSE(V) ]
    g=g';
    fclose(fid);

    vector=g(1:64,1);
    status=1; % 1=contraints satisfied; 0=not satisfied
    results=g(65:68,1); % predicted yield, Salmonella reduction, DI, DE

    ind(i,:)=vector';
    ind_Y(i,1)=results(1,1); % yield

    ind_status(i,1)=status;
    ind_results(i,:)=results';      % Evaluation of individual: Yield

%        ind_L(i,1)=Lf; %fun_nn_1stg_nlec(vector);     % Evaluation of
individual: Lethality
fprintf('%d  ',i)
end
fprintf('\n')

% If you want to generate an initial pop., unmark following marked code
and
% mark the above code.
```

```
% % Generating initial population
% ct=0;
% for i=1:nPi
%     status=0; % 0:constaints are not satisfied, 1:satisfied
%     while status==0
%         vector=Populator_PLI(nDes);    % Generating individuals
having target lethality
%         [status, results]=fun_FEM_PLI(vector);    % Calculating
result with GRNN and vector
%
%     end
%     ind(i,:)=vector'; % transposed
%     ind_Y(i,1)=results(1,1); %100-fun_nn_1stg(vector);      %
Evaluation of individual: Yield
%
%     ind_status(i,1)=status;
%     ind_results(i,:)=results';      % Evaluation of individual: Yield
%
%     passedCond(1:nDes,1)=vector;
%     passedCond(nDes+1:nDes+4,1)=results;
%     ct=ct+1;
%
fid=fopen(['C:\MATLAB7\work\Dissertation\Optimizer\InitialPopulation\ID
_',num2str(ct),'.dat'],'wt');
%     fprintf(fid,'%f\n',passedCond');
%     fclose(fid);
%
%     fprintf('%d  ',i)
% end
% fprintf('\n')

% Sorting to select parents
[value, index]=sort(ind_Y,'descend');

% Selecting parents
for i=1:nP
    Xgen(i,:)=ind(index(i),:);

    Xgen_status(i,1)=ind_status(index(i),:);
    Xgen_results(i,:)=ind_results(index(i),:);

    F_Origin(i,1)=ind_results(index(i),1);
    X_Origin(i,:)=ind(index(i),:);
end
clear index

*****************************************************************
% Generating offsprings and immigrants
for iG=1:nG

fprintf('***\n')
fprintf('Generation %d\n',iG)

% Simple crossover
    [XChild] = SimpleCrossover(nDes, nP,nSC,Xgen);
      for i=1:2*nSC
```

```
      Xgen(nP+i,:) = XChild(i,:);
      end
      fprintf('\nFinished: Simple Crossover')

% Arithmetic crossover
      [XAChild] = ArithmeticCrossover(nDes, nP,nAC,Xgen);
      for i=1:2*nAC
      Xgen(nP+2*nSC+i,:) = XAChild(i,:);
      end
      fprintf('\nFinished: Arithmetic Crossover')

% Mutation
      [n1 m1] = size(Xgen);
      [XMu]= Mutation_PLI(nDes,Xgen);% mutate all members

      for i=1:nMU % nMU mutated member selected among all muatated
population
      Xgen(nP+2*nSC+2*nAC+i,:) = XMu(i,:);
      end
      fprintf('\nFinished: Mutation')

% Immigration

    [n2 m2] = size(Xgen);

    for j = 1:nIM
        Vect = Populator_PLI(nDes);
        Xgen(n2+j,:) = Vect';
    end
    fprintf('\nFinished: Immigration\n')

% Checking the validity of parameters: testing if the actual control
% profiles are in the proper ranges
j=0;
clear tempXgen tempXgen_status tempXgen_results
[n4 m4] = size(Xgen);
for i=1:n4
    s(i,1)=CheckRange_PLI(Xgen(i,:)');
    if s(i,1)==1
        j=j+1;
        tempXgen(j,:)=Xgen(i,:);
        if (i<=nP)
            tempXgen_status(j,:)=Xgen_status(i,1);
            tempXgen_results(j,:)=Xgen_results(i,:);
        end
    end
end

clear Xgen
Xgen=tempXgen;
Xgen_status=tempXgen_status;
Xgen_results=tempXgen_results;
fprintf('Finished: Range check: ')
[n3 m3] = size(Xgen);
fprintf('Rejection rate(range)=%d/%d\n',(n4-n3),n4)
```

```
% Evaluation
    fprintf('Evaluation process\n')
    for i = (nP+1):n3
        vector = Xgen(i,:);
        [status, results]=fun_FEM_PLI(vector');

        Xgen_status(i,1)=status;
        Xgen_results(i,:)=results';
    end


% Inspecting constraints satisfaction
    kk=0;
    for i=1:n3
        if (Xgen_status(i,1)==1)&(isnan(Xgen_results(1,1))==0)
            kk=kk+1;
            passedXgen(kk,:)=Xgen(i,:);
            passedXgen_results(kk,:)=Xgen_results(i,:);
            passedXgen_status(kk,:)=Xgen_status(i,1);
        end
    end

    if kk<2
        fprintf('\nSpieces terminated\n')
        break;
    end

% Sorting
   [value, index]=sort(passedXgen_results(:,1),'descend');

   clear temp temp_status temp_results

   % Temporary storage
   temp=passedXgen;
   temp_status=passedXgen_status;
   temp_results=passedXgen_results;

   fprintf('Number of passed generation= %d',kk)
   passedXgen_results(index,:);
   fprintf('Pass ratio= %d/%d\n',kk,n3)

   Xbest=passedXgen(index(1),:); % best condition at each generation
   Fbest=passedXgen_results(index(1),:); % best yield at each
generation
   Xbest_trace(iG,:)=Xbest; % progressive history
   Fbest_trace(iG,:)=Fbest; % progressive history

   clear Xgen Xgen_status Xgen_results XChild XAChild XMu passedXgen
passedXgen_results passedXgen_status


   % Selecting parents: Same vector will be skipped for diversity
   for i=1:nP
       Xgen(i,:)=temp(index(i),:);
       Xgen_status(i,1)=temp_status(index(i),1);
       Xgen_results(i,:)=temp_results(index(i),:);
   end
```

```
   clear index
   Xgen_results

   % Saving workspace at every 50 iteration
   if(mod(iG,50)==0)

save(['C:\MATLAB7\work\Dissertation\Packages\1\1\WorkSpace_',num2str(iG
),'.mat']);
   end

end % End of cycle of a generation
%-----------------------------------------------------------
% Plotting & Exporting
%-----------------------------------------------------------
fprintf('Plotting & Exporting\n')
[n m] = size(Xbest_trace);

% Plotting the best profiles
p=FunGenFEM_PLI(Xbest_trace(n,:)');
time=p(:,1);
temp=p(:,2);
hum=p(:,3);
vel=p(:,4);

figure
h=plot(time,temp,'r-',time,hum,'b--',time,vel,'m-.');
xlabel('Process time [s]');
ylabel('Temperature [deg. C] / Humidity [%Mv] / Velocity [m/s]');
legend('Temperature','Humidity','Velocity',4);
set(h,'LineWidth',1.5);

tot=nP+2*nSC+2*nAC+Mu_count+nIM
SimpleCrossover_rate=(2*nSC)/tot % each crossover produce two children
ArithmaticCrossover_rate=(2*nAC)/tot
Mutation_rate=Mu_count/tot
Immigration_rate=nIM/tot


% Exporting result
% History of generations

fid=fopen(['C:\MATLAB7\work\Dissertation\Packages\1\1\Result_generation
s.dat'],'wt');
for i=1:nG
    for j=1:m
        fprintf(fid,'%10.4f',Xbest_trace(i,j));
    end
    fprintf(fid,'%10.4f %10.4f %10.4f %10.4f\n',Fbest_trace(i,:));
end

fclose(fid);

% Reporting GA summary
fid3=
fopen(['C:\MATLAB7\work\Dissertation\Packages\1\1\Result_summary.dat'],
'wt');
```

```
fprintf(fid3,'-------------------------------------------------------
\n');
fprintf(fid3,' SUMMARY OF GA-FEM OPTIMIZATION\n');
fprintf(fid3,' >Discretized profile for each control vector\n');
fprintf(fid3,' >GA-FEM found the best combination of 64
coefficients\n');
fprintf(fid3,'-------------------------------------------------------
\n');
fprintf(fid3,'Yield_max= %10.4f [%%]
\nlog(LR.microb)= %10.4f\nlog(DI)= %10.4f\nlog(DE)= %10.4f\n\n',Fbest);
fprintf(fid3,'[Optimization Parameters]\n');
fprintf(fid3,'1. Design variables:      %d\n',nDes);
fprintf(fid3,'2. Initial population:    %d\n',nPi);
fprintf(fid3,'3. Parents:               %d\n',nP);
fprintf(fid3,'4. Generation cycle:      %d\n',nG);
fprintf(fid3,'5. Simple crossover:      %d\n',nSC);
fprintf(fid3,'6. Arithmatic crossover:  %d\n',nAC);
fprintf(fid3,'7. Immigration:           %d\n',nIM);
fprintf(fid3,'8. Population/genertion:  %d\n\n',tot);
fprintf(fid3,'9. Simple crossover
rate:      %6.2f\n',SimpleCrossover_rate);
fprintf(fid3,'10. Arithmatic crossover
rate: %6.2f\n',ArithmaticCrossover_rate);
fprintf(fid3,'11. Mutation rate:             %6.2f\n',Mutation_rate);
fprintf(fid3,'12. Immigaration
rate:         %6.2f\n',Immigration_rate);
fprintf(fid3,'-------------------------------------------------------
\n');

fclose(fid3);
```

```
%     creating children by simple crossover

function ret = SimpleCrossover(nDes,nP,nSC,X)

XP1 = X(1,:);
XP2 = X(2,:);

for k=1:nSC
    r = rand(1,1);

    if r > 0.5
        r1 = floor(nDes*r);
    else
        r1 = ceil(nDes*r);
    end

    for i = 1:1:length(XP1)
        if i < r1
            XC1(i) = XP1(i);
            XC2(i) = XP2(i);
        else
            XC1(i)= XP2(i);
            XC2(i)= XP1(i);
        end
    end

    temp=2*(k-1)+1;
    C(temp,:)=XC1;
    C(temp+1,:)=XC2;

end

ret =C;
```

```
function ret = ArithmeticCrossover(nDes,nP,nAC,X)

XP1=X(1,:);
XP2=X(2,:);

for k=1:nAC
    r1 = rand(1,1);
    r2 = 1 - r1;

    for i = 1:nDes
        XC1(1,i) = r1*XP1(1,i) + r2*XP2(1,i);
        XC2(1,i) = r2*XP1(1,i) + r1*XP2(1,i);
    end
temp=2*(k-1)+1;
C(temp,:)=XC1;
C(temp+1,:)=XC2;

end

ret =C;
```

```
% Mutating all the vectors by changing a single element

function ret = Mutation_PLI(nDes,X)
global xv1 yv1 xv2 yv2 Mu_count

Mu_count=0; % mutation counter

XX = X;
[n m] =size(X);


for i = 1:n
    r = rand(1,1);
    if r > 0.5
        r1 = floor(nDes*r);
    else
        r1 = ceil(nDes*r);
    end

    for j = 1:m
        if(j == r1)
            x=Populator_PLI(nDes);
            XX(i,j) = x(j);
            Mu_count=Mu_count+1;
        end
    end
end

ret = XX;
```

```
% Generating control parameters within feasible ranges

function rv2=Populator_PLI(nDes)

% generates a random control vector

n=(nDes-1)/3; % w/o duration; 21 points for each control vector
t_L=60;  t_U=600;  % time bounds [s]; L=lower  U=upper
T_L=100; T_U=250;  % Temperature bounds [C]
H_L=0;  H_U=100;   % Steam content bounds [% by volume]
V_L=0;  V_U=30;    % Impingement velocity bounds [m/s]

% generating random profiles
x(1:n,1)=T_L+(T_U-T_L)*rand(n,1);           % Temperature
x(n+1:2*n,1)=H_L+(H_U-H_L)*rand(n,1);       % Humidity
x(2*n+1:3*n,1)=V_L+(V_U-V_L)*rand(n,1);     % Velocity
x(3*n+1,1)=round(t_L+(t_U-t_L)*rand(1,1)); % duration

rv2=x;
```

```
% Check if the control profiles are within the upper and lower bounds.

function status=CheckRange_PLI(X)

% Temperature, humidity, and velocity bounds
Bounds=[100 250; 0 100; 0 30]; %[TL TU; HL HU; VL VU]

[r c]=size(X);

duration=X(r,1);
P=FunGenFEM_PLI(X);
timestep=0.5;

T(:,1)=P(:,2);
H(:,1)=P(:,3);
V(:,1)=P(:,4);

% Check violations
if ((max(T) <= Bounds(1,2)) && (min(T) >= Bounds(1,1)))
    S1=1;
else
    S1=0;
end

if ((max(H) <= Bounds(2,2)) && (min(H) >= Bounds(2,1)))
    S2=1;
else
    S2=0;
end

if ((max(V) <= Bounds(3,2)) && (min(V) >= Bounds(3,1)))
    S3=1;
else
    S3=0;
end

status = S1 * S2 * S3; % 1=passed, 0=not passed
```

```
% Calculating actual time-condition data with parameters

function rv1 = GenProfile_PLI(v) %  v=vector

% Spliting parameters for each control vector
T(1:21,1)=v(1:21,1); % Temperature
M(1:21,1)=v(22:42,1); % Humidity
V(1:21,1)=v(43:63,1); % Velocity
duration=v(64,1);

t=0:(duration/(21-1)):duration;

% Generating 0.5 sec time intervals
ti=0:0.5:duration;

t=t';
ti=ti';

% Interpolation with parameters
Ti = interp1(t,T,ti);
Mi = interp1(t,M,ti);
Vi = interp1(t,V,ti);

% Merging time-condition data into a matrix
Cond(:,1)=ti;
Cond(:,2)=Ti;
Cond(:,3)=Mi;
Cond(:,4)=Vi;

% return value
rv1=Cond;
```

```
% Process evaluation with FEM and PLI

function [status, results]=fun_FEM_PLI(v) % nlc: nonlinear constraints


% Getting function value at each time
C=FunGenFEM_PLI(v);
duration=v(64,1);
Te=C(:,2);
Se=C(:,3);
Ve=C(:,4);
dt=0.5;

% FEM Solution
[rv1 rv2 rv3 rv4 rv5 rv6 rv7 rv8]=Modle_FEM(duration,Te,Se,Ve,dt);

[nr nc]=size(rv1);

results(1,1)=rv1(nr,1); % Yield
results(2,1)=rv3(nr,1); % Lethality
results(3,1)=rv4(nr,1); % DI
results(4,1)=rv6(nr,1); % DE

Lf=results(2,1);
DIf=results(3,1);
DEf=results(4,1);

if (Lf >= 6.5)
    Det_Lf=1; % passed
else
    Det_Lf=0; % failed
end

if (DIf >= 1.4)
    Det_DIf=1; % passed
else
    Det_DIf=0; % failed
end

if (DEf >= 0.263)&& (DEf <= 0.497)
    Det_DEf=1; % passed
else
    Det_DEf=0; % failed
end

% 1: feasible; 2: infeasible
status=Det_Lf*Det_DIf*Det_DEf;
```

```
% Calculating actual time-condition data with parameters

function rv1 = FunGenFEM_PLI(v) %  v=vector

% Spliting parameters for each control vector
T(1:21,1)=v(1:21,1); % Temperature
M(1:21,1)=v(22:42,1); % Humidity
V(1:21,1)=v(43:63,1); % Velocity
duration=v(64,1);

t=0:(duration/(21-1)):duration;

% Generating 0.5 sec time intervals
ti=0:0.5:duration;

t=t';
ti=ti';

% Interpolation with parameters
Ti = interp1(t,T,ti);
Mi = interp1(t,M,ti);
Vi = interp1(t,V,ti);

% Merging time-condition data into a matrix
Cond(:,1)=ti;
Cond(:,2)=Ti;
Cond(:,3)=Mi;
Cond(:,4)=Vi;

% return value
rv1=Cond;
```

```
% Dynamic neural network model and Fourier parameter inputs

function [status, results]=fun_DNNM_FS(v)

% Retrieving stored networks and parameters
clear net
load('M:\Matlab\NNs\net1','net1');
load('M:\Matlab\NNs\net2','net2');
load('M:\Matlab\NNs\net3','net3');
load('M:\Matlab\NNs\net4','net4');

% Open scaling parameters
fid=fopen('M:\Matlab\NNs\normal_para.dat');
a=fscanf(fid,'%f %f %f %f %f %f %f %f',[8, inf]); % [time temp steam
velocity yield VavgM SamonellaLR DI DE4 DEavg Tc Ts]
a=a';
fclose(fid);

% Assigning scaling parameters
minSP1=a(1:8,1); minSP2=a(1:8,3); minSP3=a(1:8,5); minSP4=a(1:8,7);
maxSP1=a(1:8,2); maxSP2=a(1:8,4); maxSP3=a(1:8,6); maxSP4=a(1:8,8);
minST1=a(9,1); minST2=a(9,3); minST3=a(9,5); minST4=a(9,7);
maxST1=a(9,2); maxST2=a(9,4); maxST3=a(9,6); maxST4=a(9,8);

m=2;        % number of system parameter (length of past time series)
mg=3*m+1+1; % total row number of training matrix
mk=3*m+1;   % row location of target output value at training matrix


%==============================
%   SIMULATION PHASE
%==============================

clear Outputs
clear Y_pred L_pred DI_pred DE_pred t
clear b Cond ittc1 ittc2 ittc3 ittc4
clear itt1 itt2 itt3 itt4

% Getting profiles
C=FourierFunGen(v);

% Rearranging input data for DNNM
Cond=Generate_SeqData_Test(C);

[rt1,ct1]=size(Cond);

% Setting initial conditions
ipp1(1:mg,1)=Cond(1:mg,1); % copy
ipp1(mk,1)=100;  % 100
itt1(1:m-1,1)=0; % initial difference is zero

ipp2(1:mg,1)=Cond(1:mg,1);
ipp2(mk,1)=-10; % which is almost zero
```

```
itt2(1:m-1,1)=0;

ipp3(1:mg,1)=Cond(1:mg,1);
ipp3(mk,1)=-10; % 0
itt3(1:m-1,1)=0;

ipp4(1:mg,1)=Cond(1:mg,1);
ipp4(mk,1)=0;
itt4(1:m-1,1)=0;


% Initial simulation

% Transform data using a precalculated minimum and maximum value
ipp1n = tramnmx(ipp1,minSP1,maxSP1);
ipp2n = tramnmx(ipp2,minSP2,maxSP2);
ipp3n = tramnmx(ipp3,minSP3,maxSP3);
ipp4n = tramnmx(ipp4,minSP4,maxSP4);

% Simulation
it1=sim(net1,ipp1n); %ittc1(1,1)=it1(1,1);
it2=sim(net2,ipp2n); %ittc2(1,1)=it2(1,1);
it3=sim(net3,ipp3n); %ittc3(1,1)=it3(1,1);
it4=sim(net4,ipp4n); %ittc4(1,1)=it4(1,1);

% Denormalization
it1 = postmnmx(it1,minST1,maxST1);
it2 = postmnmx(it2,minST2,maxST2);
it3 = postmnmx(it3,minST3,maxST3);
it4 = postmnmx(it4,minST4,maxST4);

ittc1(1,1)=it1(1,1);
ittc2(1,1)=it2(1,1);
ittc3(1,1)=it3(1,1);
ittc4(1,1)=it4(1,1);

%-----------------------------------------------
% Beginning of time stepping simulation
%-----------------------------------------------
for j=2:ct1
i=j;

% Constructing input data set
ipp1(1:mk-1,1)=Cond(1:mk-1,i); % copying conditions
ipp1(mg,1)=Cond(mg,i); % copying time
itt1(1,1)=it1(1,1);

ipp2(1:mk-1,1)=Cond(1:mk-1,i);
ipp2(mg,1)=Cond(mg,i);
itt2(1,1)=it2(1,1);

ipp3(1:mk-1,1)=Cond(1:mk-1,i);
ipp3(mg,1)=Cond(mg,i);
itt3(1,1)=it3(1,1);

ipp4(1:mk-1,1)=Cond(1:mk-1,i);
ipp4(mg,1)=Cond(mg,i);
```

```
itt4(1,1)=it4(1,1);

% replacing past time series of output
ipp1(mk,1)=ipp1(mk,1)-itt1(1,1); % value(t=n)=value(t=n-1)-difference
ipp2(mk,1)=ipp2(mk,1)-itt2(1,1);
ipp3(mk,1)=ipp3(mk,1)-itt3(1,1);
ipp4(mk,1)=ipp4(mk,1)-itt4(1,1);

% Normalization
ipp1n = tramnmx(ipp1,minSP1,maxSP1);
ipp2n = tramnmx(ipp2,minSP2,maxSP2);
ipp3n = tramnmx(ipp3,minSP3,maxSP3);
ipp4n = tramnmx(ipp4,minSP4,maxSP4);

% Simulation
it1=sim(net1,ipp1n); % yield
it2=sim(net2,ipp2n); % Salmonella inactivation
it3=sim(net3,ipp3n); % Internal color change
it4=sim(net4,ipp4n); % Surface color change

% Denormalization
it1 = postmnmx(it1,minST1,maxST1);
it2 = postmnmx(it2,minST2,maxST2);
it3 = postmnmx(it3,minST3,maxST3);
it4 = postmnmx(it4,minST4,maxST4);

% record simulation result; difference at each time
ittc1(1,i)=it1(1,1);
ittc2(1,i)=it2(1,1);
ittc3(1,i)=it3(1,1);
ittc4(1,i)=it4(1,1);

% shift down past time series of output
itt1(2:m-1,1)=itt1(1:m-2,1);
itt2(2:m-1,1)=itt2(1:m-2,1);
itt3(2:m-1,1)=itt3(1:m-2,1);
itt4(2:m-1,1)=itt4(1:m-2,1);

end % End of time stepping simulation


% copying time series to t
t=Cond(mg,1:ct1);

% Calculate cumulative results
% Yield
[b c]=size(ittc1);
ittc1m(1,1)=100;
ittc1m(1,2:c+1)=100-cumsum(ittc1);

% Salmonella inactivation
[b c]=size(ittc2);
ittc2m(1,1)=-10;
ittc2m(1,2:c+1)=-10-cumsum(ittc2);

% Internal color change
[b c]=size(ittc3);
```

```
ittc3m(1,1)=-10;
ittc3m(1,2:c+1)=-10-cumsum(ittc3);

% Surface color change
[b c]=size(ittc4);
ittc4m(1,1)=0;
ittc4m(1,2:c+1)=0+cumsum(ittc4);

% Transposing data
ittc1=ittc1m';
ittc2=ittc2m';
ittc3=ittc3m';
ittc4=ittc4m';

% Testing feasibility
results(1,1)=ittc1(ct1,1); % final yield
results(2,1)=ittc2(ct1,1); % final log(Salmonella.LR)
results(3,1)=ittc3(ct1,1); % final log(DI.LR)
results(4,1)=ittc4(ct1,1); % final DE.LR

Lf=results(2,1);
DIf=results(3,1);
DEf=results(4,1);


if (Lf >= 0.813) % Lfc=0.813,  2.5=316.2 log reduction
    Det_Lf=1; % passed
else
    Det_Lf=0; % failed
end

if (DIf >= 0.146) % DIfc=0.146
    Det_DIf=1; % passed
else
    Det_DIf=0; % failed
end

if (DEf >= 0.26)&& (DEf <= 0.497) % [0.26 0.497]
    Det_DEf=1; % passed
else
    Det_DEf=0; % failed
end

% 1: feasible, 2: infeasible
status=Det_Lf*Det_DIf*Det_DEf;
```

Simulated Annealing (SA)

File association:

SA_FEM_FS.m has following sub-programs:

- ***GenFourierParaSet.m***
- ***fun_FEM_FS.m***
- ***CheckRange.m***
- ***FourierFunGen.m***

```
% SA_FEM_FS.m
% Simulated annealing algorithm coupled with FEM and Fourier series
% parameterization method.


clear all
sprintf('Simulated Annealing Algorithm Started...')

% Parameters
Ni=20; % number of inner iteration

a=-0.5; % lower bound of random direction
b=0.5; % upper bound of random direction

N=21; % number of design variables of a control vector

iG=0;
nG=150; % total number of iteration

To=100; % initial temp
Tn=0.001;   % final temp


% Stepsize for each parameters
stepsize(1:20,1)=2; stepsize(21,1)=10; % Temperature
stepsize(22:41,1)=2; stepsize(42,1)=5; % Humidity
stepsize(43:62,1)=2; stepsize(63,1)=2; % Velocity
stepsize(64,1)=10; % duration


% Finding initial feasible control vector

% By using FEM
status=0;
while status==0
    X=GenFourierParaSet(N);
    % Evaluate performance index
    [status results]=fun_FEM_FS(X); % J=100-Yield
end

% % By reading from file
% fid=fopen(['/home/jeongsa1/Matlab/6/IP/IP_',num2str(4),'.dat']);
% g=fscanf(fid,'%f',[1, inf]); % [temp(1-21) steam(22-42) velocity(43-
63) time(64) yield SamonellaLR DI DEavg RMSE(T) RMSE(H) RMSE(V) ]
% g=g';
% fclose(fid);
% X=g(1:64,1);
% status=1;
% results=g(65:68,1);

% Assinging initial objective value
J=results(1,1)
i=1;
```

```
Jinit=J;
Xinit=X;

[rn cn]=size(X);
T=To;

% Starting iteration

for iG=1:(nG+1)
    k=0;
    C_accept=0; % Counter for acceptance
    C_improv=0; % Counter for improvement

    fprintf('%dth Generation\n',iG)

    while k < Ni % Inner loop begins
        status=0;
        while status==0
            Sf=0;

            while Sf == 0

                % generating random direction [a, b] @ n-dim.
hypersphere
                S = a + (b - a) * rand(rn,1);

                % Calculating new control vector
                Xnew = X + stepsize .* S;

                % checking feasiblility of Xnew
                Sf=CheckRange(Xnew);
            end

            % Obtaining prediction results by using process model
            [status results]=fun_FEM_FS(Xnew);
        end

        Jnew=results(1,1);
        delJ=Jnew-J;

        % Checking if the new result is improved
        if delJ > 0 % improved
            p=1;
            C_improv=C_improv+1;
        else
            p=exp(-abs(delJ)/T);% Boltzmann probability distribution
        end

        r=rand(1); %0.7; %rand(1)

        % New objective value is accepted and the design vector is
updated
        if p>r
            X=Xnew;
            J=Jnew;
            k=k+1;
```

```
            i=i+1;
            J_accepted(i,1)=J;
            X_accepted(i,:)=X;
            C_accept=C_accept+1;

        else
            k=k+1;
        end
        fprintf('k=%d  p=%f  r=%f  J=%f\n',k,p,r,J)
    end

    % Cooling Schedule
    N=iG;
    A=(To-Tn)*(nG+1)/nG; % meta variable
    B=To-A;              % meta variable
    T=A/(N+1)+B



    % Recording convergence history
    J_history(iG,1)=J; % last value at every temperature
    X_history(iG,:)=X;
    R_accept(iG,1)=C_accept; % number of acceptance
    R_improv(iG,1)=C_improv; % number of improvement

    % Saving workspace at every 5 iteration
    if(mod(iG,5)==0)

save(['/home/jeongsa1/Matlab/6/6/WorkSpace_',num2str(iG),'.mat']);
    end
end

% Plotting optimal control profiles
p=FourierFunGen(X,10);
time=p(:,1);
temp=p(:,2);
hum=p(:,3);
vel=p(:,4);

figure
h=plot(time,temp,'r-',time,hum,'b--',time,vel,'m-.');
xlabel('Process time [s]');
ylabel('Temperature [deg. C] / Humidity [%Mv] / Velocity [m/s]');
legend('Temperature','Humidity','Velocity',4);
set(h,'LineWidth',1.5);
```

```
% Generating Fourier series parameter set

function rv2=GenFourierParaSet(N)


% Upper and lower limit of each control variables
t_L=60;  t_U=600;  % time bounds [s]; L=lower  U=upper
T_L=100; T_U=250;  % Temperature bounds [C]
H_L=0;   H_U=100;  % Steam content bounds [% by volume]
V_L=0;   V_U=30;   % Impingement velocity bounds [m/s]

% generating a random cooking duration
duration=round(t_L+rand(1,1)*(t_U-t_L));
remain=mod(duration,10); % time should be integer and multiple of 10
duration=duration-remain;

% calculating Fourier coefficients
x(1:N,1)=CalculateFourierCoeffs(T_U, T_L, duration, 1);  % Temperature
x(N+1:2*N,1)=CalculateFourierCoeffs(H_U, H_L, duration, 1); % Humidity
x(2*N+1:3*N,1)=CalculateFourierCoeffs(V_U, V_L, duration, 1); %
Velocity
x(3*N+1,1)=duration; % Cooking duration


rv2=x;
```

```
% Calculating prediction results by using FEM and Fourier parametes

function [status, results]=fun_FEM_Para(v) % nlc: nonlinear constraints

% Getting profiles with 0.5s interval
C=FourierFunGen(v, 0.5);
duration=v(64,1);
Te=C(:,2);
Se=C(:,3);
Ve=C(:,4);
dt=0.5;

% FEM Solution
[rv1 rv2 rv3 rv4 rv5 rv6 rv7
rv8]=FE_DataGen_half(duration,Te,Se,Ve,dt);

[nr nc]=size(rv1);

% Re-arranging results
results(1,1)=rv1(nr,1); % Yield
results(2,1)=rv3(nr,1); % Lethality
results(3,1)=rv4(nr,1); % DI
results(4,1)=rv6(nr,1); % DE

Lf=results(2,1);
DIf=results(3,1);
DEf=results(4,1);

% Checking feasibility

% Salmonella reduction
if (Lf >= 6.5)
    Det_Lf=1; % passed
else
    Det_Lf=0; % failed
end

% Internal color change
if (DIf >= 1.4)
    Det_DIf=1; % passed
else
    Det_DIf=0; % failed
end

% Surface color change
if (DEf >= 0.263)&& (DEf <= 0.497)
    Det_DEf=1; % passed
else
    Det_DEf=0; % failed
end

% 1=feasible; 0=infeasible
sumDet=Det_Lf*Det_DIf*Det_DEf;
```

```
% Check if the data points generated by Fourier coeffs. are in the
upper
% and lower bounds.

function status=CheckRange(X)

% Limits of variables
Bounds=[100 250; 0 100; 0 30]; %[TL TU; HL HU; VL VU]

[r c]=size(X);

duration=X(r,1);
timestep=1;
kmax=10; % maximum # of series
s=2*pi/((duration/2)*kmax); % period

for k=1:3
    base=1+21*(k-1);
    a=X(base:base+(kmax-1),1);
    b=X(base+kmax:base+2*kmax-1,1);
    f0=X(base+2*kmax,1);
    i=0;
    for ti=0:timestep:duration
        i=i+1;
        sum=0;

        for m=1:kmax
            sum=sum+a(m,1)*sin(s*m*ti)+b(m,1)*cos(s*m*ti);
        end
        fs(i,1)=f0+sum;
    end

    if ((max(fs) <= Bounds(k,2)) && (min(fs) >= Bounds(k,1)))
        S(k,1)=1;
    else
        S(k,1)=0;
    end
end

% 1=feasible, 0=infeasible
status=S(1,1)*S(2,1)*S(3,1);
```

```
% Generating control profile with 0.5s interval by using Fourier
coeffs.

function rv1 = FourierFunGen(v,timestep) %  v=vector

% Assigning Fourier coefficients
aT(1:10,1)=v(1:10,1); bT(1:10,1)=v(11:20,1); foT=v(21,1); % Temperature
aM(1:10,1)=v(22:31,1); bM(1:10,1)=v(32:41,1); foM=v(42,1); % Humidity
aV(1:10,1)=v(43:52,1); bV(1:10,1)=v(53:62,1); foV=v(63,1); % Velocity
duration=v(64,1);

kmax=10; % maximum # of series
s=2*pi/((duration/2)*kmax); % period

i=0;

% Calculating
for t=0:timestep:duration %j=0:(quotient+1)

    i=i+1;
    sumT=0;
    sumM=0;
    sumV=0;
    for k=1:kmax
        sumT=sumT+aT(k,1)*sin(s*k*t)+ bT(k,1)*cos(s*k*t);
        sumM=sumM+aM(k,1)*sin(s*k*t)+ bM(k,1)*cos(s*k*t);
        sumV=sumV+aV(k,1)*sin(s*k*t)+ bV(k,1)*cos(s*k*t);
    end
    fT=foT+sumT;
    fM=foM+sumM;
    fV=foV+sumV;

    Cond(i,1)=t;
    Cond(i,2)=fT;
    Cond(i,3)=fM;
    Cond(i,4)=fV;

end

% returning profile data
rv1=Cond;
```

ICRS Algorithm

File association:

ICRS_FEM_PLI.m has following sub-programs:

- ***GetMinInterval.m***
- ***fun_FEM_PLI.m*** **
- ***CheckRange_PLI.m*** **

** *These files are included in the genetic algorithm in this study.*

```
% ICRS Algorithm

clear all
fprintf('ICRS+FEM Algorithm started\n')

% Parameters and input data
k=1;                        % iteration counter
tol_criterion=0.0001;       % convergence criterion
iG=400;                     % max. iteration

% Heuristic parameters
k1=1/3;
k2=1/2;
ne=100; % Failure counter limit

N=21; % 21 control points for each control vector
M=4;  % # of control vectors
TotDim=N*(M-1)+1; % Total # of dimensions

% Ranges of parameters
XiU(    1:N,  1)=250; % temperature
XiU(  N+1:2*N,1)=100; % steam
XiU(2*N+1:3*N,1)=30;  % velocity
XiU(3*N+1,1)=600;     % time

XiL(    1:N,  1)=100;
XiL(  N+1:2*N,1)=0;
XiL(2*N+1:3*N,1)=0.5;
XiL(3*N+1,1)=60;

% Getting initial contol vectors by reading a file
fid=fopen(['/home/jeongsa1/Matlab/3/IP/IDsmooth_',num2str(5),'.dat']);
g=fscanf(fid,'%f',[1, inf]);
g=g';
fclose(fid);

Xik=g(1:64,1);
status=1;
results=g(65:68,1);

% Assigning initial objective value
Jo=results(1,1);

% Storing initial control vector
Xik_History(k,:)=Xik';
J_History(k,1:4)=results';


tol=1;

% Starting iteration
while  (k < iG)
```

```
    % Calculate the minimum intervals of the decision variables
    Lik=GetMinInterval(XiU, XiL, Xik);

    % Calculate the vector of standard deviations, using k1
    Sik=k1*Lik;

    F=0;
    Jn=-1; % just for safety

    while (status ~= 1)||(Jn < Jo)

        [r c]=size(Xik);
        F;
        if F > (ne)
            Sik=k2*Sik;

            F=0;
        end

        ParaStatus=0;

        % Generate a new decision vector using Gaussian distribution
        while ParaStatus==0
            XikNew = Xik + Sik .* randn(r,1);
            ParaStatus=CheckRange_PLI(XikNew);
        end

        % Evaluating performance index
        [status results]=fun_FEM_PLI(XikNew);
        Jn=results(1,1);

        if (status == 0)||(Jn <= Jo)
            F=F+1; % Increase failure counter by 1
        end
    end

    % Check the convergence
    if isnan(Jn)==0
        tol=abs(Jn-Jo)/abs(Jo);
        k=k+1;
        Xik=XikNew;
        Jo=Jn;

        Xik_History(k,:)=Xik';
        J_History(k,1:4)=results';
        tol_History(k,1)=tol;
    else
        fprintf('Skipped because of NaN\n');
    end

fprintf('k=%d  F=%d  J=%f\n',k,F,Jn)

    % Saving workspace at every 2 iteration
    if(mod(k,2)==0)

save(['/home/jeongsa1/Matlab/3/IP/WorkSpace_',num2str(k),'.mat']);
    end
```

```
end

fprintf('End of ICRS+FEM Algorithm\n');

% Exporting result
% History of generations

fid=fopen(['/home/jeongsa1/Matlab/3/3/Result_history_Fourier.dat'],'wt'
);

for i=1:k
    for j=1:TotDim
        fprintf(fid,'%10.4f',Xik_History(i,j));
    end
    fprintf(fid,'%10.4f %10.4f %10.4f %10.4f',J_History(i,:));
    fprintf(fid,'%10.4f\n',tol_History(i,1));
end
fclose(fid);

% Reporting ICRS summary
fid3=
fopen(['/home/jeongsa1/Matlab/3/3/Result_summary_Fourier.dat'],'wt');

fprintf(fid3,'-------------------------------------------------------
\n');
fprintf(fid3,' SUMMARY OF ICRS-GRNN OPTIMIZATION\n');
fprintf(fid3,' >Control vector parameterization with Fourier
series\n');
fprintf(fid3,' >ICRS-GRNN found the best combination of 64 descision
parameters\n');
fprintf(fid3,'-------------------------------------------------------
\n');
fprintf(fid3,'Yield_max= %10.4f [%%] \nlog(LR.microb)= %10.4f\nlog(DI)=
%10.4f\nlog(DE)= %10.4f\n',J_History(k,1:4));
fprintf(fid3,'Cooking duration= %10.4f [s] \n',Xik_History(k,64));
fprintf(fid3,'tolerance= %10.4f  \n\n',tol_History(k,1));
fprintf(fid3,'[Optimization Parameters]\n');
fprintf(fid3,'1. # of Decision parameters:    %d\n',N);
fprintf(fid3,'2. # of Control variables:      %d\n',M);
fprintf(fid3,'3. Total dimension of problem:  %d\n',TotDim);
fprintf(fid3,'4. k1 heuristic parameter:      %6.4f\n',k1);
fprintf(fid3,'5. k2 heuristic parameter:      %6.4f\n',k2);
fprintf(fid3,'6. ne heuristic parameter:      %6.4f\n',ne);
fprintf(fid3,'7. Convergence criterion:       %6.4d\n',tol_criterion);
fprintf(fid3,'8. # of Fourier series:         %d\n',10);
fprintf(fid3,'9. Total # of iteration:        %d\n',k);
fprintf(fid3,'-------------------------------------------------------
\n');

fclose(fid3);
```

```
% Calculating minimum intervals of the decision variables

function rv=GetMinInterval(U, L, V)

[r c]=size(V);


for i=1:r
    dL=V(i,1)-L(i,1);
    dU=U(i,1)-V(i,1);

    if dL <= dU
        r(i,1)=dL;
    else
        r(i,1)=dU;
    end
end
rv=r;
```

Quality Prediction Sub-Programs

These sub-programs were added in the FEM of Watkins (2004).

- Salmonella inactivation: ***CalculateSurvivors.m***
- Internal color change: ***CalculateCenterColorChange.m***
- Surface color change: ***CalculateAveragedSurfaceColorChange.m***

```
% Calculate the number of surviving microorganisms for eanch node at
each time step

function []=CalculateSurvivors()

global  NumNodes timestep temperature Ninitial No NW Nnew NnewW
logreduction logreductionW Dvalue Tref Z TimeToLimit

m=1000000; % factor to reduce Dr

% Log-linear inactivation equation

for X = 1:NumNodes
    d = m*Dvalue * 10 ^ ((Tref - temperature(X)) / Z);
    Nnew(X) = No(X) / (10 ^ (timestep / d));
    No(X) = Nnew(X);
    logreduction(X) = -log(No(X) / Ninitial) / log(10);
    logreduction(X)=m*logreduction(X); % restoring original Dr

    % Weibull inactivation equation

    if logreductionW < 9
        b = 0.000000000011047 * exp(0.41758 * temperature(1)); %'0.03 *
(temperature(1)) ^ 2 - (2.7 * temperature(1)) + 72.19
        n = 1.12;
        NnewW(X) = NW(X) * (10 ^ (-b * ((timestep / 60) ^ n)));
        if Nnew(X) < 1
            Nnew(X) = 1;
        end
        NW(X) = NnewW(X);
        logreductionW = -log(NW(X) / Ninitial) / log(10);
    else
        logreductionW = 9;
    end

end
```

```
% Calculate log reduction OMB(Oxy-myoglobin) concentration at center

function [rv]=CalculateCenterColorChange(t, Tc,delT)

% Geileskey A. et al. (1998)
% Target log reduction = 1.387 which is 95.9% denaturation of OMB which
is
% no evidence of pink.

% The worst case parameters:
Dr=5116.86;    % [s] beef shin
z=10.41;       % [C] beef m. l. dorsi
Tr=60;         % [C]
m=1000;        % mDr=m*Dr Overflow Preventing Multiplier

mDr=m*Dr;

t=t';
Dt=mDr*10.^((Tr-Tc)/z); % Calculating D-value at time t
LRt=(-1 ./ Dt)*delT;    % Calculating Log Reduction at time t for delT

mCumulative_LR=cumsum(LRt');    % Cumulative Log Reduction
Cumulative_LR=m*mCumulative_LR; % Restoring cumulative log reduction

rv=(-Cumulative_LR');
```

```
% Calculate color change at surface

function [rv]=CalculateSurfaceColorChange(t, Td, Ts, delT)

% Dagerskog and Bengtsson (1974)
% The crust color kinetic parameters were obtained from the graphical
data

Dr=465.30; %[s]
z=90.55;   %[C]
Tr=110;    %[C]
a=1.06;    % Activation function parameter(99.5% activation level
within +/- 5C)
m=1000;    % mDr=m*Dr Overflow Preventing Multiflier

mDr=m*Dr;
[rn cn]=size(Ts);

t=t';
for i=1:rn
    f=1/(1+exp(-a*(Ts(i,1)-Td(i,1)))); % Activation function
    Dt=mDr*10^((Tr-Ts(i,1))/z); % Calculating D-value at time t
    LRt(i,1)=f*(-1 / Dt)*delT;  % Calculating Log Reduction at time t
for delT
end
mCumulative_LR=cumsum(LRt');    % Cumulative Log Reduction
Cumulative_LR=m*mCumulative_LR; % Restoring cumulative log reduction

rv=(-Cumulative_LR');
```

REFERENCES

REFERENCES

Adhikari, B. and V. K. Jindal 2000. Artificial neural networks: A new tool for prediction of pressure drop of non-Newtonian fluid foods through tubes. Journal of Food Engineering 46(1): 43-51.

Albert, S., H. Hiden, A. Conlin, E. B. Martin, G. A. Montague and A. J. Morris. 2001. Inferential quality assessment in breakfast cereal production. Journal of Food Engineering 50(3): 157-166.

AMI. 2001. "Overview of U.S. Meat and poultry production and consumption." American Meat Institute Fact Sheet Retrieved 03/21, 2005, from http://www.meatami.com/Content/NavigationMenu/PressCenter/FactSheets_InfoKits/FactSheetMeatProductionandConsumption.pdf.

AMI. 2002. "Timeline of factors impacting recall frequency and foodborne disease rates." American Meat Institute Fact Sheet. Retrieved 3/21, 2005, from http://www.meatami.com/Content/Navigation/Menu/PressCenter/FactSheets_InfoKits/FactSheetRecallFactors.pdf.

Arteaga, G. E., E. Li-Chan, M. C. Vazquez-Arteaga and S. Nakai. 1994. Systematic experimental designs for product formula optimization. Trends in Food Science & Technology 5(8): 243-254.

Ateba, P. and G. S. Mittal. 1994. Dynamics of crust formation and kinetics of quality change during frying of meatballs. Journal of Food Science 59(6): 1275-1278.

Banga, J. R., A. A. Alonso and R. P. Singh. 1997. Stochastic dynamic optimization of batch and semi-continuous bioprocesses. Biotechnol. Prog. 13(3): 326-335.

Banga, J. R., E. Balsa-Canto, C. G. Moles and A. A. Alonso. 2003. Improving food processing using modern optimization methods. Trends in Food Science & Technology 14(4): 131-144.

Banga, J. R. and J. J. Casares Long. 1987. Integrated controlled random search: Application to a wastewater treatment plant model. The Institution of Chemical Engineers Symposium Series 100: 183-192.

Banga, J. R., Z. Pan and R. P. Singh. 2001. On the optimal control of contact-cooking processes. Food & Bioproducts Processing 79: 145-151.

Banga, J. R., R. I. Perez-Martin, J. M. Gallardo and J. J. Casares. 1991. Optimization of the thermal processing of conduction-heated canned foods: Study of several objective functions. Journal of Food Engineering 14(1): 25-51.

Banga, J. R. and R. P. Singh. 1994. Optimization of air drying of foods. Journal of Food Engineering 23(2): 189-211.

Barreiro, J. A., C. R. Perez and C. Guariguata. 1984. Optimization of energy consumption during the heat processing of canned foods. Journal of Food Engineering 3(1): 27-37.

Berber, R., C. Pertev and M. Türker. 1998. Optimization of feeding profile for baker's yeast production by dynamic programming. Bioprocess Engineering 20: 263-269.

Berg, T., U. Erikson and T. S. Nordtvedt. 1997. Rigor mortis assessment of Atlantic salmon (*salmo salar*) and effects of stress. Journal of Food Science 62(3): 439-446.

Beveridge, G. S. G. and R. S. Schechter. 1970. Optimization: Theory and practice. New York, McGraw-Hill.

Boillereaux, L., C. Cadet and A. Le Bail. 2003. Thermal properties estimation during thawing via real-time neural network learning. Journal of Food Engineering 57(1): 17-23.

Brennan, J. G. 1990. Food engineering operations. London; New York, Elsevier Applied Science.

Buzby, J. C., T. Roberts, C.-T. J. Lin and J. M. MacDonald. 1996. Bacterial foodborne disease: Medical costs and productivity loses. Agricultural Economic Report (Economic Research Service/USDA) No. 741.

Casares, J. J. and J. Rodriguez. 1989. Analysis and evaluation of a wastewater treatment plant model by stochastic optimization. Appl. Math. Modeling 13(July): 420-424.

Chalabi, Z. S., L. G. van Willigenburg and G. van Straten. 1999. Robust optimal receding horizon control of the thermal sterilization of canned foods. Journal of Food Engineering 40(3): 207-218.

Chao, K., Y.-R. Chen, W. R. Hruschka and F. B. Gwozdz. 2002. On-line inspection of poultry carcasses by a dual-camera system. Journal of Food Engineering 51(3): 185-192.

Chávez-Jáuregui, R. N., M. E. M. P. Silva and J. A. G. Arêas. 2000. Extrusion cooking process for Amaranth (*Amaranthus caudatus* L.). Journal of Food Science 65(6): 1009-1015.

Chen, C. R. and H. S. Ramaswamy. 2002. Modeling and optimization of variable retort temperature (VRT) thermal processing using coupled neural networks and genetic algorithms. Journal of Food Engineering 53(3): 209-220.

Chen, C. R. and H. S. Ramaswamy. 2003. Analysis of critical control points in deviant thermal processes using artificial neural networks. Journal of Food Engineering 57(3): 225-235.

Christodoulou, C. and M. Georgiopoulos. 2001. Applications of neural networks in electromagnetics. Boston, MA, Artech House.

Dagerskog, M. and N. E. Bengtsson. 1974. Pan frying of meat patties-relationship among crust formation, yield, composition and processing conditions. Lebensmittel-Wissenschaft und-Technologie 7(4): 202-207.

Datta, A. K. 2002. Biological and bioenvironmental heat and mass transfer. New York, Marcel Dekker, Inc.

Diaz, R., L. Gil, C. Serrano, M. Blasco, E. Molto and J. Blasco. 2004. Comparison of three algorithms in the classification of table olives by means of computer vision. Journal of Food Engineering 61(1): 101-107.

Edgar, T. F., D. M. Himmelblau and L. S. Lasdon. 2001. Optimization of chemical processes. New York, McGraw-Hill.

Erdoğdu, F. 2002. Nonlinear constrained optimization of thermal processing: I. Development of a modified algorithm of complex method. Journal of Food Process Engineering 25: 1-22.

Erdoğdu, F. and M. O. Balaban. 2003. Complex method for nonlinear constrained multi-criteria (multi-objective function) optimization of thermal processing. Journal of Food Process Engineering 26(4): 357-375.

ERS. 2000. "ERS/USDA briefing room - Economics of foodborne disease: Estimating the costs of bacterial foodborne disease." Retrieved 4/28, 2005, from http://www.ers.usda.gov/briefing/FoodborneDisease/features.htm.

ERS. 2002. "ERS/USDA briefing room - Consumer food safety behavior." Retrieved 4/28, 2005, from http://www.ers.usda.gov/Briefing/ConsumerFoodSafety/consumerconcerns/.

Evans, L. B. 1982. Optimization theory and its application in food processing. Food Technology(July): 88-93.

Faqir, N. M. 1998. Optimization of glucose isomerase reactor: Optimum operating temperature mode. Bioprocess Engineering 18: 389-396.

Floquet, P., L. Pibouleau and S. Domenech. 1994. Separation sequence synthesis: How to use simulated annealing procedure? Computers & Chemical Engineering 18(11-12): 1141.

FMI. 1999. Trends in the United States--Consumer attitudes & the supermarket. In *minimizing microbiological food safety risks: Potential for preslaughter (preharvest) interventions*: White paper. Ransom J. R., Sofos J. N., Scanga, J. A., and Smith, G. C., Center for Red Meat Safety, Colorado State University, Fort Collins, CO 80523-1171.

Fravolini, M. L., A. Ficola and M. La Cava. 2003. Optimal operation of the leavening process for a bread-making industrial plant. Journal of Food Engineering 60(3): 289-299.

Frazier, P. J., A. Crawshaw, N. W. R. Daniels and P. W. Russell Eggitt. 1983. Optimisation of process variables in extrusion texturing of soya. Journal of Food Engineering 2(2): 79-103.

FSIS. 1999. Performance standards for the production of certain meat and poultry products. 9 cfr parts 301, 317, 318, 320, and 381: Final rule. Federal register. Docid: Fr06ja99-2. Food Safety and Inspection Service, U.S. Department of Agriculture, Washington, DC.

FSIS. 2001. Performance standards for the production of processed meat and poultry products; proposed rule. Federal register. February 27, 2001. 12590-12636.

Ganjyal, G. and M. Hanna. 2002. A review on residence time distribution (RTD) in food extruders and study on the potential of neural networks in RTD modeling. Journal of Food Science 67(6): 1996-2002.

Ganjyal, G. M., M. A. Hanna and D. D. Jones. 2003. Modeling selected properties of extruded waxy maize cross-linked starch with neural networks. Journal of Food Science 68(4): 1384-1388.

García-Gimeno, R. M., C. Hervás-Martínez, E. Barco-Alcalá, G. Zurera-Cosano and E. Sanz-Tapia. 2003. An artificial neural network approach to *Escherichia coli* O157:H7 growth estimation. Journal of Food Science 68(2): 639-645.

Geeraerd, A. H., C. H. Herremans, L. R. Ludikhuyze, M. E. Hendrickx and J. F. Van Impe. 1998. Modeling the kinetics of isobaric-isothermal inactivation of *Bacillus subtilis* α-amylase with artificial neural networks. Journal of Food Engineering 36(3): 263-279.

Geileskey, A., R. D. King, D. Corte, P. Pinto and D. A. Ledward. 1998. The kinetics of cooked meat haemoprotein formation in meat and model systems. Meat Science 48(3/4): 189-199.

Gergely, S., E. Bekassy-Molnar and G. Vatai. 2003. The use of multiobjective optimization to improve wine filtration. Journal of Food Engineering 58(4): 311-316.

Golan, E., T. Roberts, E. Salay, J. Caswell, M. Ollinger and D. Moore. 2004. Food safety innovation in the United States: Evidence from the meat industry. Economic Research Service/USDA. Agricultural Economic Report. No.831.

Goulcher, R. and J. J. Casares Long. 1987. The solution of steady-state chemical engineering optimization problems using a random-search algorithm. Computers and Chemical Engineering 2: 33-36.

Halsall-Whitney, H., D. Taylor and J. Thibault. 2003. Multicriteria optimization of gluconic acid production using net flow. Bioprocess & Biosystems Engineering 25: 299-307.

Haykin, S. S. 1999. Neural networks: A comprehensive foundation. Upper Saddle River, N.J., Prentice Hall.

Holland, J. H. 1962. Outline for a logical theory of adaptive systems. Journal of the ACM 9(3): 297-314.

Horiuchi, J.-i., T. Shimada, H. Funahashi, K. Tada, M. Kobayashi and T. Kanno. 2004. Artificial neural network model with a culture database for prediction of acidification step in cheese production. Journal of Food Engineering 63(4): 459-465.

Hunt, M. C., O. Sorheim and E. Slinde. 1999. Color and heat denaturation of myoglobin forms in ground beef. Journal of Food Science 64(5): 847-851.

Hussain, M. A., M. Shafiur Rahman and C. W. Ng. 2002. Prediction of pores formation (porosity) in foods during drying: Generic models by the use of hybrid neural network. Journal of Food Engineering 51(3): 239-248.

Iwe, M. O., I. Wolters, G. Gort, W. Stolp and D. J. van Zuilichem. 1998. Behaviour of gelatinisation and viscosity in soy-sweet potato mixtures by single screw extrusion: A response surface analysis. Journal of Food Engineering 38(3): 369-379.

Juneja, V. K., B. S. Eblen and G. M. Ransom. 2001. Thermal inactivation of *Salmonella spp.* in chicken broth, beef, pork, turkey, and chicken: Determination of D- and Z-values. Journal of Food Science 66(1): 146-152.

Karwe, M. V. and S. Godavarti. 1997. Accurate measurement of extrudate temperature and heat loss on a twin-screw extruder. Journal of Food Science 62: 367-372.

Kearfott, R. B. 1996. Rigorous global search: Continuous problems. Dordrecht; Boston, Kluwer Academic Publishers.

Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi. 1983. Optimization by simulated annealing. Science 220(4598): 671-680.

Kleis, D. and E. W. Sachs. 2000. Optimal control of the sterilization of prepackaged food. SIAM Journal on Optimization 10: 1180-1195.

Laarhoven, P. J. M. v. and E. H. L. Aarts. 1987. Simulated annealing: Theory and applications. Norwell, MA, Kluwer Academic Publishers.

Lee, J.-H., H. C. Lim, Y. J. Yoo and Y. H. Park. 1999. Optimization of feed rate profile for the monoclonal antibody production. Bioprocess and Biosystems Engineering 20(2): 137-146.

Levisauskas, D., V. Galvanauskas, S. Henrich, K. Wilhelm, N. Volk and A. Lubbert. 2003. Model-based optimization of viral capsid protein production in fed-batch culture of recombinant *Escherichia coli*. Bioprocess & Biosystems Engineering 25: 255-262.

Lingnert, H. 1990. Development of the Maillard reaction during food processing. In *The Maillard reaction in food processing, human nutrition and physiology*. P. A. Finot, H. U. Aeschbacher, R. F. Hurrell and R. Liardon. Basel;Boston;Berlin, Birkhauser Verlag.

Marique, T., A. Kharoubi, P. Bauffe and C. Ducattillon. 2003. Modeling of fried potato chips color classification using image analysis and artificial neural network. Journal of Food Science 68(7): 2236-2266.

Martins, S. I. F. S. 2003. Unravelling the Maillard reaction network by multiresponse kinetic modeling. Netherlands, Wageningen University. Thesis(Ph.D.): 170.

Martins, S. I. F. S., W. M. F. Jongen and M. A. J. S. van Boekel. 2001. A review of Maillard reaction in food and implications to kinetic modelling. Trends in Food Science & Technology 11(9-10): 364-373.

Martins, S. I. F. S. and M. A. J. S. Van Boekel. 2005. A kinetic model for the glucose/glycine Maillard reaction pathways. Food Chemistry 90(1-2): 257-269.

Mead, P. S., L. Slutsker, V. Dietz, L. F. McCaig, J. S. Bresee, C. Shapiro, P. M. Griffin and R. V. Tauxe. 1999. Food-related illness and death in the United States. Centers for Disease Control and Prevention. Atlanta, GA.

Millsap, S. C. 2002. Modeling condensing-convective boundary conditions in moist air impingement ovens. Dept. of Agricultural Engineering. East Lansing, Michigan State University. M.S.: xi, 117 leaves.

Mishkin, M., M. Karel and I. Saguy. 1982. Application of optimization in food dehydration. Food Technology(36): 101-109.

Mishkin, M., I. Saguy and M. Karel. 1983. Minimizing ascorbic acid loss during air drying with a constraint on enzyme inactivation for a hypothetical foodstuff. Journal of Food Processing & Preservation 7: 193-210.

Mittal, G. S. and J. Zhang. 2000. Use of artificial neural network to predict temperature, moisture, and fat in slab-shaped foods with edible coatings during deep-fat frying. Journal of Food Science 65(6): 978-983.

Morimoto, T., W. Purwanto, J. Suzuki and Y. Hashimoto. 1997. Optimization of heat treatment for fruit during storage using neural networks and genetic algorithms. Computers and Electronics in Agriculture 19: 87-101.

Mottram, D. S. 1998. Flavour formation in meat and meat product: A review. Food Chemistry 62(4): 415-424.

Murphy, R. Y., L. K. Duncan, E. R. Johnson, M. D. Davis and J. N. Smith. 2002. Thermal inactivation D- and Z-values of *Salmonella* serotypes and *Listeria innocua* in chicken patties, chicken tenders, franks, beef patties, and blended beef and turkey patties. Journal of Food Protection 65(1): 53-60.

Murphy, R. Y., E. M. Martin, L. K. Duncan, B. L. Beard and J. A. Marcy. 2004. Thermal process validation for *Escherichia coli* O157:H7, *Salmonella*, and *Listeria monocytogenes* in ground turkey and beef products. Journal of Food Protection 67(7): 1394-1402(9).

Nadkarni, M. M. and T. A. Hatton. 1985. Optimal nutrient retention during the thermal processing of conduction-heated canned foods: Application of the distributed minimum principle. Journal of Food Science 50: 1312-1321.

Ohlsson, T. 1980. Optimal sterilization temperatures for sensory quality in cylindrical containers. Journal of Food Science 45: 1517-1521.

Oliveira, F. A. R. and J. C. Oliveira. 1999. Processing foods: Quality optimization and process assessment. Boca Raton, Florida, CRC Press.

Olkku, J., A. Hagqvist and P. Linko. 1983. Steady-state modelling of extrusion cooking employing response surface methodology. Journal of Food Engineering 2(2): 105-128.

Ollinger, M. and N. Ballenger. 2003. Weighing incentives for food safety in meat and poultry. Amber Waves. April.

Pan, Z. 1998. Predictive modeling and optimization of hamburger patty contact-cooking process. Dept. of Biological and Agricultural Engineering. Davis, University of California. Ph.D.: xv, 169 leaves.

Quintero-Ramos, A., M. C. Bourne, J. Barnard and A. Anzaldúa-Morales. 1998. Optimization of low temperature blanching of frozen jalapeño pepper (*capsicum annuum*) using response surface methodology. Journal of Food Science 63(3): 519-522.

Radhakrishnan, T. K., S. Sundaram and M. Chidambaram. 1999. Non-linear control of continuous bioreactors. Bioprocess Engineering 20: 173-178.

Raptis, C. G., C. I. Siettos, C. T. Kiranoudis and G. V. Bafas. 2000. Classification of aged wine distillates using fuzzy and neural network systems. Journal of Food Engineering 46(4): 267-275.

Saad, N. R., A. S. Mujumdar and W. J. M. Douglas. 1980. Heat transfer under multiple turbulent slot jets impinging on a flat plate. In Drying '80, developments in drying (pp. 422-430). A. S. Mujumdar. Washington, Hemisphere Pub. Corp.

Sablani, S. S. and W. H. Shayya. 2001. Computerization of Stumbo's method of thermal process calculations using neural networks. Journal of Food Engineering 47(3): 233-240.

Sablani, S. S. and W. H. Shayya. 2003. Neural network based non-iterative calculation of the friction factor for power law fluids. Journal of Food Engineering 57(4): 327-335.

Saguy, I. and M. Karel. 1979. Optimal retort temperature profile in optimizing thiamine retention in conduction-type heating of canned foods. Journal of Food Science 44: 1485-1490.

Segerlind, L. J. (1984). Applied finite element analysis. New York, Wiley.

Shigley, J. E. and C. R. Mischke. 1989. Mechanical engineering design. New York, McGraw-Hill.

Silva, C., M. Hendrickx, F. Oliveira and P. Tobback. 1992. Optimal sterilization temperatures for conduction heating foods considering finite surface heat transfer coefficients. Journal of Food Science 57: 743-748.

Singh, P. P. and V. K. Jindal. 2003. Pressure drop estimation in tube flow of non-Newtonian fluid foods by neural networks. Journal of Food Process Engineering 26: 49-65.

Specht, D. F. 1991. A general regression neural network. IEEE Transactions on Neural Networks 2(6): 568-576.

Tartakovsky, B., S. Ulitzur and M. Sheintuch. 1995. Optimal control of fed-batch fermentation with autoinduction of metabolic production. Biotechnology and Progress 11: 80-87.

Teixeira, A. A., G. E. Zinsmeister and J. W. Zahradnik. 1975. Computer simulation of variable retort control and container geometry as a possible means of improving thiamine retention in thermally processed foods. Journal of Food Science 40: 656-659.

Teixeria, A. A., J. R. Dixon, J. W. Zahradnik and G. E. Zinsmeister. 1969. Computer optimization of nutrient retention in the thermal processing of conduction-heated foods. Food Technology 23: 137-142.

Therdthai, N. and W. Zhou. 2001. Artificial neural network modelling of the electrical conductivity property of recombined milk. Journal of Food Engineering 50(2): 107-111.

Therdthai, N., W. Zhou and T. Adamczak. 2002. Optimisation of the temperature profile in bread baking. Journal of Food Engineering 55(1): 41-48.

Toledo, R. T. 1991. Fundamentals of food process engineering. New York, Van Nostrand Reinhold.

Tominaga, O., F. Ito, T. Hanai, H. Honda and T. Kobayashi. 2001. Sensory modeling of coffee with fuzzy neural network. Journal of Food Science 67(1): 363-368.

Torrecilla, J. S., L. Otero and P. D. Sanz. 2004. A neural network approach for thermal/pressure food processing. Journal of Food Engineering 62(1): 89-95.

Torrecilla, J. S., L. Otero and P. D. Sanz. 2005. Artificial neural networks: A promising tool to design and optimize high-pressure food processes. Journal of Food Engineering 69(3): 299-306.

Trelea, I. C., G. Trystram and F. Courtois. 1997. Optimal constrained non-linear control of batch processes: Application to corn drying. Journal of Food Engineering 31(4): 403-421.

Trystram, G. 2004. Symposium 8: Engineering: Solutions to enhance food safety reevaluation of thermal food processes in order to increase food safety and quality: Frying, drying, salting, smoking. Journal of Food Science 69(5): E251-7.

Tsoneva, R. G., T. D. Patarinska and I. P. Popchev. 1998. Augmented Lagrange decomposition method for optimal control calculation of batch fermentation processes. Bioprocess Engineering 18: 143-153.

Vainionpaa, J. 1991. Modelling of extrusion cooking of cereals using response surface methodology. Journal of Food Engineering 13(1): 1-26.

Venkataraman, P. 2002. Applied optimization with MATLAB programming. New York, John Wiley & Sons.

Voller, V. and M. Cross. 1981. Accurate solutions of moving boundary problems using the enthalpy method. International Journal of Heat and Mass Transfer 24: 545-556.

Wasserman, P. D. 1993. Advanced methods in neural computing. New York, Van Nostrand Reinhold.

Watkins, A. E. 2004. A combined convection cooking and salmonella inactivation model for ground meat and poultry products. Dept. of Biosystems and Agricultural Engineering. East Lansing, Michigan State University. Ph.D.: xix, 260 leaves.

Xie, G. and R. Xiong. 1999. Use of hyperbolic and neural network models in modelling quality changes of dry peas in long time cooking. Journal of Food Engineering 41(3-4): 151-162.

Zanoni, B., C. Peri and D. Bruno. 1995. Modelling of browning kinetics of bread crust during baking. Lebensmittel-Wissenschaft und-Technologie 28(6): 604-609.

Zorrilla, S. E., J. R. Banga and R. P. Singh. 2003. Dynamic optimization of double-sided cooking of meat patties. Journal of Food Engineering 58(2): 173-182.