uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Wei Liu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Mechanical Engineering)

GRADE / DEGREE

Department of Mechanical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Multi-objective Approach for RMT Design

TITRE DE LA THÈSE / TITLE OF THESIS

M. Liang

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

R. Liu

D. Redekop

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

A Multi-objective Approach for RMT Design

Wei Liu

Thesis submitted to the
Faculty of Graduate and Postdoctoral Students
In partial fulfillment of the requirements
For the MASc degree in Mechanical Engineering

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-14922-1
Our file Notre référence
ISBN: 0-494-14922-1

Abstract

A reconfigurable manufacturing system (RMS) is designed for rapid adjustment of manufacturing capacity and functionality in response to market changes. An RMS consists of a number of reconfigurable machine tools (RMTs) which can process different jobs by quickly changing processing modules. The potential benefits of an RMS may not be achieved if an RMS is not properly designed. Most of the related studies focus on a few individual technical issues, in particular on modularity or configurability of individual RMTs. Other important concerns such as cost and processing accuracy have not been adequately addressed. As a result, many highly reconfigurable manufacturing systems turn out to be unprofitable.

For the above reason, this study focuses on optimization of RMT design, including the design of modules and module warehouse, with consideration of three factors: configurability, cost and accuracy. Since the three design factors are non-commensurable and often conflicting with each other, the design optimization problem is formulated as a multi-objective model. To solve the problem, a mechanism is developed to generate and evaluate the alternative designs. A set of rules are also established for merging RMT modules and their operations to ensure that alternative designs can be automatically generated and evaluated. A modified fuzzy-Chebyshev programming (MFCP) method is developed to solve the multi-objective problem and to achieve the best compromise of the three design factors without violating the pre-specified priority order of the three design criteria. With the proposed MFCP method, the relative weights between objectives can be automatically generated during the optimization process based on the satisfaction levels of the objectives. Due to the complexity and large size of the multi-objective problem, it is unrealistic to use commercial software to provide quick solutions. A particle swarm optimization algorithm is accordingly developed to provide quick and near optimal solutions for large design problems. The application of the proposed multi-objective design approach is demonstrated using a reconfigurable boring machine.

Acknowledgements

First, I would like to thank my supervisor, Dr. Ming Liang, for his constructive guidance, insightful suggestions, his careful and patient examination of my thesis, and his financial support. Without him, it is impossible for me to complete this study.

I also want to express my appreciation to those academic staff who gave their friendly help, and to the members of writing help center who provided their instrumental ideas about my thesis.

In addition, I would like to thank my beloved wife, Bin Gong, for her faithful and spiritual support on my study, for her great help in thesis writing, for the sacrifice she has made for me.

Table of Contents

List of Figures

List of Tables

Nomenclature

Parameters:

CF_i	the total number of configurations corresponding to design i
CO_i	the total cost corresponding to design i
CP_i	the purchasing cost of an RMT corresponding to design i
CE_i	the exchange cost of the modules of an RMT corresponding to design i
CSt_i	the storage cost of an RMT's module warehouse corresponding to design i
CP_i^j	the purchasing cost of module j of an RMT corresponding to design i
CE_i^j	the exchange cost of module j of an RMT corresponding to design i
CSt_i^j	the storage cost of module j in the module warehouse corresponding to design i
E_i	the resultant homogeneous transformation matrix describing the error in the position of module i with respect to its ideal position
ERx_i	the maximum tools position error caused by interfaces among modules on the x-axis of the RMT as determined by selecting the ith candidate RMT design
ERy_i	the maximum tools position error caused by interfaces among modules on the y-axis of the RMT as determined by selecting the ith candidate RMT design
ERz_i	the maximum tools position error caused by interfaces among modules on the z-axis of the RMT as determined by selecting the ith candidate RMT design
Mb_j	the number of basic modules in design i
Ml_j	the number of auxiliary modules in design i
MN	number of modules in basic design
NI_i^j	the number of instances of module j in design i;
N^{al}	the number of alternative designs

${}^{i-1}T_i$	A homogeneous transformation matrix representing the coordinate system transformation to the Cartesian coordinate system of module *i-1* from the system of the module *i*
${}^{i-1}T_i^{err}$	A homogeneous transformation matrix representing coordinate system transformation with error to the Cartesian coordinate system of module *i-1* from the system of the module *i*
V_i^{bas}	the number of variations of the basic modules' combinations in design *i*
V_i^{aux}	the number of variations of the auxiliary modules' combinations in design *i*
V_j^i	the number of instances of basic module *j* in design *i*
V'^i_j	the number of instances of auxiliary module *j* in design *i*
a_i , b_i , c_i	the position of the module *i* coordinate system's origin with respect to the coordinate frame of module *i-1* on x, y, z axes
c_1, c_2	learning factors
c_3	the passive congregation coefficient
m_i	the number of modules in the modules warehouse of design *i*
n_i	the number of main-chain modules, which are connected in a series from the workpiece to the cutting tool, in the design *i*
o^i_{ux} , o^i_{vx} , o^i_{wx}	direction cosines (unit vectors u, v, w) representing the orientation of the Cartesian direction (x-axis) of module *i* with respect to that of module *i-1*.
o^i_{uy} , o^i_{vy} , o^i_{wy}	direction cosines (unit vectors u, v, w) representing the orientation of the Cartesian direction (y-axis) of module *i* with respect to that of module *i-1*.
o^i_{uz} , o^i_{vz} , o^i_{wz}	direction cosines (unit vectors u, v, w) representing the orientation of the Cartesian direction (z-axis) of module *i* with respect to that of module *i-1*.
r_1, r_2, r_3	uniform random sequence in range (0,1)

ω	inertia term

Variables:

mc_{ij}	the element at row i and column j in module conjunction matrix
mi_{ij}	the element at row i and column j in module instance-requirement matrix
t_x	the location point of tools along x axis
t_y	the location point of tools along y axis
t_z	the location point of tools along z axis
t'_x	the relative distance of the movement of the tool along x axis
t'_y	the relative distance of the movement of the tool along y axis
t'_z	the relative distance of the movement of the tool along z axis
x_i	a binary variable
x_k^d	the position of particle d at time k
ε_{xj}	the axis rotation error of module j at axis x
ε_{yj}	the axis rotation error of module j at axis y
ε_{zj}	the axis rotation error of module j at axis z

Constants:

$t_x'^{max}$	the maximum movement distance of tools along axis x
$t_y'^{max}$	the maximum movement distance of tools along axis y
$t_x'^{max}$	the maximum movement distance of tools along axis z
ε_{xj}^{max}	the maximum rotation error along axis x of module j
ε_{yj}^{max}	the maximum rotation error along axis y of module j
ε_{zj}^{max}	the maximum rotation error along axis z of module j
$\boldsymbol{I}$	unit matrix

Abbreviation

FCP	Fuzzy-Chebyshev programming
GA	Genetic algorithm

GP	Goal Programming
HS	Head Support
HTM	Homogeneous transformation matrix
MF	Main Frame (Base)
MFCP	Modified fuzzy-Chebyshev programming
MT	Motor
PF	Product Fixer
PSOA	Particle swarm optimization algorithm
REM	Rotation error matrix
RMS	Reconfigurable manufacturing systems
RMT	Reconfigurable machining tool
SP	Spindle
TR	Transmission
TVM	Transformation error vector
XD	X-slide Driver
XH	X-slide Hand-driver
XT	X-slide Table
YD	Y-slide Driver
YH	Y-slide Hand-driver
YT	Y-slide Table
ZD	Z-slide Driver
ZH	Z-slide Hand-driver
ZS	Z-slide Support
ZT	Z-slide Table

Chapter 1. Introduction

1.1 Reconfigurable manufacturing system (RMS) and reconfigurable machine tools (RMT)

The current fierce global competition, rapid development of process technologies and rapidly changing customer requirements and demands are forcing companies to constantly improve their manufacturing environment. The improvement focuses on production systems and related technologies. The new production systems and related technologies must be rapidly developed and adapted to meet the requirements of production changes, capacity adjustments, technological integration, and cost effectiveness (Mehrabi *et al* 2000a).

The concept of a reconfigurable manufacturing system (RMS) was introduced in response to these market changes (Koren and Ulsoy 1997). RMS is a new production system that is more flexible than the dedicated manufacturing system (DMS) and more productive and economically viable than the flexible manufacturing system (FMS).

As one of the most important elements of RMS, the concept of reconfigurable machine tools (RMT) was also introduced (Koren and Kota 1999). The fundamental characteristic of RMT, as indicated by Pérez (2004), is a modular construction that possesses reconfigurability-oriented modularity. Modules with reconfigurable modularity allow modifications to the structural and functional configuration of a machine so that significant changes in product form and demand can be adapted quickly.

In Europe, some RMT researchers suggest that RMT module design should follow a worldwide standard so that RMT modules can be used as basic elements in different machines, similar to the hardware standards applied to the computer industry. However, given that the development of an internationally accepted standard is difficult and time consuming, this suggestion still remains at the theoretical stage.

In North America, a study group at the Engineering Research Center (ERC) of the University of Michigan has focused on easily exchangeable RMT modules with a standard interface that can be applied to specific RMSs. This approach is much easier for implementing RMS/RMT applications than the suggestion by European researchers.

Based on the research results from ERC at the University of Michigan, Tri-Way Manufacturing Technologies Corp., a machine tool builder in Windsor, Ontario, developed a reconfigurable machining system for Federal-Moful, a company that is also located in Windsor, to make automotive connecting rods (DeGaspari 2002).

1.2 Outline of the work

Many articles discussed the significance of RMS and predicted that it would become one of the most important production systems in the near future (Rogers and Bottaci 1997, Mehrabi *et al* 2000a, Mellor 2002 and Mehrabi *et al* 2002). The methodology of RMT/RMS design is currently a hot topic in RMS research. Optimizing RMT design is one of the basic steps to materialize the benefits of RMS. Three factors, configurability, cost and accuracy, must be considered in optimization.

In this study, alternative designs, which can perform the same tasks, are to be generated to achieve the goal of optimization, and a method to evaluate alternative designs will be proposed. Then a multi-objective method will be developed to solve the optimization problem, with consideration of the three factors according to a given order of relative importance. Finally, an efficient algorithm will be applied to the problem to find the optimal design.

1.3 Organization of thesis

A brief introduction to reconfigurable manufacturing systems, reconfigurable machine tools and their prospects is provided in this chapter. A review of the studies that have been carried out so far by others in the RMS/RMT field is presented in Chapter 2. The limitations of previous work and the motivation of this study are also presented in Chapter 2.

Chapter 3 introduces a method to evaluate RMT designs with a consideration of three factors – configurability, cost and accuracy. A mathematical model for the evaluation purpose is also formulated.

Chapter 4 presents a modified fuzzy-Chebyshev programming (MFCP) method as well as a multi-objective mathematical model to simultaneously obtain best results with regard to the three factors mentioned above. A brief review of a few related approaches

for solving multi-objective problems is also provided. Additionally, two examples of multi-objective problems are illustrated in the same chapter to compare the results derived from MFCP with those from other existing methods.

An automatic mechanism to generate alternative RMT designs based on a basic RMT design is presented in Chapter 5. This mechanism will be used to evaluate the three factors for each alternative design generated.

In Chapter 6, an efficient algorithm, the particle swarm optimization algorithm (PSOA), is suggested to solve the optimization problem. Moreover, an analysis of the complexity of the optimization problem and the classification of the problem's complexity, as well as a review of recent research on PSOA is also provided.

In Chapter 7, an RMT boring machine is utilized as a case study to compare the results obtained from FCP with those from MFCP method. The convergence and efficiency of the algorithm, using different sets of parameters, and the final parameter choice for the algorithm are also discussed in Chapter 7.

Finally, Chapter 8 provides a summary of this study and outlines a few potential research directions in the RMT field.

Chapter 2. Literature Review

The goal of this thesis is to provide a methodology to optimize RMT design, including its module warehouse, which is a set of modules ready to use for exchanging modules in an RMT. As discussed in Chapter 1.2, this methodology has to incorporate three design criteria (configurations, cost and accuracy) in the RMT design, and thus, apply a multi-objective approach to the optimization process. Given that RMT design is an essential part of RMS design and has a significant impact on the performance of an RMS, a review of the most important studies in RMS, RMT and multi-objective approaches is indispensable.

2.1 Reconfigurable manufacturing system (RMS)

A reconfigurable manufacturing system (RMS), according to the definition provided by Koren *et al* (1997), "is designed at the outset for rapid change in structure as well as in the hardware and software components." Koren *et al* (2002) also proposed methodologies for systematic RMS design considering production capacity change, reconfiguration and ramp-up. His related work has been patented in 2002. The patent integrated the essential elements from all achievements in RMS research at the Engineering Research Center (ERC) at the University of Michigan and shaped RMS into a valuable manufacturing design tool.

Before obtaining the patent, Koren *et al* (1999) systematically described the features of RMS/RMT, the differences among RMS, DMS and FMS, and the advantages of RMS in one of their publications. In the same publication, they presented the technologies enabling reconfiguration and other technical developments in RMS research. These technologies and developments, such as modular design and customized machine design, revealed a trend toward design of systems with reconfigurable software and hardware, as well as flexible system capacity and functionality. Finally, Koren *et al* also outlined RMS research issues, including future challenges, which became a valuable reference for researchers in related fields.

As described by Koren *et al* (1999), the core of the RMS paradigm is an approach to reconfiguration based on the design and integration of reconfigurable machine tools and

open-architecture reconfigurable controllers. Hence, RMS design can be separated into two aspects: software design and hardware design.

Regarding hardware design, Asl *et al* (2000) provide a dynamic simulation model for system-level RMS design. This model offers a fluid dynamic system analogy to demonstrate key concepts and ideas of RMS, such as reconfiguration and its relationship to scheduling. This model became a theoretical basis for product scheduling through using RMS.

Based on Koren *et al*'s achievements, Abdi and Labib (2003) used a fuzzy analytical hierarchical process to design an RMS, while Maier-Speredelozzi *et al* (2002) as well as Abdi and Labib (2004) applied an analytic hierarchy process (AHP) method to the selection of the RMS configuration and the optimization of RMS design, respectively. These approaches provided feasible solutions for RMS design optimization, including configuration selection, machine tools position and process determination.

Compared to the methods for RMS hardware design, more methods have been adapted to RMS software components to increase their flexibility and facilitate upgrades. Mehrabi *et al* (2000b) suggested that the RMS control system should become a modular open-architecture control system. Based on this, many promising studies have been conducted on the modular control (Koren *et al* 1996, Chen *et al* 1998 and Park *et al* 2000), the network control system for RMS (Lian *et al* 2000, Otanez *et al* 2002) and the software agent technology in intelligent control (Brucolleri *et al* 2003).

It should be noted that these efforts in RMS software study have had a significant impact on RMT control. In fact, the connection between RMS and RMT control system design is so strong that it is difficult to study the two separately. This is why RMT and RMS control systems are considered as one system in most research. In other words, the RMT control system is usually treated as a subsystem or even a component of the RMS control system. As a result, RMT control system design normally is not regarded as a part of RMT design.

2.2 Reconfigurable machine tools (RMT) design

RMT design is often referred to as RMS hardware design, even though RMS hardware design consists of many other aspects such as machine tool positioning.

As Koren and Ulsoy (2002a) point out, cost, quality and responsiveness are the three foundations on which every manufacturing company stands. As for an RMT, quality and responsiveness rely on processing accuracy and configurations, respectively. Hence, most studies on RMT hardware design mainly focus on these three aspects (cost, configuration and accuracy).

2.2.1 Reconfigurability of RMT

Given that RMT modular design is one of the crucial steps in achieving RMS reconfigurability, many recent studies on RMT design concentrate on seeking a method to systematically design modular and reconfigurable machines. One effort in RMT design has been to develop a method to modularize the basic components of machine tools. Walczyk *et al* (1998) proposed a reconfigurable tool used in forming processes. Cecil (2001) suggested adopting a computer-aided method to design a modular fixture. Pérez et al (2004) applied a concurrent design reference model to RMT development. Although those methods are capable of solving specific component modular design problems and the last method provided by Pérez et al can even modularize all components of RMT, no solution so far has been offered to systematically design an RMT with all possible configurations.

The Engineering Research Center (ERC) at the University of Michigan has made great contributions to RMT design. Moon and Kota (2002a) developed a generalized kinematics model of RMT. Koren *et al* (1999) obtained a patent on an RMT, a machine tool having customized flexibility and reconfigurability, based on their research on a Virtual Arch Type Reconfigurable Machine Tool (Katz and Moon 2000). Landers *et al* (2000, 2001), Moon and Kota (2002b) and Moon *et al* (2001) provided a methodology for systemic design of reconfigurable machine tools by using the screw theory for kinematics and graph theory structural synthesis. Figure 2-1 includes an overview of the RMT kinematic design methodology.

In the RMT kinematic design methodology, a machining operation is transformed into a task matrix (e.g. a homogeneous transformation matrix). Then, graph representations of candidate machine tools, which are generated in accordance with the functional requirements of the machining operation, are used to describe the structural

configuration. By verifying with the task matrix, modules of an RMT are then selected from a parameterized module library, which includes many modules with formal and unified representation schemes of mechanical functions. The product of HTMs of selected modules is compared to the task matrix. If these matrices are equal, the machine tool is kinematically viable. From this process, all possible configurations can be determined. Finally, the viable configurations are examined by using other criteria (e.g. degree of freedom, static and dynamic stiffness). As a result, a set of alternative RMT designs can be automatically generated by applying this methodology.

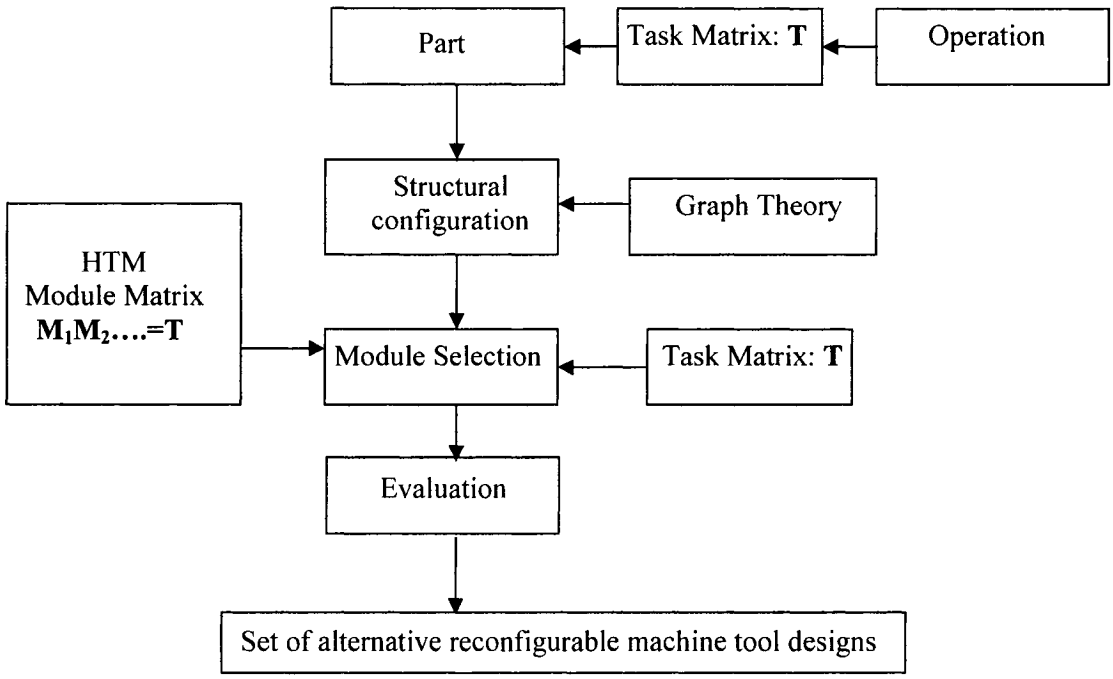


Figure 2-1 Overview of RMT kinematic design methodology

As shown in the above approach, new modules that have the same interface as that of one or more of existing modules can be easily added to a module library, thereby increasing design viability. Modular construction of an RMT ensures the reconfigurability of the machine tools. However, the inadequacies of the approach are also obvious.

First, modules in the current module library are not optimal. Normally, modules in a module library are the basic components of an RMT, and most of them have only one

function. In fact, most suppliers can provide relatively complex modules that have more than one function. These complex modules are less expensive than those in a current module library that perform the same function as the complex modules do.

Second, the above approach does not deal with module optimization, which is a critical step to reduce the difficulties associated with exchanging modules on an RMT and thereby shorten the reconfiguration time. For example, module merging, which is one part of module optimization, can reduce the numbers of modules in the reconfiguration process and thus reduce the reconfiguration time. Also, a shortened reconfiguration time through module warehouse optimization enhances responsiveness of an RMT. As responsiveness is a critical characteristic of the RMS/RMT, the time for reconfiguring a product line and machine tools should be limited to less than 2 hours (Heisel and Meitzner 2003). It should be pointed out that, in this study, "module warehouse" refers to a set of modules which are ready to use for exchanging modules in an RMT and associated with a specific RMT.

2.2.2 Cost of RMT

The cost of an RMT consists of the initial cost and operation costs. Many studies have been focused on operation costs, especially processing cost. Practical approaches to reduce the processing cost in machine tools design include but are not limited to carefully selecting the process sequence (Zhang and Wang 1993, Yeo and Hgoi 1996 and Ming and Mak 2001), machining parameters (Juan *et al* 2003 and Hui *et al* 2001), and tolerance design (Wu *et al* 1988, Yeo and Hgoi 1996 and Diplaris and Sfantsikopoulos 2000), or optimizing two or both of them (Wang 2003).

Given that the approaches previously discussed are able to eventually reduce the initial investment in machine tools, few researchers have paid attention to this area. Initial investments in RMT include both the investments in machine tools and in modules of the module warehouse. The latter is determined by the configurations of an RMT. According to the latest information from industry, the initial cost of an RMT is estimated to be 40 percent more than that of current CNC machine tools (Cole 2004). For this reason, reducing the initial investment cost of an RMT is still an essential part of RMT design, as it can considerably increase companies' acceptance and willingness to use RMS/RMT.

2.2.3 Accuracy of RMT

The geometric accuracy of an RMT is one of the critical factors in RMT design. Machine tool geometric errors create part geometric errors and thus indirectly affect part quality.

Moon *et al* (2000) developed a systematic methodology to analyze the accuracy of an RMT. In the methodology, all errors, including geometric module errors and assembly errors, can be identified and represented statistically. These representations are mathematically described using an error model such as Homogeneous Transformation Matrix (HTM). Then, simulations are performed to analyze the distribution of the tool position errors, given the statistical distribution of module and assembly errors. The resultant data from the simulations can be used for performance evaluation or post processing. An overview of the methodology is shown in Figure 2-2.

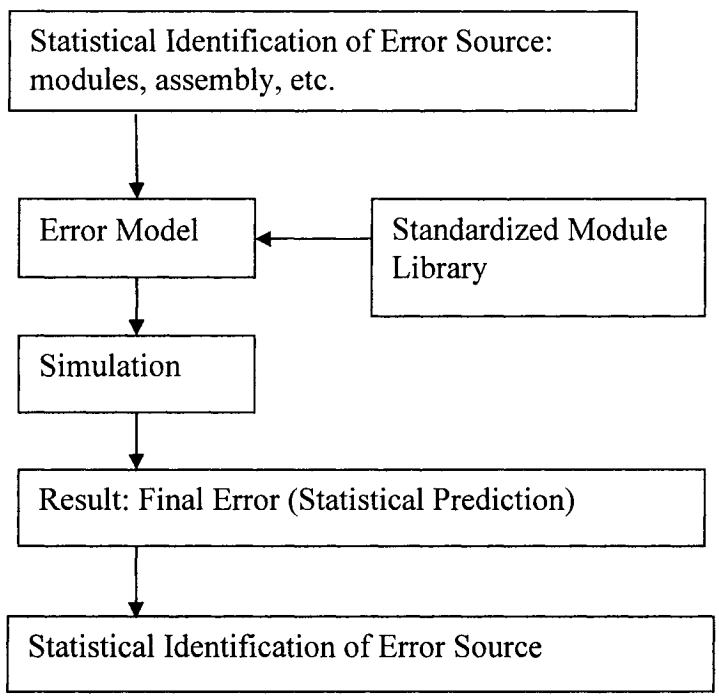


Figure 2-2 RMT error evaluation methodology

Researchers represented by Okafor *et al* (2000) suggested applying rigid body kinematics to RMT error model derivation. They adopted the HTM to represent the module error and calculated the tool position error by using rigid body kinematics. Other researchers, such as Moon *et al* (2001), proposed using screw theory to generate an error model. In that model, a matrix exponential with a unit screw and scalar, which is utilized to express the module error and tool tip error, is evaluated through applying screw theory. Though these are two different error models, both can be adopted to efficiently evaluate the accuracy of an RMT and conduct error compensation.

Nevertheless, these methods are only employed to examine and control the accuracy of an RMT within a required range. No publications so far have indicated that the error evaluation approaches have been applied to an RMT and its modules design, taking the requirement of reconfigurability into account.

2.3 Multi-objective approach to optimization

As discussed in Chapter 1.3, in this study, RMT design optimization is a multi-objective issue. Given the non-commensurability of the three aspects in RMT design, the multi-objective issue cannot be transformed directly into a single objective problem by simply adding a coefficient to each objective.

Ignizio and Cavalier (1994a) suggested applying the fuzzy-Chebyshev programming (FCP) method, a modified Chebyshev goal programming method, to the multi-objective programming problem. This method involves a fuzzy membership function to ensure that equal consideration is given to all sub-objectives, and transforms the multi-objective problem to a single objective problem without adding any coefficient. The same method was also presented by Mohanty and Vijayaraghavan (1995), as well as by El-Wahed and Abo-Sinna (2001). The general model of fuzzy-Chebyshev programming method can be written as follows.

Max δ

Subject to:

$$Z_i - d_i\delta \le L_i, (\text{or } \delta \ge (Z_i - L_i)/d_i) \qquad \text{for all } i \text{ objectives} \qquad (2\text{-}1)$$

$$\mathbf{Ax}(\le, \ge, \text{or} =)\mathbf{b} \qquad \text{for all } m \text{ constraints}$$

δ and all x_j are nonnegative

where,

Z_i is the function that represents the i^{th} objective

δ is the dummy variable that represents the worst deviation level

U_i is maximum value that Z_i can take on (through the solutions by individually solving all the single objective problems associated with all the objectives)

L_i is minimum value that Z_i can take on (through the solutions by individually solving all the single objective problems associated with all the objectives)

$d_i = U_i - L_i$

However, when using this method, decision makers have to treat all objectives as being equally important and hence the fuzzy-Chebyshev programming method cannot be applied to issues where some objectives are regarded as more essential than others.

Alternatively, Mohanty and Vijayaraghavan (1995), as well as El-Wahed and Abo-Sinna (2001), incorporated the conflict and non-conflict matrix concept into fuzzy programming to resolve multi-objective issues. This approach can be generally described by following steps for a general model of a multi-objective problem:

Max

$$Z_i = \sum C_{ik} x_{ik} \quad . \qquad i=1, 2, \ldots, m; \; k=1, 2, \ldots, n$$

Subject to

$\mathbf{Ax}(\leq, \geq, \text{or} =)\mathbf{b}$

Step 1: Generate a non-conflict matrix through the following procedure.

a) Calculate the angle θ_{ij} between Z_i and Z_j by:

$$\cos\theta_{ij} = \frac{\sum_{k=1}^{n} C_{ik} C_{jk}}{\sqrt{\sum_{k=1}^{n} C_{ik}^2 \sum_{k=1}^{n} C_{jk}^2}} \tag{2-2}$$

b) Compute non-conflict degree, η_{ij}, between Z_i and Z_j according to:

$$\eta_{ij} = \begin{cases} 1 & \theta_{ij} = 0 \\ \dfrac{(\pi - \theta_{ij})}{\pi} & 0 \leq \theta_{ij} \leq \pi \\ 0 & \theta_{ij} = \pi \end{cases} \tag{2-3}$$

c) Generate the non-conflict matrix in the form as shown below:

	Z_1	Z_2	...	Z_m
Z_1	1	η_{21}	...	η_{1m}
Z_2	η_{12}	1	...	η_{2m}
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Z_m	η_{m1}	η_{m2}	...	1

Step 2: Calculate the value W_i based on:

$$W_i = \frac{\sum_{j=1}^{k} \eta_{ij}}{k} \tag{2-4}$$

Then, set the priority order of all objectives based on the values of W_i's. The higher value of W_i, the higher the priority of the i^{th} objective.

Step 3: Obtain a new aspiration level of each objective by applying the following equation:

$$\mu^{-1}(Z_i) = W_i \tag{2-5}$$

where,

$$\mu(z_i) = \begin{cases} 1 & z_i \geq z_i^{\max} \\ \dfrac{z_i - z_i^{\min}}{z_i^{\max} - z_i^{\min}} & z_i^{\max} \leq z_i \leq z_i^{\min} \\ 0 & z_i \leq z_i^{\min} \end{cases}$$

Step 4: Based on the new aspiration level and the priority order acquired from steps 2 and 3, solve the problem using the traditional GP method.

Unlike the fuzzy-Chebyshev programming method, the conflict and non-conflict matrix method will yield order of priorities for the objectives based on the W_i values. However, such an order of priority is often inconsistent with the pre-specified and preferred order of importances. In addition, this method cannot be applied to non-linear problems. However, results derived from this method often lead to a higher average satisfaction than that from fuzzy-Chebyshev programming.

It should also be pointed out that both Goal Programming (GP), including Archimedean and non-Archimedean goal programming, and Analytic Hierarchy Process

(AHP) methods can be used in solving multi-objective problems (Ignizio and Cavalier, 1994b). In these methods, objectives are considered either equally important or at different importance levels. In goal programming method, when objectives belong to different levels, those objectives at high levels are considered overwhelmingly important. That is, they have to be achieved with all available resources before considering any objectives at lower levels. In the AHP, the weights representing the relationship among objectives must be arbitrarily preset directly or using a pairwise comparisons matrix. Both methods are based on the designers' experience and judgement. Hence, the weight setting process involves a subjective approach.

In view of the above, a more relevant approach must be developed to avoid the difficulties present in the methods described above.

2.4 Motivation

RMT design is different from traditional machine tool design. Reconfigurability, which determines the responsiveness of an RMT, is a critical factor. According to the most recent studies in RMT, reconfigurability (reflected by the number of configurations), accuracy and cost are the three crucial criteria of RMT design. Therefore, by carefully choosing modules in an RMT and its module warehouse and balancing the requirements of cost, accuracy and configuration, an RMT design, along with its module warehouse, can be optimized.

Unfortunately, as reviewed in Chapter 2.3, most studies on RMT design optimization focus on modular structure, control system integration and cost-efficient balancing. Although a few studies discuss the effect of the three aspects (cost, accuracy and configurations) on RMT design, studies on those three aspects are mostly conducted separately, thus leading to inconsistent and conflicting solutions. In addition, given the nature of RMS/RMT, reconfigurability should be considered more important than the other two aspects in the design of an RMT. Therefore, it is highly desirable to develop a model and solution method to deal with a broad range of multi-objective issues and obtain an optimal RMT design which considers the three aspects simultaneously without violating their priority order. An effort to develop such a method will be presented in this study.

Chapter 3. Analysis and Modelling of the RMT Design Problem

In this chapter, three RMT design factors (configuration, cost and accuracy) are analyzed and their respective models are developed. A multi-objective model that integrates the three factors is also formulated.

3.1 Problem statement

The purpose of RMT design and its module warehouse optimization is to build an RMT at low cost but with desirable configurability and process accuracy. Accordingly, an RMT design and its module warehouse should be evaluated according to three criteria: configurability, cost and accuracy. The design objectives are therefore to

1) Maximize the number of configurations of a machine;

2) Minimize the cost of all machine modules; and

3) Minimize the error of tool positions caused by the interface among machine modules, thereby improving accuracy of machine performance.

Due to the characteristics of RMT, different priories should be given to the different design objectives. Among the above three objectives, maximizing the number of configurations is the most (but not overwhelmingly) important. Here and also as stated in Chapter 2, the term "overwhelmingly" is used to indicate that an objective at a higher priority level is so important that it has to be achieved as much as possible using all available resources before the objectives of lower priority have any chance to be considered. Only the "leftover" resources can be used for the objectives of lower priorities. In the RMT design context, it is necessary to specify the order of importance of the three objectives but none of them is overwhelmingly more important than others. In this study, the priority order is assumed to be configurability, followed by cost and then by accuracy. Of course, the change of priority order will not change the methodology presented in this thesis. However, unlike in all the previous multi-objective problems where the relative importance is fixed, the priority in this study takes effect differently at different satisfaction levels of a high-priority objective. In other words, the relative

importance level varies with the satisfaction level. For example, when the configurability is too low (close to the lower limit), it is reasonable to pay "much" more attention to it. However, when the configurability approaches the upper limit, less consideration (but still more than that given to other objectives) should be given. The same consideration approach illustrated above also applies to lower level objectives. The quantitative treatment of the above satisfaction-dependent priority assignment will be presented in Chapter 4.

Three sub-problems associated with the three design objectives are as follows:

1) Configurability

Configurability refers to the flexibility of an RMT. It can be measured by the number of module combinations through selecting a module warehouse to which an RMT design corresponds. The number of module combinations also represents the number of RMT configurations.

Let CF_i represent the number of RMT configurations by selecting the i^{th} candidate RMT design. The higher the CF value, the higher the degree of the configurability. Therefore, the objective of optimizing the degree of configurability is:

$$\max CONF = \sum_{i=1}^{n} CF_i \cdot x_i \tag{3-1}$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{if the } i^{th} \text{ candidate design is selected} \\ 0 & \text{otherwise} \end{cases}$$

where n is the number of candidate (alternative) RMT designs.

2) Cost

Similarly, the lower the cost, the better the RMT design. Denoting CO_i as the cost of an RMT by selecting the i^{th} candidate RMT design, the objective of optimizing RMT cost is therefore:

$$\min COST = \sum_{i=1}^{n} CO_i \cdot x_i \tag{3-2}$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{if the } i^{th} \text{ candidate design is selected} \\ 0 & \text{otherwise} \end{cases}$$

3) Accuracy

Accuracy affects the quality of products processed by an RMT. The process accuracy of an RMT can be evaluated by estimating the tool position errors (*ER*s) in three Cartesian directions. The lower the *ER* values, the higher the process accuracy. Here, ERx_i, ERy_i and ERz_i represent the tool position errors caused by interfaces among modules on the x, y and z axes, respectively, of the RMT by selecting the i^{th} candidate RMT design. Consequently, the objectives of optimizing the degree of accuracy are:

$$\begin{aligned} \min ERRX &= \sum_{i=1}^{n} ERx_i \cdot x_i \\ \min ERRY &= \sum_{i=1}^{n} ERy_i \cdot x_i \\ \min ERRZ &= \sum_{i=1}^{n} ERz_i \cdot x_i \end{aligned} \tag{3-3}$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{if the } i^{th} \text{ candidate design is selected} \\ 0 & \text{otherwise} \end{cases}$$

3.2 Sub-problem models

3.2.1 The configurability model

Before discussing the configurability model for the RMT design, some important concepts have to be presented first.

Basic module: The basic module is the critical module in the construction of an RMT. Adding or removing any basic module will cause functional change to an RMT. An RMT must consist of all the basic modules. Each basic module can have only one instance in an RMT.

Auxiliary module: Auxiliary modules are the supplemental modules in the construction of an RMT. Adding or removing an auxiliary module will not cause any basic functional variations in an RMT, but may significantly improve or reduce the functional performance of an RMT.

Module instance: Module instances are modules that share the same module name and certain common features. All module instances of the same module must have an identical interface and a rotation error matrix (REM), which represents the axis rotation error in the coordinate system of the module, but may possess other parameters. In other words, if two instances have different interfaces or different REM, then they do not belong to the same module. The structure of an RMT module warehouse is shown in Figure 3-1.

Having defined the above concepts, the number of configurations in an RMT can be expressed as:

$$CF_i = V_i^{bas} \cdot V_i^{aux} \tag{3-4}$$

where,

V_i^{bas} is the number of variations of basic module combinations in design i

V_i^{aux} is the number of variations of auxiliary module combinations in design i

V_i^{bas} and V_i^{aux} are obtained by:

$$V_i^{bas} = \prod_{j=1}^{Mb_i} V_j^i \tag{3-5}$$

$$V_i^{aux} = \begin{cases} 1 & \text{if there is no luxury module in the design} \\ \prod_{j=1}^{Ml_i} (V_j'^i + 1) & \text{otherwise} \end{cases} \tag{3-6}$$

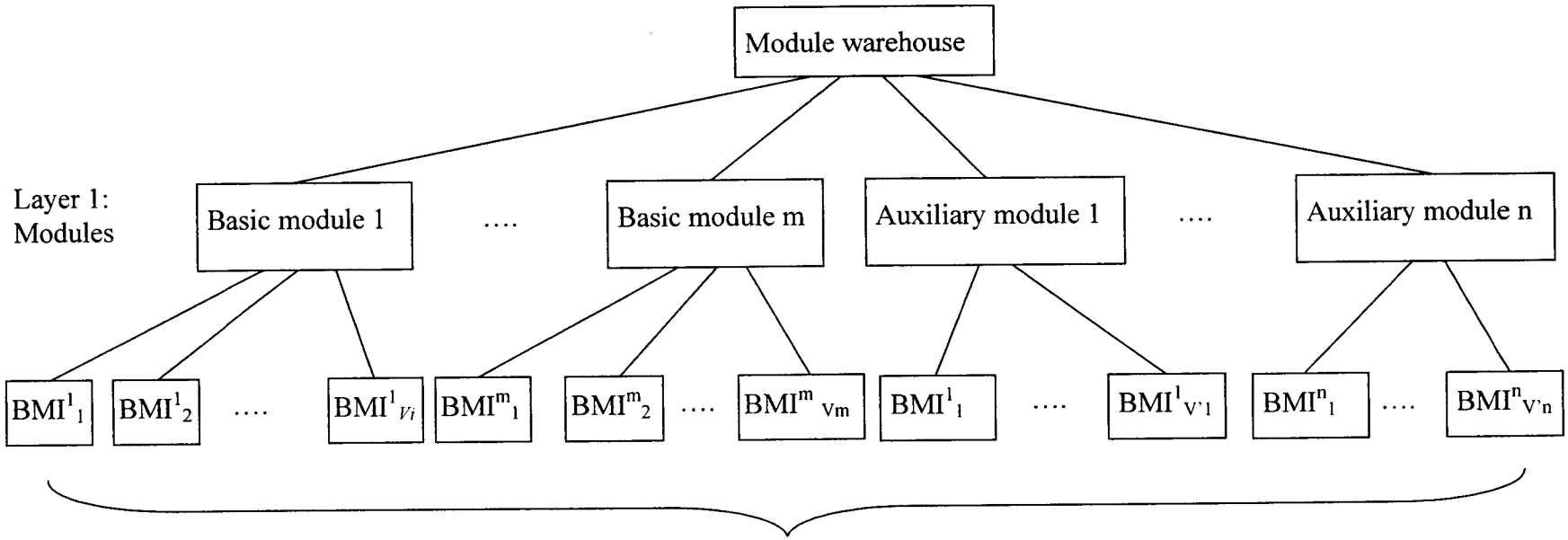
where,

V_j^i is the number of instances of basic module j in design i

V'^i_j is the number of instances of auxiliary module j in design i

Mb_i is the number of basic modules in design i

Ml_i is the number of auxiliary modules in design i



BMI^i_j is the j^{th} module instance of basic module i
AMI^p_q is the q^{th} module instance of auxiliary module p
Vi is the number of instances of basic module i
V'p is the number of instances of auxiliary module p

Figure 3-1 Structure of a module warehouse

3.2.2 The accuracy model

The homogeneous transformation matrix (HTM) representation of structures has existed for decades. Matrix ${}^{i-1}T_i$ is an HTM that represents the coordinate system transformation from the Cartesian coordinate system of module i to that of module i-1 obtained as follows:

$$ {}^{i-1}T_i = \begin{bmatrix} o_{ux}^i & o_{uy}^i & o_{uz}^i & a_i \\ o_{vx}^i & o_{vy}^i & o_{vz}^i & b_i \\ o_{wx}^i & o_{wy}^i & o_{wz}^i & c_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{i-1}R_i & {}^{i-1}B_i \\ 0 & 1 \end{bmatrix} \tag{3-7} $$

where ${}^{i-1}R_i$ is the axis rotation transformation matrix (ARTM) from module i to module i-1, and ${}^{i-1}O_i$ is the origin point transformation vector (OPTV) from module i to module i-1. They are given as:

$$ {}^{i-1}R_i = \begin{bmatrix} o_{ux}^i & o_{uy}^i & o_{uz}^i \\ o_{vx}^i & o_{vy}^i & o_{vz}^i \\ o_{wx}^i & o_{wy}^i & o_{wz}^i \end{bmatrix}, \quad {}^{i-1}B_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} $$

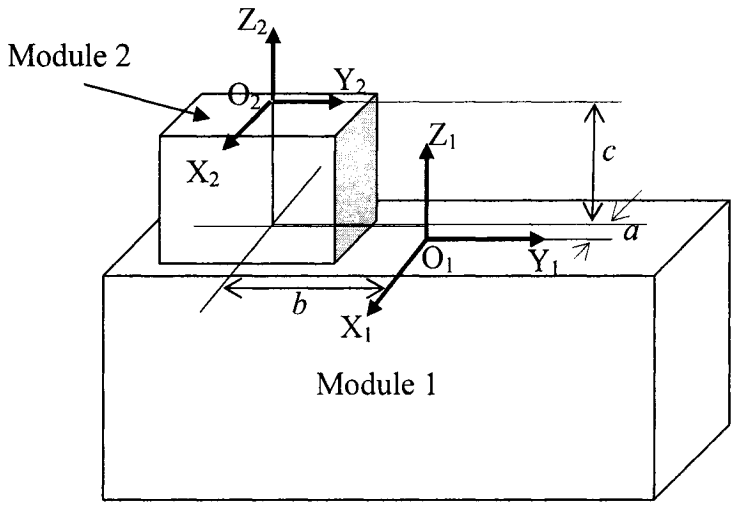


Figure 3-2 Two modules coordinate systems

For illustration, consider Figure 3-2. The HTM transformed coordinate system from module 2 to module 1 in the figure is:

$$ {}^{1}T_{2} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

If n modules are connected in series and the relative HTMs among modules are known as ${}^{R}T_{1}$, ${}^{1}T_{2}$, ..., ${}^{n-1}T_{n}$, the HTM of the top (n^{th}) module in terms of the reference coordinate system (R=0) can be expressed as:

$$ {}^{R}T_{n} = \prod_{i=1}^{n} {}^{i-1}T_{i} = \begin{bmatrix} \prod_{i=1}^{n} {}^{i-1}R_{i} & \sum_{j=1}^{n} \left(\prod_{i=0}^{j-1} {}^{i-1}R_{i} \right) \cdot {}^{j-1}B_{j} \\ 0 & 1 \end{bmatrix} \tag{3-8} $$

where ${}^{-1}R_{0} = \boldsymbol{I}$

Thus, the equivalent coordinates of a point in a coordinate frame n with respect to a reference frame R are

$$ \begin{bmatrix} X_{R} \\ Z_{R} \\ Y_{R} \\ 1 \end{bmatrix} = {}^{R}T_{n} \begin{bmatrix} X_{n} \\ Y_{n} \\ Z_{n} \\ 1 \end{bmatrix} $$

Slocum (1992) suggested that the HTM method be applied to the calculations of linear motion and error. Most of the 3 degrees of freedom (DOF) machine tool motions are linear in each coordinate system direction. Therefore, their method can be used to model process accuracy for most 3-DOF machine tools.

The resultant HTM that describes the error in the coordinate frame position of module i with respect to its ideal position is defined as E_i and expressed as:

$$ E_{i} = \begin{bmatrix} 1 & -\varepsilon_{zi} & \varepsilon_{yi} & \delta_{xi} \\ \varepsilon_{zi} & 1 & -\varepsilon_{xi} & \delta_{yi} \\ -\varepsilon_{yi} & \varepsilon_{xi} & 1 & \delta_{zi} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \Gamma_{i} & \Delta_{i} \\ 0 & 1 \end{bmatrix} \tag{3-9} $$

$$ \text{where } \Gamma_{i} = \begin{bmatrix} 1 & -\varepsilon_{zi} & \varepsilon_{yi} \\ \varepsilon_{zi} & 1 & -\varepsilon_{xi} \\ -\varepsilon_{yi} & \varepsilon_{xi} & 1 \end{bmatrix}, \Delta_{i} = \begin{bmatrix} \delta_{xi} \\ \delta_{yi} \\ \delta_{zi} \end{bmatrix} $$

In equation (3-9), ε_{xi}, ε_{yi} and ε_{zi} represent the rotation errors corresponding to axes x, y and z, respectively, in the coordinate system of module i. δ_{xi}, δ_{yi} and δ_{zi} refer to

transformation errors in the coordinate system's origin of module i along x, y and z, respectively. Γ_i is the rotation error matrix (REM) of module n whereas Δ_i is the transformation error vector (TEV). The values of those rotation errors are determined by the module's form tolerances.

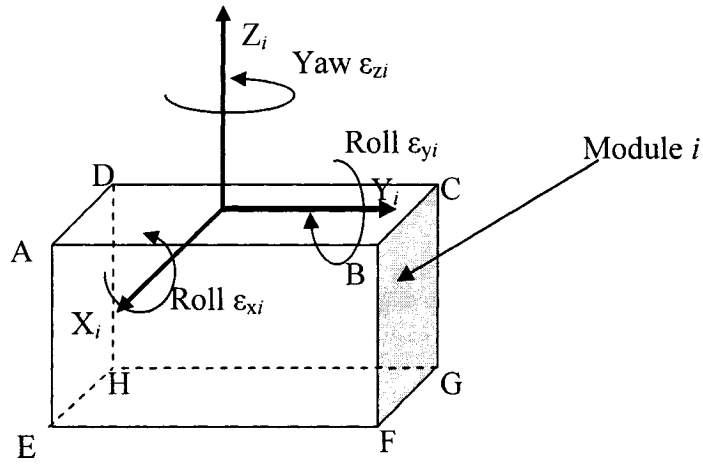


Figure 3-3 Axis rotation error of a module

For example, in Figure 3-3, the module's coordinate frame is based on plane ABCD and, hence, the tolerances on verticality or parallelism will affect the values of axis rotation error on both axis x and y (ε_{xi}, ε_{yi}). Also, ε_{zi} has the same value as that of the rotation error at axis z of the plane EFGH (caused by assembly dimension tolerance). For example, suppose that AB=300mm, BC=200mm, L_1=250mm, L_2=150mm, ΔL=0.01mm, and surface ABCD's tolerance related to EFGH on parallelism is 0.01mm. The connection type and dimension on surface EFGH are shown in Figure 3-4 (O is the ideal origin of the hole and O' is the real origin under the worst conditions). Then, under the worst conditions:

$$\varepsilon_{xi}^{\max} = \tan^{-1}(\frac{error}{AB}) = \tan^{-1}(\frac{0.01}{300}) = 3.33\times10^{-5} rad$$

$$\varepsilon_{yi}^{\max} = \tan^{-1}(\frac{error}{BC}) = \tan^{-1}(\frac{0.01}{200}) = 5.0\times10^{-5} rad$$

$$\varepsilon_{zi}^{\max} = \tan^{-1}(\frac{\sqrt{0.01^2+0.01^2}}{\sqrt{250^2+150^2}}) = \tan^{-1}(\frac{0.014}{291.55}) = 4.84\times10^{-5} rad$$

Or $|\varepsilon_{xi}| \leq 3.33\times10^{-5}$ rad, $|\varepsilon_{yi}| \leq 5.00\times10^{-5}$ rad, $|\varepsilon_{zi}| \leq 4.84\times10^{-5}$ rad.

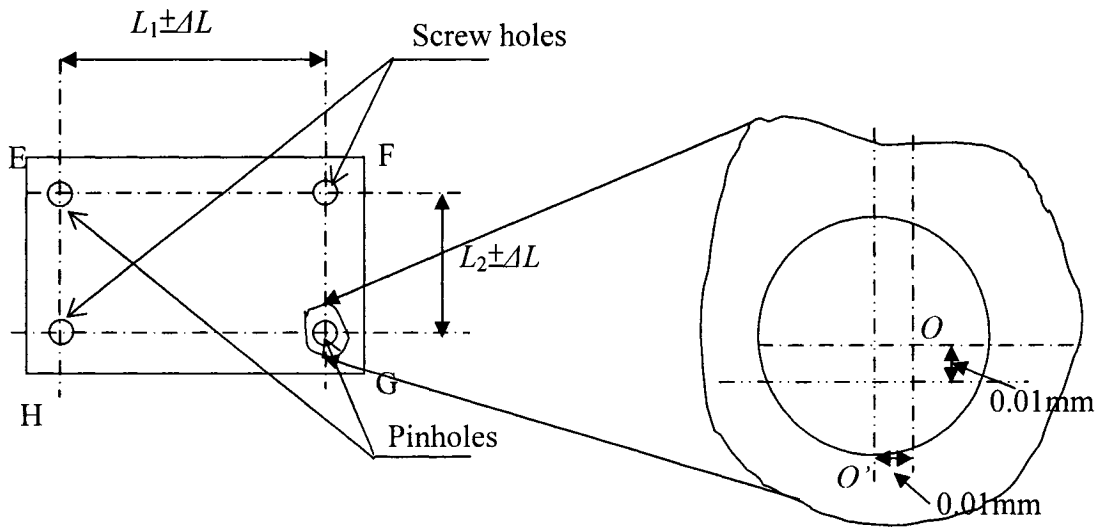


Figure 3-4 Dimension of flat EFGH

Suppose that modules i and i-1 are adjacent to each other. Defining ${}^{i-1}T_i$ as the HTM transform coordinate system from module i to module i-1, E_i as the resultant HTM that describes the error in the coordinate frame position of the module i with respect to its ideal position, the actual HTM with error transform coordinate system from module i to module i-1, ${}^{i-1}T_i^{err}$, can be described as a product of ${}^{i-1}T_i$ and E_i, i.e.,

$$\begin{aligned} {}^{i-1}T_i^{err} &= {}^{i-1}T_i E_i = \begin{bmatrix} {}^{i-1}R_i & {}^{i-1}B_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Gamma_i & \Delta_i \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{i-1}R_i\Gamma_i & {}^{i-1}R_i\Delta_i + {}^{i-1}B_i \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{3-10}$$

If n modules are connected in a series, based on the above principle, the actual HTM with error transform coordinate system from module n to reference coordinate system can be represented as:

$$\begin{aligned} {}^{R}T_n^{err} &= {}^{R}T_1^{err}\,{}^{1}T_2^{err}\,{}^{2}T_3^{err} \cdots {}^{n-1}T_n^{err} \\ &= \prod_{i=1}^{n} {}^{i-1}T_i^{err} = \begin{bmatrix} \prod_{i=1}^{n} {}^{i-1}R_i\Gamma_i & \sum_{j=1}^{n}\left(\prod_{i=0}^{j-1} {}^{i-1}R_i\Gamma_i\right)\left({}^{j-1}R_j\Delta_j + {}^{j-1}B_j\right) \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{3-11}$$

where Γ_0 and ${}^{-1}R_0$ are unit matrix $\boldsymbol{I}$.

When all the coordinate systems of the modules in the series are Cartesian coordinate systems and the axes of each coordinate system are correspondingly parallel, the axis rotation transformation matrix from module i to module i-1 is equal to unit matrix $\boldsymbol{I}$:

$$^{i-1}R_i = I \quad \forall i \; i = 1,2,3,\ldots,n$$

Thus, equation (3-11) reduces to:

$$^{R}T_n^{err} = \begin{bmatrix} \prod_{i=1}^{n}\Gamma_i & \sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j + {}^{j-1}B_j) \\ 0 & 1 \end{bmatrix} \tag{3-12}$$

Similarly, equation (3-8) becomes:

$$^{R}T_n = \prod_{i=1}^{n} {}^{i-1}T_i = \begin{bmatrix} I & \sum_{j=1}^{n} {}^{j-1}B_j \\ 0 & 1 \end{bmatrix} \tag{3-13}$$

The translational error of the tool position is the difference between the actual and ideal tool positions and can be expressed as:

$$\begin{bmatrix} \delta_{xt} \\ \delta_{yt} \\ \delta_{zt} \\ 0 \end{bmatrix} = \begin{bmatrix} X_t \\ Y_t \\ Z_t \\ 1 \end{bmatrix}_{\text{actual}} - \begin{bmatrix} X_t \\ Y_t \\ Z_t \\ 1 \end{bmatrix}_{\text{ideal}} \tag{3-14}$$

where the coordinates of the actual tool position in the reference coordinate system are given by:

$$\begin{bmatrix} X_t \\ Y_t \\ Z_t \\ 1 \end{bmatrix}_{\text{actual}} = {}^{R}T_n^{err} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} = \begin{bmatrix} \prod_{i=1}^{n}\Gamma_i & \sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j + {}^{j-1}B_j) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} \tag{3-15}$$

The coordinates of the ideal tool position in the reference coordinate system are given by:

$$\begin{bmatrix} X_t \\ Y_t \\ Z_t \\ 1 \end{bmatrix}_{\text{ideal}} = {}^{R}T_n \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} = \begin{bmatrix} I & \sum_{j=1}^{n} {}^{j-1}B_j \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \\ 1 \end{bmatrix} \tag{3-16}$$

Substituting equations (3-15) and (3-16) into equation (3-14) leads to:

$$\begin{bmatrix}\delta_{xt}\\ \delta_{yt}\\ \delta_{zt}\\ 0\end{bmatrix}=\begin{bmatrix}X_t\\ Y_t\\ Z_t\\ 1\end{bmatrix}_{\text{actual}}-\begin{bmatrix}X_t\\ Y_t\\ Z_t\\ 1\end{bmatrix}_{\text{ideal}}={}^{R}T_n^{err}\begin{bmatrix}t_x\\ t_y\\ t_z\\ 1\end{bmatrix}-{}^{R}T_n\begin{bmatrix}t_x\\ t_y\\ t_z\\ 1\end{bmatrix}$$

$$=\left\{\begin{bmatrix}\prod_{i=1}^{n}\Gamma_i & \sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j+{}^{j-1}B_j)\\ 0 & 1\end{bmatrix}-\begin{bmatrix}I & \sum_{j=1}^{n}{}^{j-1}B_j\\ 0 & 1\end{bmatrix}\right\}\begin{bmatrix}t_x\\ t_y\\ t_z\\ 1\end{bmatrix}$$

$$=\begin{bmatrix}\prod_{i=1}^{n}\Gamma_i-I & \sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j+{}^{j-1}B_j)-\sum_{j=1}^{n}{}^{j-1}B_j\\ 0 & 0\end{bmatrix}\begin{bmatrix}t_x\\ t_y\\ t_z\\ 1\end{bmatrix}$$

$$=\begin{bmatrix}(\prod_{i=1}^{n}\Gamma_i-I)\begin{bmatrix}t_x\\ t_y\\ t_z\end{bmatrix}+\sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j+{}^{j-1}B_j)-\sum_{j=1}^{n}{}^{j-1}B_j\\ 0\end{bmatrix} \tag{3-17}$$

Equation (3-17) can be rewritten as:

$$\begin{bmatrix}\delta_{xt}\\ \delta_{yt}\\ \delta_{zt}\\ 0\end{bmatrix}=\begin{bmatrix}P_1+P_2\\ 0\end{bmatrix} \tag{3-18}$$

where,

$$P_1=(\prod_{i=1}^{n}\Gamma_i-I)\begin{bmatrix}t_x\\ t_y\\ t_z\end{bmatrix} \tag{3-19}$$

$$P_2=\sum_{j=1}^{n}(\prod_{i=0}^{j-1}\Gamma_i)(\Delta_j+{}^{j-1}B_j)-\sum_{j=1}^{n}{}^{j-1}B_j \tag{3-20}$$

Equation (3-18) shows that the accuracy error of the tool position is the sum of P_1 and P_2.

P_1 is determined by two factors. One is the REM of all main-chain modules that are connected in a series from the workpiece to the cutting tool. Another factor is the moving distance of the cutting tool.

P_2 relies on REMs and HTMs of main-chain modules. As a result, each element of P_2 is a constant once the RMT is assembled. P_2 can thus be eliminated by resetting the origin point of the cutting tool.

Set point 1 at $[t_{x1}\ t_{y1}\ t_{z1}]^T$ as original point and point 2 at $[t_{x2}\ t_{y2}\ t_{z2}]^T$ as target point. The position error of the target point relative to point 1 is:

$$\begin{bmatrix}\delta_{xt}\\ \delta_{yt}\\ \delta_{zt}\\ 0\end{bmatrix}=\left(\begin{bmatrix}X_{t2}\\ Y_{t2}\\ Z_{t2}\\ 1\end{bmatrix}_{\text{actual}}-\begin{bmatrix}X_{t1}\\ Y_{t1}\\ Z_{t1}\\ 1\end{bmatrix}_{\text{actual}}\right)-\left(\begin{bmatrix}X_{t2}\\ Y_{t2}\\ Z_{t2}\\ 1\end{bmatrix}_{\text{ideal}}-\begin{bmatrix}X_{t1}\\ Y_{t1}\\ Z_{t1}\\ 1\end{bmatrix}_{\text{ideal}}\right)$$

$$=\begin{bmatrix}(\prod_{i=1}^{n}\Gamma_i-I)\begin{bmatrix}t_{x2}-t_{x1}\\ t_{y2}-t_{y1}\\ t_{z2}-t_{z1}\end{bmatrix}\\ 0\end{bmatrix} \tag{3-21}$$

Set

$$\vec{t}'=\begin{bmatrix}t'_x\\ t'_y\\ t'_y\end{bmatrix}=\begin{bmatrix}t_{x2}-t_{x1}\\ t_{y2}-t_{y1}\\ t_{z2}-t_{z1}\end{bmatrix} \tag{3-22}$$

then,

$$\begin{bmatrix}\delta_{xt}\\ \delta_{yt}\\ \delta_{zt}\\ 0\end{bmatrix}=\begin{bmatrix}(\prod_{i=1}^{n}\Gamma_i-I)\begin{bmatrix}t'_x\\ t'_y\\ t'_y\end{bmatrix}\\ 0\end{bmatrix} \tag{3-23}$$

In equations (3-19), (3-21) and (3-23)

$$\prod_{i=1}^{n}\Gamma_i=\begin{bmatrix}1 & -\varepsilon_{z1} & \varepsilon_{y1}\\ \varepsilon_{z1} & 1 & -\varepsilon_{x1}\\ -\varepsilon_{y1} & \varepsilon_{x1} & 1\end{bmatrix}\begin{bmatrix}1 & -\varepsilon_{z2} & \varepsilon_{y2}\\ \varepsilon_{z2} & 1 & -\varepsilon_{x2}\\ -\varepsilon_{y2} & \varepsilon_{x2} & 1\end{bmatrix}\cdots\begin{bmatrix}1 & -\varepsilon_{zn} & \varepsilon_{yn}\\ \varepsilon_{zn} & 1 & -\varepsilon_{xn}\\ -\varepsilon_{yn} & \varepsilon_{xn} & 1\end{bmatrix}$$

$$=\begin{bmatrix}1-\varepsilon_{z1}\varepsilon_{z2}-\varepsilon_{y1}\varepsilon_{y2} & -\varepsilon_{z1}-\varepsilon_{z2}+\varepsilon_{y1}\varepsilon_{x2} & \varepsilon_{y1}+\varepsilon_{y2}+\varepsilon_{z1}\varepsilon_{x2}\\ \varepsilon_{z1}+\varepsilon_{z2}+\varepsilon_{x1}\varepsilon_{y2} & -\varepsilon_{z1}\varepsilon_{z2}+1-\varepsilon_{x1}\varepsilon_{x2} & -\varepsilon_{x1}-\varepsilon_{x2}+\varepsilon_{z1}\varepsilon_{y2}\\ -\varepsilon_{y2}-\varepsilon_{y2} & -\varepsilon_{x1}-\varepsilon_{x2}+\varepsilon_{z1}\varepsilon_{y2} & 1-\varepsilon_{x1}\varepsilon_{x2}-\varepsilon_{y1}\varepsilon_{y2}\end{bmatrix}\begin{bmatrix}1 & -\varepsilon_{z3} & \varepsilon_{y3}\\ \varepsilon_{z3} & 1 & -\varepsilon_{x3}\\ -\varepsilon_{y3} & \varepsilon_{x3} & 1\end{bmatrix}\cdots\begin{bmatrix}1 & -\varepsilon_{zn} & \varepsilon_{yn}\\ \varepsilon_{zn} & 1 & -\varepsilon_{xn}\\ -\varepsilon_{yn} & \varepsilon_{xn} & 1\end{bmatrix}$$

The value of rotation error ε in each direction is so small ($\times 10^{-5}$) that the terms of higher order of product of ε's are negligible. Therefore, one obtains:

$$\prod_{i=1}^{n}\Gamma_i \cong \begin{bmatrix} 1 & -\sum_{i=1}^{n}\varepsilon_{zi} & \sum_{i=1}^{n}\varepsilon_{yi} \\ \sum_{i=1}^{n}\varepsilon_{zi} & 1 & -\sum_{i=1}^{n}\varepsilon_{xi} \\ -\sum_{i=1}^{n}\varepsilon_{yi} & \sum_{i=1}^{n}\varepsilon_{xi} & 1 \end{bmatrix} \tag{3-24}$$

Substituting equation (3-24) into equation (3-23) yields:

$$\begin{bmatrix} \delta_{xt} \\ \delta_{yt} \\ \delta_{zt} \\ 0 \end{bmatrix} = \begin{bmatrix} (\prod_{i=1}^{n}\Gamma_i - I)\begin{bmatrix} t'_x \\ t'_y \\ t'_y \end{bmatrix} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 & -\sum_{i=1}^{n}\varepsilon_{zi} & \sum_{i=1}^{n}\varepsilon_{yi} \\ \sum_{i=1}^{n}\varepsilon_{zi} & 0 & -\sum_{i=1}^{n}\varepsilon_{xi} \\ -\sum_{i=1}^{n}\varepsilon_{yi} & \sum_{i=1}^{n}\varepsilon_{xi} & 0 \end{bmatrix}\begin{bmatrix} t'_x \\ t'_y \\ t'_y \end{bmatrix} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} (-\sum_{i=1}^{n}\varepsilon_{zi})t'_y + (\sum_{i=1}^{n}\varepsilon_{yi})t'_z \\ (\sum_{i=1}^{n}\varepsilon_{zi})t'_x + (-\sum_{i=1}^{n}\varepsilon_{xi})t'_z \\ (-\sum_{i=1}^{n}\varepsilon_{yi})t'_x + (\sum_{i=1}^{n}\varepsilon_{xi})t'_y \\ 0 \end{bmatrix} \tag{3-25}$$

According to equation (3-25), the position errors of the target point in three directions, δ_{xt}, δ_{yt} and δ_{zt}, are:

$$\delta_{xt} = (-\sum_{i=1}^{n}\varepsilon_{zi})t'_y + (\sum_{i=1}^{n}\varepsilon_{yi})t'_z \tag{3-26}$$

$$\delta_{yt} = (\sum_{i=1}^{n}\varepsilon_{zi})t'_x + (-\sum_{i=1}^{n}\varepsilon_{xi})t'_z \tag{3-27}$$

$$\delta_{zt} = (-\sum_{i=1}^{n}\varepsilon_{yi})t'_x + (\sum_{i=1}^{n}\varepsilon_{xi})t'_y \tag{3-28}$$

Assuming that there are n_i main-chain modules in the i^{th} RMT design, the following equations that describe the position errors of the RMT based on the i^{th} design can be obtained through equations (3-26), (3-27) and (3-28).

$$\delta_{xt}^{i} = (-\sum_{j=1}^{n_i} \varepsilon_{zj}^{i})t_y' + (\sum_{j=1}^{n_i} \varepsilon_{yj}^{i})t_z' \tag{3-29}$$

$$\delta_{yt}^{i} = (\sum_{j=1}^{n_i} \varepsilon_{zj}^{i})t_x' + (-\sum_{j=1}^{n_i} \varepsilon_{xj}^{i})t_z' \tag{3-30}$$

$$\delta_{zt}^{i} = (-\sum_{j=1}^{n_i} \varepsilon_{yj}^{i})t_x' + (\sum_{j=1}^{n_i} \varepsilon_{xj}^{i})t_y' \tag{3-31}$$

Given that the final *ER*s of the RMT corresponding to the i^{th} design are calculated under the worst assembly conditions, the respective tool position errors caused by interfaces among modules on the x, y and z axes of the RMT through selecting the i^{th} candidate RMT design can be represented as:

$$ERx_i = \max \delta_{xt}^{i} = \max\left|(-\sum_{j=1}^{n_i} \varepsilon_{zj}^{i})t_y' + (\sum_{j=1}^{n_i} \varepsilon_{yj}^{i})t_z'\right| \tag{3.32}$$

$$ERy_i = \max \delta_{yt}^{i} = \max\left|(\sum_{j=1}^{n_i} \varepsilon_{zj}^{i})t_x' + (-\sum_{j=1}^{n_i} \varepsilon_{xj}^{i})t_z'\right| \tag{3-33}$$

$$ERz_i = \max \delta_{zt}^{i} = \max\left|(-\sum_{j=1}^{n_i} \varepsilon_{yj}^{i})t_x' + (\sum_{j=1}^{n_i} \varepsilon_{xj}^{i})t_y'\right| \tag{3-34}$$

Subject to:

$0 \le t_x' \le t_x{}^{,max}$

$0 \le t_y' \le t_y{}^{,max}$

$0 \le t_z' \le t_z{}^{,max}$

$|\varepsilon_{xj}^{i}| \le \varepsilon_{xj}^{i\,max} \quad \forall j = 1, 2, 3, \ldots, n_i$

$|\varepsilon_{yj}^{i}| \le \varepsilon_{yj}^{i\,max} \quad \forall j = 1, 2, 3, \ldots, n_i$

$|\varepsilon_{zj}^{i}| \le \varepsilon_{zj}^{i\,max} \quad \forall j = 1, 2, 3, \ldots, n_i$

where,

$t_x{}^{,max}$, $t_y{}^{,max}$, $t_x{}^{,max}$ refer to the maximum moving distance of tools along the x, y and z axes respectively

$\varepsilon^{i\ max}_{xj}$, $\varepsilon^{i\ max}_{yj}$, $\varepsilon^{i\ max}_{zj}$ refer to the maximum value of rotation errors with respect to axis x, y and z of module j in design i, respectively.

n_i is the number of main-chain modules in design i. These modules are connected in series from the workpiece to the cutting tool.

3.2.3 The cost model

The cost of a specific RMT design (including its module warehouse) consists of the module purchasing cost, the cost of exchanging a module from a module warehouse to the RMT (exchange cost), and the storage cost (space rental and maintenance costs) of the module warehouse. Therefore, the cost of an RMT corresponding to the i^{th} design can be expressed as:

$$CO_i = CP_i + CE_i + CSt_i \tag{3-35}$$

where,

CP_i is the purchasing cost of the RMT corresponding to design i;

CE_i is the exchange cost of the modules of the RMT corresponding to design i;

CSt_i is the storage cost of an RMT warehouse corresponding to design i

CP_i, CE_i and CSt_i can be respectively calculated as:

$$CP_i = \sum_{j=1}^{m_i} NI_i^j CP_i^j \tag{3-36}$$

$$CE_i = \sum_{j=1}^{m_i} (NI_i^j - 1) \cdot CE_i^j \tag{3-37}$$

$$CSt_i = \sum_{j=1}^{m_i} (NI_i^j - 1) CSt_i^j \tag{3-38}$$

where,

CP_i^j is the purchasing cost of module j corresponding to design i

CE_j^i is the exchange cost of module j corresponding to design i

CSt_i^j is the storage cost of module j corresponding to design i

m_i is the number of modules in the module warehouse corresponding to design i

NI_i^j is the number of instances of module j corresponding to design i

The exchange cost is determined by the number of interfaces that have to be separated or connected to the RMT as well as the weight and size of the module. It is mainly affected by the latter.

In equations (3-37) and (3-38), the exchange and storage costs of module *i* are only related to the number of remaining instances of module *i* in the module warehouse, as one instance of each module is already assembled in the RMT.

3.3 Multi-objective model of the problem

As indicated above, designing an RMT is a very complex task. Factors, such as configuration, accuracy and cost, have to be considered to obtain a design at low cost and with a large number of configurations and a high degree of accuracy. Therefore, the RMT design problem is a multi-objective issue. By integrating all sub-problem models presented above into one model, a multi-objective model can be written as follows:

$$\text{Max } CONF = \sum_{i=1}^{n} CF_i \cdot x_i = \sum_{i=1}^{n} V_i^{bas} \cdot V_i^{lux} \cdot x_i$$

$$\text{Min } COST = \sum_{i=1}^{n} CO_i \cdot x_i = \sum_{i=1}^{n} (CP_i + CE_i + CSt_i) \cdot x_i$$

$$\text{Min } ERRX = \sum_{i=1}^{n} ERx_i \cdot x_i = \sum_{i=1}^{n} \max\left(\left| (-\sum_{j=1}^{n_i} \varepsilon_{zj}^i) t_y' + (\sum_{j=1}^{n_i} \varepsilon_{yj}^i) t_z' \right| \right) \cdot x_i \qquad (3\text{-}39)$$

$$\text{Min } ERRY = \sum_{i=1}^{n} ERy_i \cdot x_i = \sum_{i=1}^{n} \max\left(\left| (\sum_{j=1}^{n_i} \varepsilon_{zj}^i) t_x' + (-\sum_{j=1}^{n_i} \varepsilon_{xj}^i) t_z' \right| \right) \cdot x_i$$

$$\text{Min } ERRZ = \sum_{i=1}^{n} ERz_i \cdot x_i = \sum_{i=1}^{n} \max\left(\left| (-\sum_{j=1}^{n_i} \varepsilon_{yj}^i) t_x' + (\sum_{j=1}^{n_i} \varepsilon_{xj}^i) t_y' \right| \right) \cdot x_i$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{if the } i\text{th candidate design is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$0 \le t_x' \le t_x{}^{,\max}$$

$$0 \le t_y' \le t_y{}^{,\max}$$

$$0 \le t_z' \le t_z{}^{,\max}$$

$$|\varepsilon_{xj}^i| \le \varepsilon_{xj}^{i\,\max} \quad \forall j = 1, 2, 3, \ldots, n_i$$

$$|\varepsilon^i_{yj}| \le \varepsilon^{i\,\max}_{yj} \quad \forall j = 1, 2, 3, \ldots, n_i$$

$$|\varepsilon^i_{zj}| \le \varepsilon^{i\,\max}_{zj} \quad \forall j = 1, 2, 3, \ldots, n_i$$

where the order of the importance of the objectives is *CONF*>*COST*>(*ERRX*, *ERRY*, *ERRZ*).

This model is the mathematical expression of the optimal RMT design problem. The constraints in the model, including all the constraints in the sub-problem models discussed previously, set limitations according to the physical features and capabilities of the RMT modules. By using a proper procedure to solve this model, an optimal RMT design that balances the three criteria can be obtained. The proper procedure will be discussed in the next chapter.

Chapter 4. Solution Procedure of the Multi-objective Model

In Chapter 3, RMT design was formulated as a multi-objective problem. It was also explained in Chapter 3 that none of the three objectives is "overwhelmingly" more important than others. In the previous studies, an objective of higher priority either was considered "overwhelmingly" more important than the ones of lower priority (e.g., in goal programming) or was allocated with a higher weight. The level of "overwhelmingness" and the magnitude of the weight will not change regardless of the satisfaction level of the high priority objective. In reality, it is obviously desirable to pay much more attention and allocate much more resource to the high priority objective when its satisfaction level is still low and there is much more room to improve. However, if its satisfaction level has approached the upper limit, it may not be worthwhile paying the same level attention to it and in the meantime disregarding the needs of other objectives. In such cases, instead of "over-feeding" the high-priority objective for a slight marginal improvement, it would make more economical sense to allocate the resources to lower level objectives with substantial return in satisfaction levels. This will require a satisfaction-dependent approach for priority weight assignment. It should be pointed out that with this approach the priority of the high-level objective is always higher than that of a lower-level one even when its satisfaction level approaches the upper limit. Such a satisfaction-dependent weight assignment approach has not been reported in the literature and the existing multi-objective optimization methods cannot be readily used for this purpose. A new method is therefore developed to generate the weight or quantify the level of importance for each objective based on its satisfaction level.

4.1 Modified fuzzy-Chebyshev programming (MFCP)

A general model of the multi-objective RMT design problem can be expressed as:

$$\text{Max } Z_k = f_k(\boldsymbol{X}) \qquad \forall k = 1, 2, \ldots, K$$

Subject to: (4-1)

$$\boldsymbol{AX}(\leq, \geq, \text{or} =)\boldsymbol{b}$$

where the priority order or order of importance is $Z_1 > Z_2 > ... > Z_K$ and the objective with a higher priority is more important (but not overwhelmingly so) than the objective with a lower priority.

As indicated in Chapter 2.3, most existing methods for solving multi-objective problems require arbitrary coefficients to define the relative importance of various objectives. Although goal programming (GP) method does not require this, it does consider the objective with a higher priority as overwhelmingly more important than the one with a lower priority. The fuzzy Chebyshev programming (FCP) and the non-conflict matrix methods do not have this requirement either. However, these two methods are used to handle objectives of equal importance and cannot be applied to deal with objectives in a pre-specified priority order.

As such, this chapter introduces a new method - modified fuzzy Chebyshev programming (MFCP). The use of the original fuzzy Chebyshev (FCP) method leads to achieving equal satisfaction level, δ, for all objectives. This implies that all objectives are of equally important and is obviously inconsistent with our intent that the objective should be prioritized in a pre-specified order. Hence, the FCP has to be modified to deal with this particular multi-objective problem. In doing so, the satisfaction levels of the objectives should follow an ascending order to reflect the priority order, i.e.,

$$\delta_1 > \delta_2 > ... > \delta_k > \delta_{k+1} > ... > \delta_K \tag{4-2}$$

To achieve this, a weight, w, selected in the range [0, 1) is used to define the relationship between objectives k and k+1 as:

$$\delta_{k+1} = w_k \delta_k \tag{4-3}$$

To avoid arbitrary weight selection, the value w_i must be systematically generated and should be dependent on the satisfaction level of each objective. More specifically, w_i must be a function of the k^{th} objective (i.e., $w_k = f(Z_k)$). $f(L_k)$ and $f(U_k)$ must be equal to 0 and 1, respectively, so that objective i can be given more consideration within a range of $[L_k, U_k]$ than objective k+1. This function ensures that the importance of objective k depends on the value of Z_k. Thus, the weight is systematically generated and self-adjusted. The weight should also be defined such that objective k is always more important than objective k+1 but the degree of superiority diminishes with the increase of value Z_k. This would help free up resources from high priority objectives when their

improvement margin becomes increasingly small. Under this situation, it would be more meaningful to allocate resources to lower-priority objectives for large increase of their satisfaction levels.

To satisfy the above requirements, the weight of the k^{th} objective, w_k, can be chosen, for example, as the following fuzzy function of the objective value (Note: other types of functions could also be used as long as they satisfy the above requirements):

$$w_k = f(Z_k) = \frac{Z_k - L_k}{U_k - L_k} \tag{4-4}$$

This relationship is depicted in Figure 4-1. As shown in the figure, the value of w_k (i.e., the ratio of δ_{k+1} to δ_k) increases linearly, in the range of [0,1), with Z_i. Hence, the relative importance level of objective k compared to objective k+1 drops accordingly as desired.

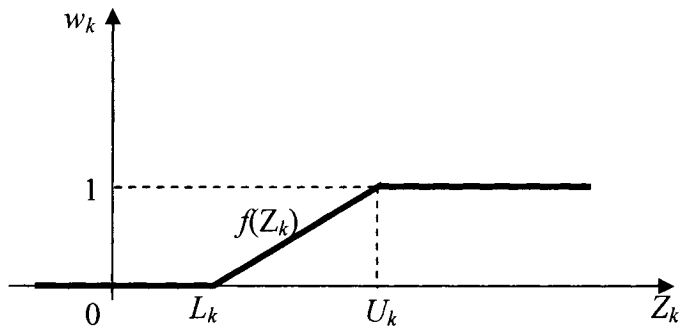


Figure 4-1 Fuzzy membership function of w_k

By using equations (4-3) and (4-4), the FCP can be converted to an MFCP. The conversion involves two steps. First, the objective must be changed from maximizing deviation levels of all objectives to maximizing deviation level of the most important objective. Then, the weights, which represent the relationship between the deviation levels of all objectives, must be defined using a fuzzy function. In this study, the linear fuzzy function defined in equation (4-4) is adopted to specify the weights. As a result, the multi-objective problem is transferred into a single objective problem.

By using the MFCP method, the general model presented above can be transformed as follows (It should be noted that there is only one MFCP problem for all the alternative designs):

$$\max \ \delta_1$$

subject to

$$\left.\begin{aligned} \delta_k &\le \frac{Z_k - L_k}{U_k - L_k} \\ \delta_{k+1} &= w_k \delta_k \\ w_k &= \frac{Z_k - L_k}{U_k - L_k} \end{aligned}\right\} \forall k = 1,2,3,...K \qquad (4\text{-}5)$$

$$\mathbf{Ax}(\le, \ge, \text{or } =)\mathbf{b}$$

where,

δ_k is the dummy variable that represents the deviation level of objective Z_k;

U_k is the maximum value that Z_k can take on (through the solutions by solving problem (4-1) as K individual single objective problems);

L_k is the minimum value that Z_k can take on (through the solutions by solving problem (4-1) as K individual single objective problems);

w_k is a fuzzy function that represents the relationship between best deviation level of objective Z_k and Z_{k+1}.

4.2 Comparison of MFCP, FCP and non-conflict matrix method

As discussed in Chapter 2.3, two popular approaches, the FCP and non-conflict matrix methods, are used by many researchers to solve the non-commensurable multi-objective problem. The studies by Mohantyand and Vijayaraghavan (1995) as well as El-Wahed and Abo-Sinna (2001) represent typical applications of such approaches. However, it has been indicated in Chapter 2.3 that the two approaches cannot solve the problems in this study. The differences among MFCP, FCP and non-conflict matrix methods are further illustrated using two examples as follows.

Example 4.1

maximize $Z_1 = 3x_1 + x_2$

maximize $Z_2 = 2x_1 + 3x_2$

Subject to:

$2x_1 + x_2 \leq 50$

$x_1 \leq 20$

$x_2 \leq 30$

and all x_j are nonnegative

It is specified in this example that objective Z_1 is more (but not overwhelmingly) important than Z_2.

The above problem can be written as follows using the MFCP approach:

max δ_1

subject to

$$\delta_1 \leq \frac{Z_1 - L_1}{U_1 - L_1} \tag{4-6}$$

$$\delta_2 \leq \frac{Z_2 - L_2}{U_2 - L_2} \tag{4-7}$$

$$\delta_2 = w_1 \delta_1 \tag{4-8}$$

$$2x_1 + x_2 \leq 50 \tag{4-9}$$

$$x_1 \leq 20 \tag{4-10}$$

$$x_2 \leq 30 \tag{4-11}$$

$$w_1 = \frac{Z_1 - L_1}{U_1 - L_1} \tag{4-12}$$

where $U_1 = 70$, U_2=110, $L_1 = 60$ and L_2=70.

The weight w_1 represents the relationship between δ_1 and δ_2. Weight w_1 can ensure that the deviation level δ_1 will be much greater than δ_2 if the optimized value of objective Z_1 is too far below the value of U_1. In other words, the relative importance of objective Z_1 relies on the value of Z_1 in the form of equation (4-12). The values of δ_1 and δ_2 are limited in the range of (0, 1) by constraints (4-6) and (4-7). Given that the value of Z_1 is within the range of (L_1, U_1), it can be easily concluded that the value of weight w_1 is in the range of (0, 1) based on equation (4-12).

In summary, the relative importance of Z_1 as compared to Z_2 is a function of the satisfaction level of itself. If Z_1 is far from being satisfied, more attention should be paid to it than to Z_2. If Z_1 is close to complete satisfaction, more attention should be given to objective Z_2 than to Z_1, since Z_2 is still too far from being satisfied.

By using FCP method, the example problem can be transformed into the following model:

max δ

subject to

$$\delta \le \frac{Z_1 - L_1}{U_1 - L_1} \tag{4-13}$$

$$\delta \le \frac{Z_2 - L_2}{U_2 - L_2} \tag{4-14}$$

$$2x_1 + x_2 \le 50 \tag{4-15}$$

$$x_1 \le 20 \tag{4-16}$$

$$x_2 \le 30 \tag{4-17}$$

where $U_1 = 70$, U_2=110, $L_1 = 60$ and L_2=70.

It can be seen from this model that, by using the FCP method, two objectives in the example problem have been assigned the same level of satisfaction.

The MFCP and FCP models of the problem in example 4.1 can be solved by using LINGO, a popular commercial software. The MFCP and FCP models for example 4.1 programmed by LINGO are shown in Appendix B and C correspondingly, and their results are listed in Tables 4-1 and 4-2, respectively.

Table 4-1 Results of example 4.1 using MFCP

Objective	Optimal	Satisfaction Degree $(Z_k\text{-}L_k)/(U_k\text{-}L_k)$ k=1,2
Z_1	66.18034	61.8%
Z_2	85.27864	38.19%

Table 4-2 Results of example 4.1 using FCP

Objective	Optimal	Satisfaction Degree $(Z_k\text{-}L_k)/(U_k\text{-}L_k)$ k=1,2
Z_1	65	50%
Z_2	90	50%

The results show that, after optimization using the MFCP method, the value of the 1st objective is 66.18, with a satisfaction degree of 61.8%, whereas the value of the 2nd objective is 85.28 with a satisfaction degree of 38.19% and the weight w is 0.618.

However, the satisfactory degrees of both objectives obtained using FCP method are 50%, which indicates that the two objectives are equally important. This is however against the pre-specified priority order, i.e., objective 1 should be more important than objective 2.

The procedure of the non-conflict matrix method for solving example 4.1 is as follows.

Step 1: Generate a non-conflict matrix

a) Calculate angle θ_{ij} between Z_1 and Z_2 by applying equation (2-2)

θ_{12} =0.661043

b) Compute non-conflict degree between Z_1 and Z_2 using equation (2-3)

$\eta_{12} = \eta_{21}$ =0.789583

c) Generate non-conflict matrix:

	Z_1	Z_2
Z_1	1	0.789583
Z_2	0.789583	1

Step 2: Calculate the value of W_i through equation (2-4):

W_1=0.8947

W_2=0.8947

The above process illustrates that if there are only two objectives in a problem, the method suggested by El-Wahed and Abo-Sinna (2001) always grants the two objectives the same degree of importance. The result is obviously inconsistent with the given relative importance order between the two objectives, i.e., Z_1 is more important than Z_2.

Example 4.2

Solving a problem with two objectives may not demonstrate the full advantage of the MFCP and all the limitations of the FCP and non-conflict matrix methods. Here a problem with three objectives is used in the following to further illustrate the differences among the three approaches.

Max

$$Z_1 = 4x_1 + x_2$$
$$Z_2 = 3x_1 + 2x_2$$
$$Z_3 = x_1 + 2x_2$$

subject to :

$x_1 + x_2 \leq 5$

$2x_1 + x_2 \leq 7$

$x_1 - 3 \leq 0$

$x_2 - 4 \leq 0$

$x_1, x_2 \geq 0$

It is assumed that the priority order of the three objectives is Z_1 followed by Z_2 and then Z_3. It should be understood that Z_i is not overwhelmingly more important than Z_{i+1}.

The above problem can be transformed into a single objective problem as follows by using the MFCP approach:

max δ_1

subject to

$$\delta_1 \leq \frac{Z_1 - L_1}{U_1 - L_1} \tag{4-18}$$

$$\delta_2 \leq \frac{Z_2 - L_2}{U_2 - L_2} \tag{4-19}$$

$$\delta_3 \leq \frac{Z_3 - L_3}{U_3 - L_3} \tag{4-20}$$

$$\delta_3 = w_2 \delta_2 \tag{4-21}$$

$$\delta_2 = w_1 \delta_1 \tag{4-22}$$

$$x_1 + x_2 \leq 5 \tag{4-23}$$

$$2x_1 + x_2 \leq 7 \tag{4-24}$$

$$x_1 - 3 \leq 0 \tag{4-25}$$

$$x_2 - 4 \leq 0 \tag{4-26}$$

$$x_1, x_2 \geq 0 \tag{4-27}$$

$$w_1 = \frac{Z_1 - L_1}{U_1 - L_1} \tag{4-28}$$

$$w_2 = \frac{Z_2 - L_2}{U_2 - L_2} \tag{4-29}$$

where U_1 =13, U_2 = 12, U_3 = 9, L_1 =8, L_2 = 11 and L_3 = 5.

The above single objective problem is again solved using LINGO. The related LINGO model is shown in Appendix D and the result is summarized in Table 4-3:

Table 4-3 Results of example 4.2 using MFCP

Objective	Optimal	Satisfaction Degree $(Z_k-L_k)/(U_k-L_k)$ k=1,2,3	Average Satisfaction Degree
Z_1	11.82782	76.56%	56.51
Z_2	11.58609	58.61%	
Z_3	6.758267	34.35%	

Table 4-4 Results of example 4.2 using FCP

Objective	Optimal	Satisfaction Degree $(Z_k-L_k)/(U_k-L_k)$ k=1,2,3	Average Satisfaction Degree
Z_1	11.26087	65.22%	65.22%
Z_2	11.86957	65.22%	
Z_3	7.608696	65.22%	

Table 4-4 is the result of example 4.2 by using the FCP in accordance with the model in Chapter 2.3. Its LINGO model is presented in Appendix E.

The above result shows that the value of the optimized objective Z_1 (=11.82782) with a satisfaction degree of 76.56% derived from using MFCP is greater than that (Z_1=11.2687), with a satisfaction degree of 65.22%, from using the fuzzy-Chebyshev method.

Following the procedure in Chapter 2.3, the problem in example 4.2 can also be solved using the non-conflict matrix method as follows:

In step 1, the angles θ_{ij} between Z_i and Z_j are calculated by applying equation (2-2):

θ_{12} =0.343024

θ_{13} =0.862170

θ_{23} =0.519146

Then, the non-conflict degrees between Z_i and Z_j are obtained via equation (2-3):

$\eta_{12} = \eta_{21}$ =0.890812

$\eta_{13} = \eta_{31}$ =0.725563

$\eta_{23} = \eta_{32}$ =0.834751

Finally, the non-conflict matrix is shown as below:

	Z_1	Z_2	Z_3
Z_1	1	0.890812	0.725563
Z_2	0.890812	1	0.834751
Z_3	0.725563	0.834751	1

In step 2, the values of W_i are computed according to equation (2-4):

W_1=0.8721

W_2=0.9085

W_3=0.8534

Based on above computations, the priority order for the last step is that Z_2 is more important than Z_1 which is further more important than Z_3.

The purpose of step 3 is to obtain a new aspiration level for each sub-objective. By using equation (2-5), the following result can be obtained:

Z_1=12.3605

Z_2=11.9085

Z_3= 8.4736

In the last step, according to the new aspiration level and priority order obtained in previous steps, the problem is solved by using the traditional GP method:

Lexmin $\{(\rho_1+\rho_2+\rho_3+\rho_4), (\eta_6), (\eta_5), (\eta_7)\}$

The above Lexmin satisfies the following conditions:

$$x_1 + x_2 + \eta_1 - \rho_1 = 5$$
$$2x_1 + x_2 + \eta_2 - \rho_2 = 7$$
$$x_1 + \eta_3 - \rho_3 = 3$$
$$x_2 + \eta_4 - \rho_4 = 4$$
$$4x_1 + x_2 + \eta_7 - \rho_5 = 12.3605$$
$$3x_1 + 2x_2 + \eta_6 - \rho_6 = 11.9085$$
$$x_1 + 2x_2 + \eta_5 - \rho_7 = 8.4736$$

x_j, η_i, ρ_i are nonnegative

The results of example 4.2 using the non-conflict matrix method are shown in Table 4-5.

Table 4-5 Results of example 4.2 using the non-conflict matrix method

Objective	Optimal	Satisfaction Degree $(Z_k-L_k)/(U_k-L_k)$ k=1,2,3	Average Satisfaction Degree
Z_1	11.1830	63.66%	74.21%
Z_2	11.9085	90.85%	
Z_3	7.7255	68.14%	

As indicated from the results, the order of satisfaction degree of the optimized objectives is $Z_2 > Z_3 > Z_1$. Both this order and the priority order obtained in Step 3 are different from the priority order given in the original problem in example 4.2.

In the example, an objective is given the highest priority using the non-conflict matrix method. As this objective has the least conflict with other objectives, the negative effect on others will be minimized when it is optimized. The non-conflict matrix method, therefore, reduces the aspiration level to ensure that the satisfaction degrees of optimized objectives are as close as possible. Consequently, the results from using the non-conflict matrix method are very close to those from using the FCP method but have a relatively higher average satisfaction degree.

Furthermore, the non-conflict matrix method generates as good results as possible for each optimized objective, nevertheless, it gives as equal as possible consideration to each one. It is notable that, in this method, the weights W_i's are not preset to describe the priority of each objective, but are used to calculate the new aspiration levels and to determine the solving order in the last step. Therefore, the result of the objective priority order from using the non-conflict matrix method does not follow the priority order stated in the preconditions of example 4.2. The non-conflict matrix method thus cannot accurately determine the order of relative importance among objectives.

4.3 Advantages of MFCP

Based on the discussion in Chapter 2.3 and the observations made in Chapters 4.1 and 4.2, a more general comparison among the GP, non-conflict matrix, FCP and MFCP methods can be made as summarized in the following table.

Table 4-6 Comparisons of GP, non-conflict matrix, FCP and MFCP methods

	GP	Non-conflict matrix method	FCP	MFCP
Weights setting for priority	Arbitrary	None	None	Automatically
Value of weight	Fixed	None	None	Dynamic and self-adjusted
Ability to solve polynomial problem	Yes	Yes	Yes	Yes
Ability to solve non-polynomial problem	Yes	no	Yes	Yes
Ability to solve the problem with priority order	Yes	no	No	Yes

Accordingly, the advantages of the MFCP method can be stated as follows:

- Compared to the GP method, MFCP method uses automatically generated weight coefficients, thus avoiding subjective influences on the weight setting.
- Rather than fixed, the weight coefficients are dynamic and self-adjusted. Therefore, resources (considerations) can be effectively distributed to the objectives.
- The MFCP method can be adopted to solve the multi-objective problem either with or without priority order (by setting weights as variables or equal to 1);
- The MFCP method can be applied to solve both the polynomial and non-polynomial problems.

4.4 Transformation of the multi-objective problem of the RMT design using MFCP

As presented in Chapter 3.3, the problem in this study can be expressed as:

$$\begin{cases} \max CONF = \sum_{i=1}^{n} CF_i \cdot x_i \\ \min COST = \sum_{i=1}^{n} CO_i \cdot x_i \\ \min ERRX = \sum_{i=1}^{n} ERx_i \cdot x_i \\ \min ERRY = \sum_{i=1}^{n} ERy_i \cdot x_i \\ \min ERRZ = \sum_{i=1}^{n} ERz_i \cdot x_i \end{cases} \tag{4-30}$$

Subject to:

$$\sum_{i=1}^{n} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{the } i\text{th candidate be selected} \\ 0 & \text{the } i\text{th candidate not be selected} \end{cases}$$

where n is the number of possible designs of an RMT.

In this problem, the priority order among sub-objectives is $CONF > COST > ERRX$, $ERRY$ and $ERRZ$ (Note: here symbol ">" implies "more important than"). Thus, there are three levels of sub-objectives in the problem:

Level 1 is to maximize the number of configurations;

Level 2 is to minimize the cost; and

Level 3 is to minimize the error in the three directions.

There is no level of objectives considered overwhelmingly more important than others. The relative importance of objectives between two levels relies on the satisfaction degrees of the objectives. At the same level, all sub-objectives are considered equally important.

Thus, the problem can be converted into the following by using the MFCP method:

max δ_1

subject to

$$\delta_1 \le \frac{CONF - L_{CONF}}{U_{CONF} - L_{CONF}}$$

$$\delta_2 \le \frac{COST - L_{COST}}{U_{COST} - L_{COST}}$$

$$\delta_3 \le \frac{ERRX - L_{ERRX}}{U_{ERRX} - L_{ERRX}}$$

$$\delta_3 \le \frac{ERRY - L_{ERRY}}{U_{ERRY} - L_{ERRY}}$$

$$\delta_3 \le \frac{ERRZ - L_{ERRZ}}{U_{ERRZ} - L_{ERRZ}} \qquad (4\text{-}31)$$

$$\delta_2 = w_1 \delta_1$$

$$\delta_3 = w_2 \delta_2$$

$$w_1 = \frac{CONF - L_{CONF}}{U_{CONF} - L_{CONF}}$$

$$w_2 = \frac{COST - L_{COST}}{U_{COST} - L_{COST}}$$

where,

U_{CONF}, U_{ERRX}, U_{ERRY}, U_{ERRZ} represent the respective maximum values that *CONF, ERRX, ERRY, ERRZ* can take on (through the solutions by solving problem as 5 individual single objective problems);

U_{COST} is either the maximum values that *COST* can take on (through the solutions by solving problem as 5 individual single objective problems) or the affordable cost that is decided by the designer or customer, whichever is lower;

L_{CONF}, L_{COST}, L_{ERRX}, L_{ERRY}, L_{ERRZ} represents the respective minimum values that *CONF, COST, ERRX, ERRY, ERRZ* can take on (through the solutions by solving problem as 5 individual single objective problems);

w_i is a fuzzy function that represents the relative importance relationship between level i and level i+1.

Chapter 5. Generation of Alternative RMT Designs

As described in Chapter 3, the optimal RMT design must be selected from a large number of alternative designs. However, it is very time-consuming and impractical to manually create alternative designs. Therefore, a method that can automatically generate the alternative designs is presented in this chapter.

5.1 Terminologies

Before introducing the method, some important concepts and definitions are described as follows.

Basic design: An RMT design using as many modules as possible to fulfill the functional requirements. Each module in this design cannot be further decomposed into sub-modules.

Alternative design: An RMT design derived from the basic design through merging modules and removing interfaces among modules.

Main-chain modules: The modules that are connected in a series from the workpiece to the cutting tool in an RMT. Connection errors of the interfaces among these modules will affect the tool position error.

Table 5-1 An example of a module conjunction matrix

	Module 0	Module 1	Module 2	… …	Module n-1
Module 0	0	1	0	… …	0
Module 1	1	0	1		
Module 2	0	1	0	… …	1
. . .	. . .	. . .	. . .	. . .	. . .
Module n-1	0	0	1	… …	0

Module conjunction matrix: A matrix that shows how RMT modules connect. In this matrix, mc_{ij} is the element at row i and column j. If module i and module j are directly connected through a common interface, the value of mc_{ij} is equal to "1."

Otherwise, the value of mc_{ij} is "0." Each design corresponds to one module conjunction matrix. An example of a module conjunction matrix is shown in Table 5-1.

Module instance-requirement matrix: A matrix that demonstrates the relationship between the requirements of product process, product design and productivity, and the number of instances of each module in an RMT design. In this matrix, mi_{ij} is the element at row i and column j. The value of mi_{ij} represents the number of instances of module j requested according to the i^{th} requirement in the first column in the matrix. Each design corresponds to one module instance-requirement matrix. Table 5-2 shows an example of a module instance-requirement matrix.

Table 5-2 An example of a module instance-requirement matrix

	Module 0	Module 1	Module 2	... ...	Module n-1
Future forecast	2	1	1	... ...	1
Requirement 1	1	1	1	... ...	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
Requirement *MR*	2	1	1	... ...	1

In the matrix, the first row ("Future forecast") shows the forecast numbers of each module's instances that may be required in the future, based on technology development and production capacity changes.

Module attribute matrix: A matrix that represents the attributes of all modules in a certain RMT design. Each design corresponds to one module attribute matrix. An example of a module attribute matrix is illustrated in Table 5-3.

Table 5-3 An example of a module attribute matrix

	Module 0	Module 1	Module 2	... ...	Module n-1
Index	0	1	2	... ...	n-1
Name	PF	XT	YT	... ...	MF
Main chain index	1	2	3	... ...	5
Basic module	1	1	1	... ...	1
Purchasing cost($) *CP*	400	2800	2800	... ...	20000
Storage cost($) *CE*	200	1500	1500	... ...	1200
Exchange cost($) *CSt*	20	55	100	... ...	1200
Error on x axis (x10^{-5}rad) ε_x^{max}	1	1	1	... ...	2
Error on y axis (x10^{-5}rad) ε_y^{max}	1	1	1	... ...	2
Error on z axis (x10^{-5}rad) ε_z^{max}	5	5	5	... ...	4
Number of instance *V* or *V'*	2	1	1	... ...	1

In the matrix, the sequence of *Index* starts from 0 to *n*-1, where *n* is the number of modules in a given design. These index numbers are first given to the modules in the main-chain as the order of the *Main chain index* sequence, and then are given to the module in branches under the main-chain module using the same order as the order of the *Main chain index* sequence.

The second row gives a brief name for each module.

The sequence of the *Main chain index* starts from 1 to the number of all modules in the main-chain. The index number is given to the main-chain module corresponding to its position in the order from the workpiece to the cutting tool. The module with the "0" *Main chain index* is not a main-chain module.

The value in the fourth row, *Basic module,* is either "1," which indicates the module is a basic module, or "0," which means the module is an auxiliary module.

The values in the rows of *Error on x axis*, *Error on y axis* and *Error on z axis* are obtained using the method discussed in Chapter 3.2.2.

The last row represents the number of instances for each module. This number is the product of the elements in the corresponding column of the module instance-requirement matrix.

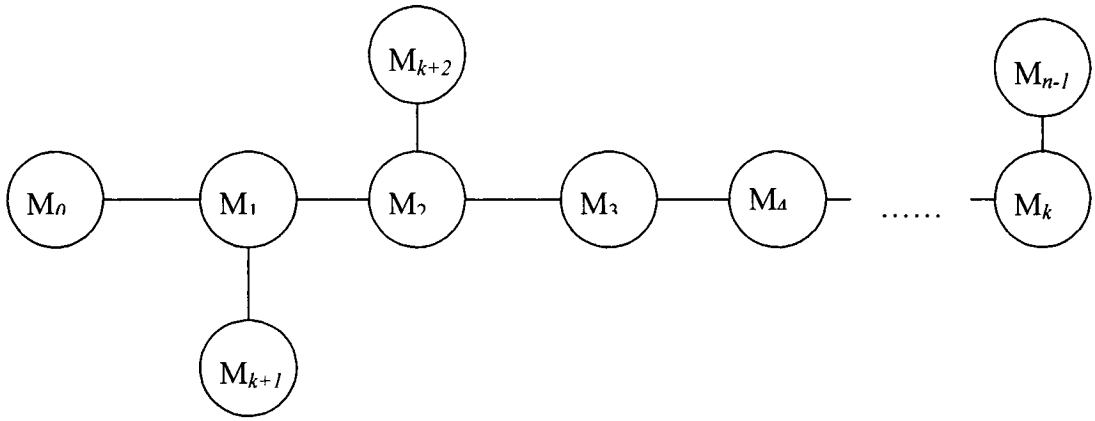
Basic interface vector: A vector whose elements' values are all equal to "1." The element I_{ij} represents the existing interface between modules i and j in a basic design, where i and j are the index numbers of modules. For example:

Basic interface vector: $(I_{ni}, \ldots, I_{54}, I_{43}, I_{32}, I_{21}, I_{10})$=(1,1,1,1,1,…,1)

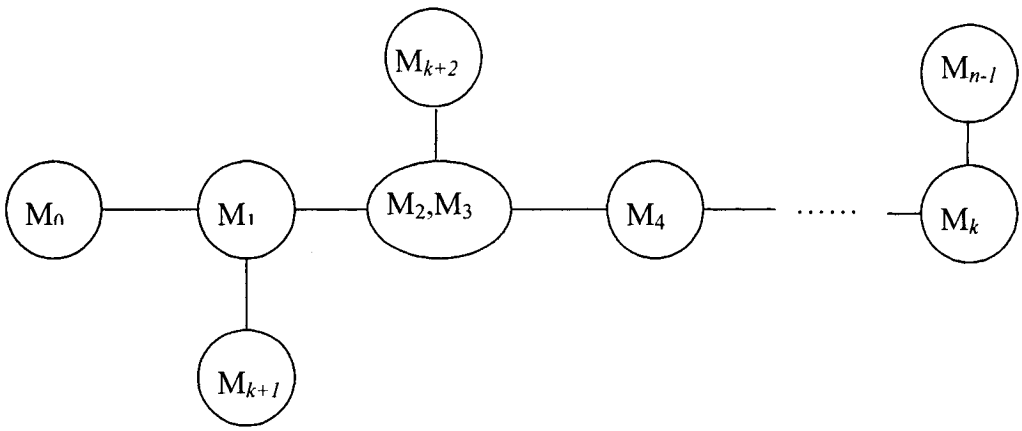
Interface vector: A vector derived from the basic interface vector. This vector has the same number of elements as basic interface vector does. The value of the interface vector elements is either "1" or "0." The interface vector can represent different RMT designs by selecting the value of the elements. These RMT designs do not cause any functional changes in the RMT, as they result from merging modules rather than adding or removing modules, and do not change the machine task matrix. If the vector is considered as a binary string, the interface vector can be expressed as a unique decimal number through the conversion from binary to decimal.

In the following example, element $I_{43} = 0$ means that module 3 and 4 are merged to become an alternative design. Each design corresponds to one interface vector. An example of an alternative design's interface vector is $(I_{ni}, \ldots, I_{54}, I_{43}, I_{32}, I_{21}, I_{10})$ = (1,…,1,1,1,0,1).

Topological representative of an RMT structure: An RMT can be illustrated in a topological structure graph. General topological structures representing a basic and an alternative RMT design are illustrated in Figure 5-1. Figure 5-2 shows the topological structure of an example RMT.



a) A basic design of an RMT



b) An alternative design of an RMT

represents interface between two modules

M_i represents module *i*

Figure 5-1 General topological structures of an RMT design

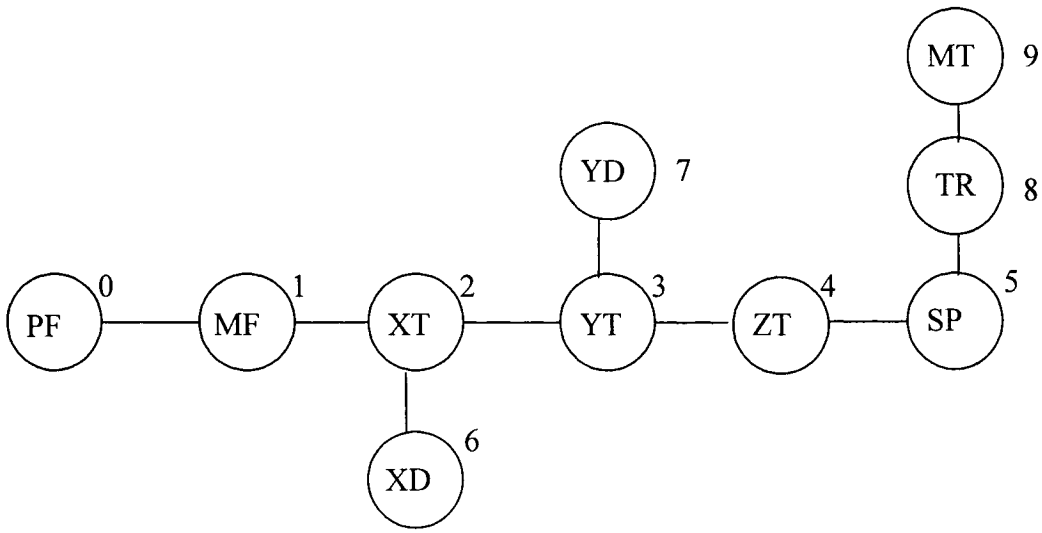


Figure 5-2 An example of the topological structure of an RMT design

5.2 Data processing of an RMT's alternative design

As introduced in Chapter 2.2, Lander *et al* (2001) and Moon *et al* (2001) have developed a methodology to design the RMT structure. By applying this methodology and setting all modules in the module library as fundamental modules that cannot be decomposed into sub-modules, the basic design and its interface vector can be derived. It should be noted that the generated interface vector is the basic interface vector. All new designs can be obtained by changing the value of elements in the basic interface vector. For example, it is assumed that the basic interface vector is (I_{65}, I_{54}, I_{43}, I_{32}, I_{21}, I_{10}) = (1,1,1,1,1,1). In a new interface vector, the fifth item is changed to "0" such that (I_{65}, I_{54}, I_{43}, I_{32}, I_{21}, I_{10}) = (1,1,1,1,0,1). Then, in the new design, the two affected modules, i.e., modules 1 and 2, are merged to become a new module, and the interface between them is eliminated. To simplify the computation, if module *i* and module *j* are to be merged into a new module, the module with the higher index number should be merged into the module with the lower index number. In other words, if $i<j$, module *j* will be merged into module *i*.

According to the new interface vector, a module conjunction matrix, a module instance-requirement matrix and a module attribute matrix of new design can be generated through the following operations.

A. Module merging processes in the module conjunction matrix:

1) Set $mc_{ik} = mc_{ik}$ XOR mc_{jk} $\forall k$=0,1,2, … , MN-1
2) Set $mc_{ki} = mc_{ki}$ XOR mc_{kj} $\forall k$=0,1,2, … , MN-1
3) Delete row j
4) Delete column j

B. Module merging rule in the module instance-requirement matrix:

1) $mi_{0i} = mi_{0i} \bullet mi_{0j}$ (The reason for this rule is because a future change does not follow a special requirement)
2) $mi_{ki} = \max(mi_{ki}, mi_{kj})$ $\forall k$=1,2,…, MN-1 (It is because that the maximum instances can perform requirement k)

C. Module merging rule in module attribute matrix:

1) The *Name* of new module will be the combined *name* of module i and module j.
2) If the *Main chain index* of module i and module j are both greater than 0, the *Main chain index* of the newly generated module will be either the index of module i or that of module j, whichever is less. If only one *Main chain index* is "0", the *Main chain index* of the newly generated module will be either the index of module i or that of the module j, whichever is greater. If the *Main chain index* of both module i and module j are "0", the *Main chain index* of the new module will be "0".
3) Only when the *Basic module values* of both modules i and j are equal to "0" will the *Basic module* value of the new module be "0". Otherwise, it will be "1".
4) The *Purchasing cost*, *Storage cost* and *Exchange cost* of the new module will be the total of the purchasing, storage and exchange costs of modules i and j, respectively.
5) The *Error on x axis* of the new module will be the *Error on x axis* of module j, as the interface that causes the *error on x axis* of module i will no longer exist after the merge. Similarly, the *Error on y axis* and *Error on z axis* of the new module will be the *Error on y axis* and the *Error on z axis* of module j as well.
6) The *Number of instance* of the new module will be the product of the elements in column i in the modified module instance-requirement matrix.

5.3 Example

To illustrate how to obtain a new design from the basic design and the three matrices of the new design, a simple example is presented as follows.

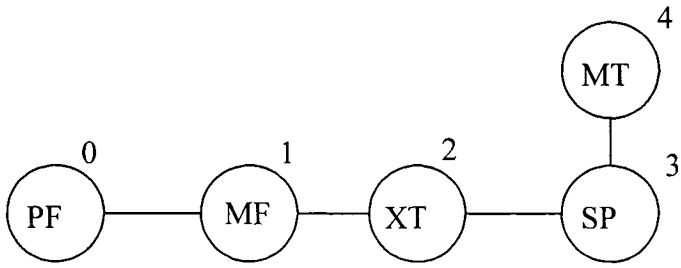


Figure 5-3 Topological structure of a basic design of an RMT

The interface vector of the basic design is:

$(I_{43}, I_{32}, I_{21}, I_{10}) = (1,1,1,1)$.

Figure 5-3 shows an example of an RMT basic design's topological structure. Its module conjunction matrix, module instance-requirement matrix and module attribute matrix are shown in Tables 5-4, 5-5 and 5-6, respectively.

Table 5-4 Module conjunction matrix of an RMT basic design

	Module 0	Module 1	Module 2	Module 3	Module 4
Module 0	0	1	0	0	0
Module 1	1	0	1	0	0
Module 2	0	1	0	1	0
Module 3	0	0	1	0	1
Module 4	0	0	0	1	0

Table 5-5 Module instance-requirement matrix of an RMT basic design

	Module 0	Module 1	Module 2	Module 3	Module 4
Future forecast	2	1	3	3	2
Require 1	1	1	2	1	2
Require 2	1	1	1	1	2
Require 3	2	1	2	1	2

Table 5-6 Module attribute matrix of an RMT basic design

	Module 0	Module 1	Module 2	Module 3	Module4
Index	0	1	2	3	4
Name	PF	MF	XT	SP	MT
Main chain index	1	3	2	4	0
Basic module	1	1	1	1	1
Purchasing cost($) *CP*	400	28000	2800	4000	2000
Storage cost($) *CE*	200	2500	1500	800	200
Exchange cost *CSt*	20	1500	150	150	150
Error on x axis ($\times 10^{-5}$rad) ε_x^{max}	1	4	1	5	0
Error on y axis ($\times 10^{-5}$rad) ε_y^{max}	1	1	1	1	0
Error on z axis ($\times 10^{-5}$rad) ε_z^{max}	5	1	5	1	0
Number of instance V or V'	4	1	12	3	16

According to equations (3-36), (3-37), (3-38) and (3-35), the costs of the above basic design are:

$CP = 400\times4+28000\times1+2800\times12+4000\times3+2000\times16 = \$107,200$

$CE = 200\times(4-1) +2500\times(1-1)+1500\times(12-1) +800\times(3-1)+200\times(16-1) = \$21,700$

$CSt = 20\times(4-1)+ 1500\times(1-1) +150\times(12-1) +150\times(3-1)+150\times(16-1) = \$4,260$

$CO = CP + CE + CSt = 107200+21700+4260 = \$133,160$

The configurations of the basic design are calculated as follows using equations (3-5), (3-6) and (3-4):

$V^{bas} = 4\times1\times12\times3\times16 = 2304$

$V^{aux} = 1$

$CF = V^{bas} \cdot V^{aux} = 2304\times1 = 2304$

Assuming that the maximum moving distances of the cutting tool in x, y and z directions are 400mm, 400mm and 250mm respectively, the maximum tool position errors caused by interfaces among modules along directions x, y and z are calculated based on equations (3-32), (3-33) and (3-34):

$$ERx = \max\left|(-\sum_{j=1}^{n_i}\varepsilon_{zj})t'_y + (\sum_{j=1}^{n_i}\varepsilon_{yj})t'_z\right| = 0.058\text{mm}$$

$$ERy = \max\left|(\sum_{j=1}^{n_i}\varepsilon_{zj})t'_x + (-\sum_{j=1}^{n_i}\varepsilon_{xj})t'_z\right| = 0.0755\text{mm}$$

$$ERz = \max\left|(-\sum_{j=1}^{n_i}\varepsilon_{yj})t'_x + (\sum_{j=1}^{n_i}\varepsilon_{xj})t'_y\right| = 0.0435\text{mm}$$

where,

$0 \le t'_x \le 400$ mm; $0 \le t'_y \le 400$ mm; $0 \le t'_z \le 250$ mm

$|\varepsilon_{x1}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{x2}| \le 4\times10^{-5}$ *rad*, $|\varepsilon_{x3}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{x4}| \le 5\times10^{-5}$ *rad*,

$|\varepsilon_{y1}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{y2}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{y3}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{y4}| \le 1\times10^{-5}$ *rad*,

$|\varepsilon_{z1}| \le 5\times10^{-5}$ *rad*, $|\varepsilon_{z2}| \le 1\times10^{-5}$ *rad*, $|\varepsilon_{z3}| \le 5\times10^{-5}$ *rad*, $|\varepsilon_{z4}| \le 1\times10^{-5}$ *rad*,

n_i=4

Suppose the interface vector of the new design is (I_{43}, I_{32}, I_{21}, I_{10}) = (1,1,0,1) (in other words, module 1 and module 2 in the basic design will be merged into a single module in the new design). The changes in the module conjunction, module instance-requirement and module attribute matrices of the basic design are demonstrated in Tables 5-7, 5-8 and 5-9.

Table 5-7 Changes in the module conjunction matrix of the basic design

	Module 0	Module 1	Module 2	Module 3	Module 4
Module 0	0	1	0	0	0
Module 1	1	0	1	1	0
Module 2	0	1	0	1	0
Module 3	0	1	1	0	1
Module 4	0	0	0	1	0

Table 5-8 Changes in the module instance-requirement matrix of the basic design

	Module 0	Module 1	Module 2	Module 3	Module 4
Future forecast	2	3	3	3	2
Require 1	1	2	2	1	2
Require 2	1	1	1	1	2
Require 3	2	2	2	1	2

Table 5-9 Changes in the module attribute matrix of the basic design

	Module 0	Module 1	Module 2	Module 3	Module 4
Index	0	1	2	3	4
Name	PF	XT,MF	XT	SP	MT
Main chain index	1	2	2	4	0
Basic module	1	1	1	1	1
Purchasing cost($) *CP*	400	30800	2800	4000	2000
Storage cost($) *CE*	200	4000	1500	800	200
Exchange cost($) *CSt*	20	1630	150	150	150
Error on x axis ($\times 10^{-5}$rad) ε_x^{max}	1	4	1	5	0
Error on y axis ($\times 10^{-5}$rad) ε_y^{max}	1	1	1	1	0
Error on z axis ($\times 10^{-5}$rad) ε_z^{max}	5	1	5	1	0
Number of instance *V* or *V'*	4	12	12	3	16

The grey areas in the above three matrixes are the row or columns that will be deleted after the merging of modules.

The topological structure of the new design after merging modules 1 and 2 is shown in Figure 5-4.

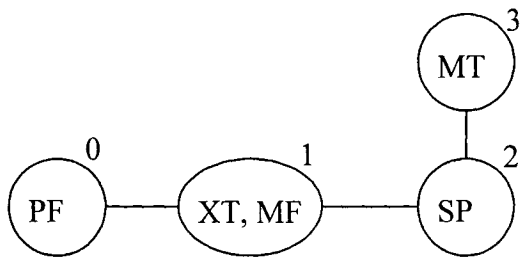


Figure 5-4 Topological structure of a new design of the example

The module conjunction, module instance-requirement and module attribute matrices for the new design after the merging process are shown in Tables 5-10, 5-11 and 5-12, respectively.

Table 5-10 Module conjunction matrix of the new design

	Module 0	Module 1	Module 2	Module 3
Module 0	0	1	0	0
Module 1	1	0	1	0
Module 3	0	1	0	1
Module 4	0	0	1	0

Table 5-11 Module instance-requirement matrix of the new design

	Module 0	Module 1	Module 2	Module 3
Future forecast	2	3	3	2
Require 1	1	2	1	2
Require 2	1	1	1	2
Require 3	2	2	1	2

Table 5-12 Module attribute matrix of the new design

	Module 0	Module 1	Module 2	Module 3
Index	0	1	2	3
Name	PF	XT, MF	SP	MT
Main chain index	1	2	4	0
Basic module	1	1	1	1
Purchasing cost($) *CP*	400	30800	4000	2000
Storage cost($) *CE*	200	4000	800	200
Exchange cost($) *CSt*	20	1630	150	150
Error on x axis ($\times 10^{-5}$rad) ε_x^{max}	1	4	5	0
Error on y axis ($\times 10^{-5}$rad) ε_y^{max}	1	1	1	0
Error on z axis($\times 10^{-5}$rad) ε_z^{max}	5	1	1	0
Number of instance *V* or *V'*	4	12	3	16

According to equations (3-5), (3-6) and (3-4), the configuration of the new design is computed as:

$$V^{bas} = 4 \times 12 \times 3 \times 16 = 2304$$

$$V^{aux} = 1$$

$$CF = V^{bas} \cdot V^{aux} = 2304 \times 1 = 2304$$

Using equations (3-36), (3-37), (3-38) and (3-35), the costs of new design are calculated as:

$CP = 400\times4+30800\times12+4000\times3+2000\times16 = \$415{,}200$

$CE = 200\times(4-1)+4000\times(12-1)+800\times(3-1)+200\times(16-1) = \$49{,}800$

$CSt = 20\times(4-1)+1630\times(12-1)+150\times(3-1)+150\times(16-1) = \$20540$

$CO = CP+CE+CSt = 415200+49800+20540 = \$485540$

Now, the CO is now \$485,540, which is much higher than \$133,160 obtained before the merge. However, it should be noted that this is just an example used to illustrate the calculation process. There is no guaranty that the new design after merge will necessarily be better than the previous design and hence it does not mean the new design will be selected. In fact, if the new design after merge is worse that the original one it will be eliminated in the following optimization step. The new design's maximum tool position errors caused by interfaces among modules on x, y and z directions are obtained through equations (3-32), (3-33) and (3-34):

$$ERx = \max\left|(-\sum_{j=1}^{n_i}\varepsilon_{zj})t'_y + (\sum_{j=1}^{n_i}\varepsilon_{yj})t'_z\right| = 0.0355\text{mm}$$

$$ERy = \max\left|(\sum_{j=1}^{n_i}\varepsilon_{zj})t'_x + (-\sum_{j=1}^{n_i}\varepsilon_{xj})t'_z\right| = 0.053\text{mm}$$

$$ERz = \max\left|(-\sum_{j=1}^{n_i}\varepsilon_{yj})t'_x + (\sum_{j=1}^{n_i}\varepsilon_{xj})t'_y\right| = 0.052\text{mm}$$

where,

$0 \le t'_x \le 400\text{mm};\ 0 \le t'_y \le 400\text{mm};\ 0 \le t'_z \le 250\text{mm}$

$|\varepsilon_{x1}| \le 1\times10^{-5}\ rad,\ |\varepsilon_{x2}| \le 4\times10^{-5}\ rad,\ |\varepsilon_{x3}| \le 5\times10^{-5}\ rad$

$|\varepsilon_{y1}| \le 1\times10^{-5}\ rad,\ |\varepsilon_{y2}| \le 1\times10^{-5}\ rad,\ |\varepsilon_{y3}| \le 1\times10^{-5}\ rad$

$|\varepsilon_{z1}| \le 5\times10^{-5}\ rad,\ |\varepsilon_{z2}| \le 1\times10^{-5}\ rad,\ |\varepsilon_{z3}| \le 1\times10^{-5}\ rad$

$n_i=3$

Each alternative design is obtained by one or several merging processes according to the basic design. The merging only occurs between adjacent modules. The number of the alternative design can be calculated using equation (5-1):

$$N^{al} = C_1^{MN-1} + C_2^{MN-1} + C_3^{MN-1} + \cdots + C_{MN-1}^{MN-1} = 2^{MN-1} \tag{5-1}$$

Chapter 6. Algorithm for Design Selection

In this Chapter, the complexity of RMT design optimization using MFCP is discussed and a Particle Swarm Optimization Algorithm (PSOA) integrated with MFCP is proposed.

6.1 Complexity analysis of the RMT design problem

According to equation (5-1), the number of RMT alternative designs is 2^{MN-1}. In this expression, *MN* is the number of modules in a basic design. The increase in *MN* will lead to substantial increase in the number of variables x_i in equation (3-33), determines the size of the problem in this study. No suitable commercial software is currently available to handle this problem when *MN* becomes sufficiently large. Therefore, an efficient algorithm is required to obtain an optimal or near optimal solution. The complexity of the problem should be analyzed, since they form the basis of algorithm selection.

6.1.1 Complexity of the problem

The complexity of a problem is affected by the number of steps that have to be undertaken to solve an instance in the problem using the most efficient algorithm. It can be expressed as a function of the size of the input. Usually, the input can be represented as a vector *X*, and size of the input can be represented as $|X|$

A naïve algorithm of the RMT design problem is used to examine all possible alternative designs based on the rules and models discussed in Chapters 3 and 4. The input to an overall algorithm is the *MN* modules of a basic design, whereas the output is the best design, considering the three objectives (configurations, cost and accuracy).

Accordingly, the algorithm of the problem in this study can be shown in the following flow chart (Figure 6-1).

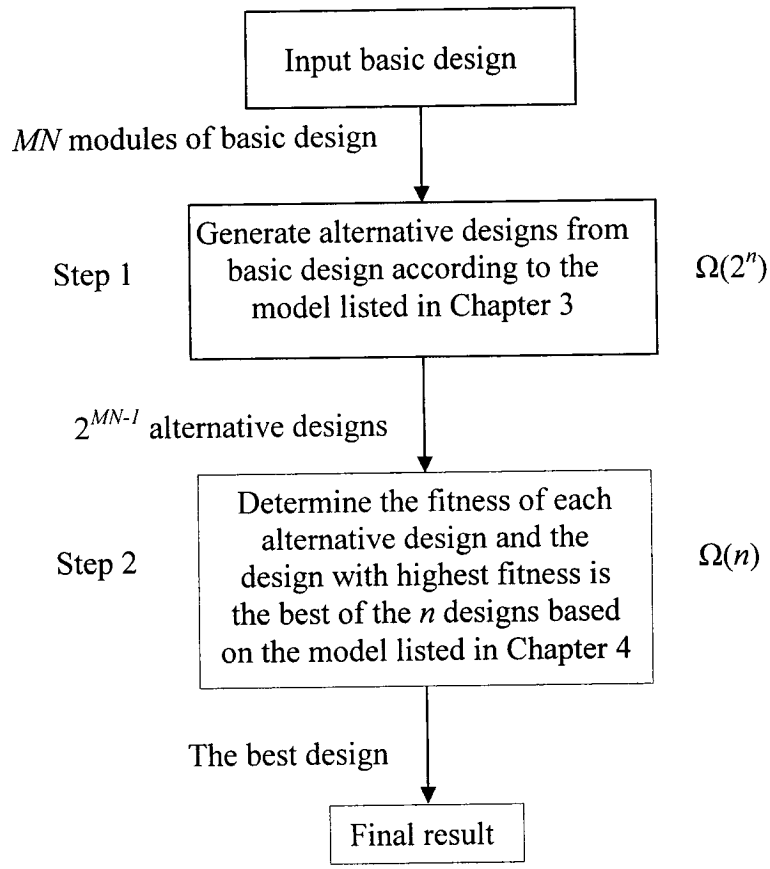


Figure 6-1 Flow chart of the algorithm

As indicated in the flow chart above, the algorithm consists of two steps. In the first step, the input is still the basic design with *MN* modules. The length of input, thus, is *MN*. In this step, the algorithm generates all possible alternative designs by merging modules in a basic design following the rules discussed in Chapter 4. As mentioned in Chapter 3, the number of combinations of modules (including basic design) can be calculated as:

$$N^{al} = \binom{MN-1}{0} + \binom{MN-1}{1} + \binom{MN-1}{2} + \ldots + \binom{MN-1}{MN-1} = 2^{MN-1} \quad (6\text{-}1)$$

As shown in equation (6-1), the algorithm automatically generates 2^{MN-1} alternative designs by merging different modules into new modules and performs 2^{MN-1} times of computations to find the number of configurations, costs and accuracy based on the respective module attribute matrices, module instance-requirement matrices and module conjunction matrices.

Moreover, in the same step, the process of merging modules, generating new matrices and calculating the number of configurations, cost and the accuracy is repeated 2^{MN-1} times by following the same rules. As a result, the complexity of the first step is $O(2^n)$, where $n = MN-1$.

In the second step, the inputs are the 2^{MN-1} alternative designs generated from the first step. Then, search the optimal from the 2^{MN-1} alternative designs by using the PSOA algorithm. Using the MFCP method, the algorithm obtains the best design among the 2^{MN-1} designs by comparing the fitness δ of each alternative design. Thus, the problem can be converted into the MFCP method by combining (4-30) and (4-31):

max δ_1

subject to

$$\left.\begin{aligned}
\delta_1 &\le \frac{\sum_{m=0}^{MN-1}\sum_{n=0}^{\binom{MN-1}{m}} CF_{mn}\cdot x_{mn} - L_{CONF}}{U_{CONF}-L_{CONF}} = \frac{\sum_{i=1}^{2^{MN-1}} CF_i\cdot x_i - L_{CONF}}{U_{CONF}-L_{CONF}} \\
\delta_2 &\le \frac{\sum_{m=0}^{MN-1}\sum_{n=0}^{\binom{MN-1}{m}} CO_{mn}\cdot x_{mn} - L_{COST}}{U_{COST}-L_{COST}} = \frac{\sum_{i=1}^{2^{MN-1}} CO_i\cdot x_i - L_{COST}}{U_{COST}-L_{COST}} \\
\delta_3 &\le \frac{\sum_{m=0}^{MN-1}\sum_{n=0}^{\binom{MN-1}{m}} ERx_{mn}\cdot x_{mn} - L_{ERRX}}{U_{ERRX}-L_{ERRX}} = \frac{\sum_{i=1}^{2^{MN-1}} ERx_i\cdot x_i - L_{ERRX}}{U_{ERRX}-L_{ERRX}} \\
\delta_3 &\le \frac{\sum_{m=0}^{MN-1}\sum_{n=0}^{\binom{MN-1}{m}} ERy_{mn}\cdot x_{mn} - L_{ERRY}}{U_{ERRY}-L_{ERRY}} = \frac{\sum_{i=1}^{2^{MN-1}} ERy_i\cdot x_i - L_{ERRY}}{U_{ERRY}-L_{ERRY}} \\
\delta_3 &\le \frac{\sum_{m=0}^{MN-1}\sum_{n=0}^{\binom{MN-1}{m}} ERz_{mn}\cdot x_{mn} - L_{ERRZ}}{U_{ERRZ}-L_{ERRZ}} = \frac{\sum_{i=1}^{2^{MN-1}} ERz_i\cdot x_i - L_{ERRZ}}{U_{ERRZ}-L_{ERRZ}}
\end{aligned}\right\} \quad (6\text{-}2)$$

Subject to:

$$\sum_{i=1}^{2^{MN-1}} x_i = 1 \qquad x_i = \begin{cases} 1 & \text{the } i\text{th candidate be selected} \\ 0 & \text{the } i\text{th candidate not be selected} \end{cases}$$

It can be seen in equation (6-2) that there are five "less than" constraints for the three δ's. According to equation (4-3), there are five upper-limits of the δ_1. Therefore, the algorithm obtains the fitness δ_1 of each alternative design by doing four comparison operations. Then, in order to choose the best design (with highest $\delta_1$value), the algorithm has to do $4 \cdot 2^{MN-1}$ comparison operations. Therefore the complexity of the second step is also $O(2^n)$ with $n = MN$. It should be noted that this is the complexity when all the δ_1's are available. However, obtaining each of these δ_1's requires solving an MFCP problem as defined by equation (6-2). This would, of course, further complicate the problem substantially.

6.1.2 Classification of the problem's complexity

To illustrate the complexity class of the RMT design problem, the Knapsack problem will be used as a reference. The Knapsack problem is a well-known NP-hard problem. The 0-1 Knapsack problem can be written as:

$$\begin{aligned} &\max \sum_{i=1}^{n} P_i x_i \\ &\text{subject to:} \\ &\sum_{i=1}^{n} w_i x_i \le c \\ &x_i = 0 \text{ or } 1, i = 1,2,...,n \end{aligned} \tag{6-3}$$

To solve this problem, "a naïve approach would be to program a computer to examine all possible binary vectors x then select the best of those which satisfy the constraint. The number of such vectors is 2^n" (Martello and Toth 1990). It can be readily concluded that the complexity of the 0-1 Knapsack problem is $O(2^n)$.

As indicated in Chapter 6.1.2, the direct approach to solving the problem is to examine all alternative designs according to all possible interface vectors, whose number is 2^n, where n is equal to MN-1, and the complexity of the approach is also $O(2^n)$. As a result, the problem has the same degree of difficulty as the 0-1 Knapsack problem, in addition to solving the MFCP problems.

6.2 Particle swarm optimization algorithm (PSOA)

In this problem, the best design is chosen from 2^{n-1} possible designs (including the basic design and all derived alternative designs), where n is the number of modules in a basic design. If n is greater than 30, a more efficient algorithm is required to conduct the calculation. However, the algorithms used to find the optimal solution for the Knapsack problem, are very time-consuming. The algorithm for the problem in this study, which is similar, has the same weaknesses as those used for the Knapsack problem. Therefore, the PSOA method, which is an algorithm to seek a near optimal solution, is applied to the problem to obtain the near optimal design.

6.2.1 Introduction of the PSOA

The PSOA method was first proposed by Kennedy and Eberhart (1995) and is based on the simulation of a simplified social model. PSOA simulates the behaviours of bird flocking. A group of birds are randomly searching for food in an area. However, there is only one piece of food available. None of the birds knows where the food is. The most effective bird is the one who follows the bird that is standing nearest to the food. In the PSOA, the "bird" is called a "particle". Each solution is a "bird" in the searching area. All of particles have fitness values, which are evaluated by the fitness function and are to be optimized, and velocities, which direct the flying of the particles. The particles fly through the problem (searching) space by following the current optimum particles.

The PSOA has been applied to analytical test function (Kennedy and Spears 1998). Recently, it was successfully applied to the optimal shape and size design of structures (Fourie and Groenwold 2002). This new application signifies that PSOA has started to be adopted in the engineering field.

A PSOA is initialized with a group of random particles (solutions) and then searches for the optimal solution by updating generations. In each iteration, every particle is updated by the two "best" values. The first one, which is called *pbest*, is the best solution (fitness) that the particle has achieved so far. The fitness value is also stored. The second one, which is tracked by the particle swarm optimizer, is the best value obtained so far by any particle in the population. This best value is a global best value and is called *gbest*.

For particle d, Kennedy and Eberhart (1995) originally proposed that position x^d is updated as:

$$x_{k+1}^d = x_k^d + v_{k+1}^d \tag{6-4}$$

The velocity v^d is updated as:

$$v_{k+1}^d = v_k^d + c_1 r_1 (p_k^d - x_k^d) + c_2 r_2 (p_k^g - x_k^d) \tag{6-5}$$

where the subscript k indicates a pseudo-time increment. While p_k^d represents the best previous position of particle d at time k, p_k^g represents the global best position in the swarm at time k. r_1 and r_2 symbolize uniform random numbers between 0 and 1. c_1 and c_2 are learning factors and are selected in range (0, 4). Kennedy and Eberhart suggest c_1=c_2=2. Figure 6-2 shows a PSOA flowchart.

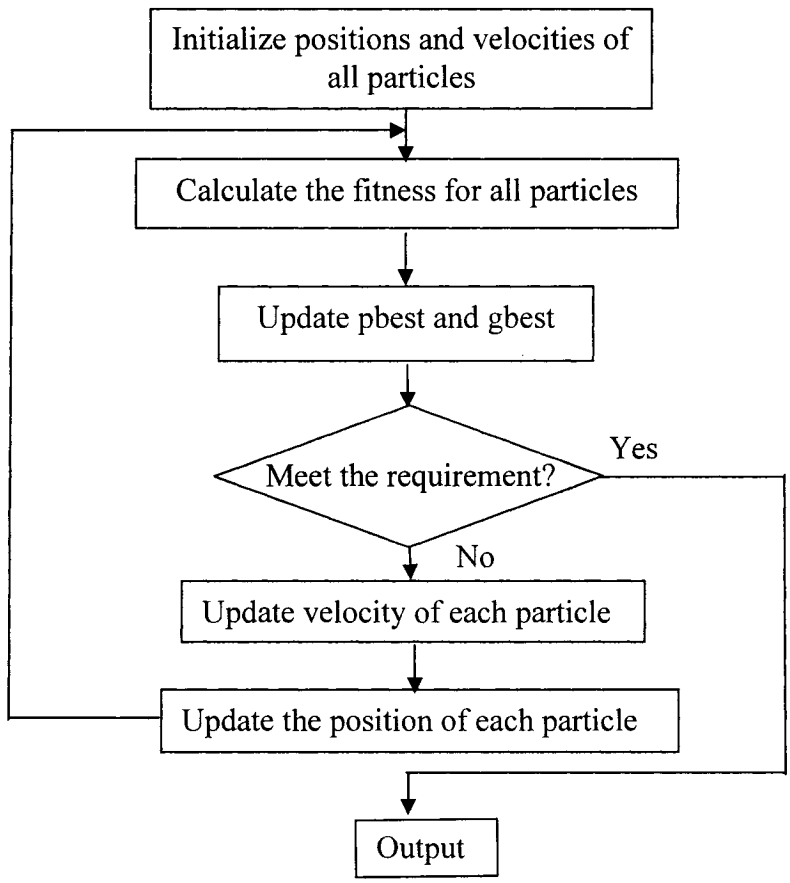


Figure 6-2 A PSOA flow chart

Shi and Eberhart (1998) introduced an inertia term ω. They suggested that ω be within a range of (0.8, 1.4). Accordingly, equation (6-5) should be modified to:

$$v_{k+1}^d = \omega v_k^d + c_1 r_1 (p_k^d - x_k^d) + c_2 r_2 (p_k^g - x_k^d) \quad (6\text{-}6)$$

Fourie and Groenwold (2002) later used the best ever position p_g to replace p_k^g, to improve the rate of convergence of PSOA. Following this, equation (6-6) becomes:

$$v_{k+1}^d = \omega v_k^d + c_1 r_1 (p_k^d - x_k^d) + c_2 r_2 (p_g - x_k^d) \quad (6\text{-}7)$$

Recently, a hybrid PSOA with passive congregation (PSOPC) was introduced (He *et al* 2004). As a result, equation (6-7) was changed to:

$$v_{k+1}^d = \omega v_k^d + c_1 r_1 (p_k^d - x_k^d) + c_2 r_2 (p_g - x_k^d) + c_3 r_3 (R_k^d - x_k^d) \quad (6\text{-}8)$$

where R_i is a particle randomly selected from the swarm, c_3 is the passive congregation coefficient, and r_3 is a uniform random sequence in the range of (0, 1). He *et al* (2004) suggested that $c_1 = c_2 = c_3 = 0.5$. If c_3 is equal to zero, equation (6-8) reduces to equation (6-7).

The above description indicates that PSOA shares many features with evolutionary computation techniques such as Genetic Algorithms (GA). Both the GA and PSOA are initialized with a population of random solutions and searches for the optimal solution by updating generations. However, unlike GA, the PSOA has no evolution operators such as crossover and mutation. In a PSOA, the potential solutions, which are called particles, fly through the problem space by following the current optimum particles. The information sharing mechanism in the PSOA is also significantly different from GA. In the PSOA, only *gbest* gives out the information to others. It is a one-way information sharing mechanism. Particles update themselves according to the internal velocity. The evolution only looks for the best solution. In most cases, all the particles tend to converge to the best solution quickly even in the local version.

Compared to GA, PSOA only has two core updating equations, i.e., position and velocity equations, and fewer parameters to adjust. Additionally, due to its "one-way" information sharing mechanism and simple updating operation, PSOA requires less memory space than GA. Thus, PSOA is relatively simple in both formulation and computer implementation. Furthermore, PSOA can easily be parallelized in massive parallel processing machines (Schutte *et al* 2004). Finally, PSOA has been successfully

applied in many areas, such as function optimization, artificial neural network training, fuzzy system control, and others where GA can be applied.

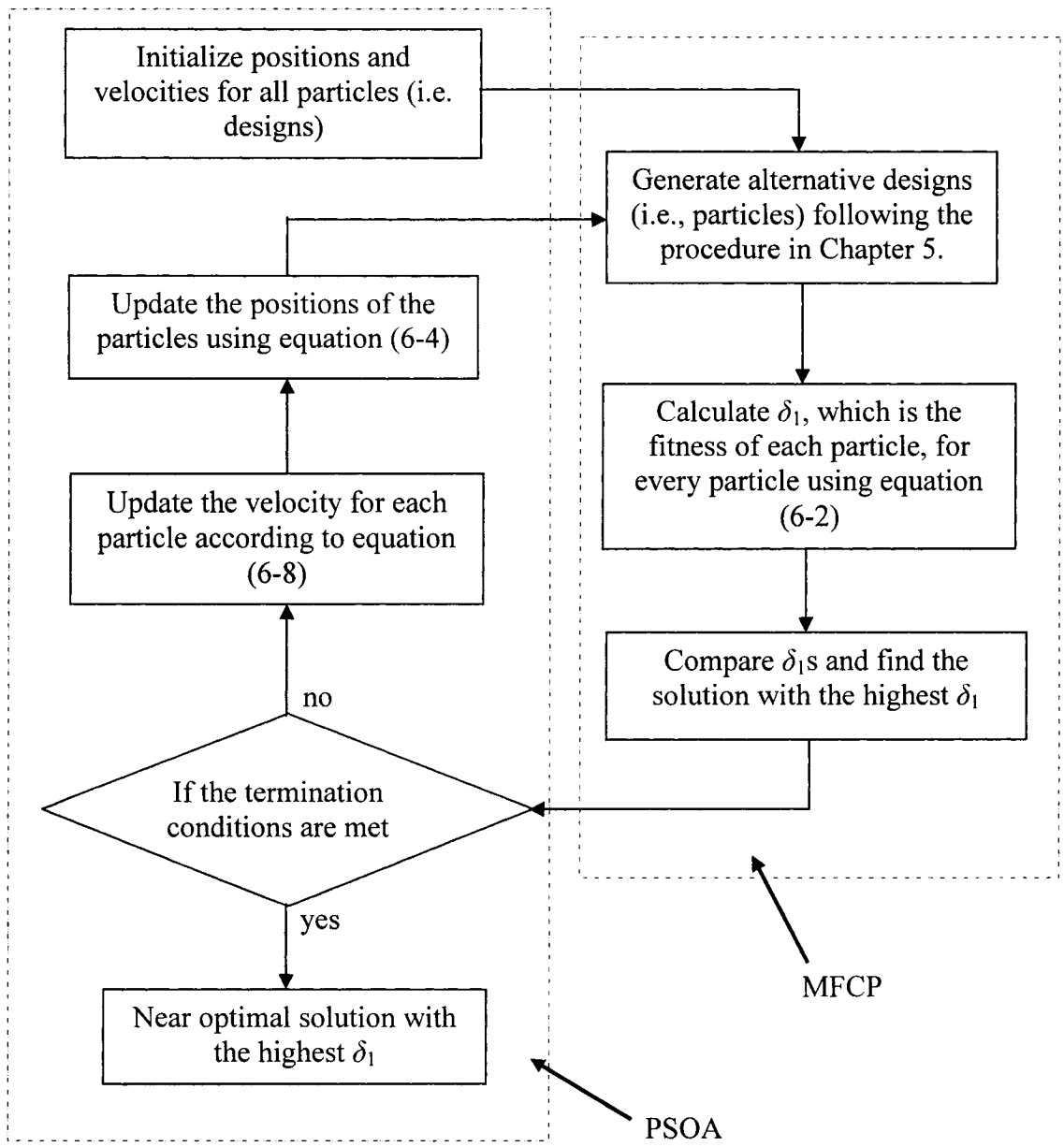


Figure 6-3 Flow chart of the procedure for the problem combined MFCP and PSOA

6.2.2 Application of PSOA to RMT design

In this study, PSOA is combined with MFCP and applied to solving the problem discussed. The interface vector is used to describe the position of a particle in PSOA,

since it is identical for each alternative design and can be converted into a decimal number correspondingly within the range ($0 \sim 2^{MN-1}$). Moreover, due to the fact that the value of δ_1 in equation (6-2) is used to evaluate the alternative design, the value of δ_1 becomes the fitness for each particle of PSOA in this study. Accordingly, the flow chart of the combined PSOA and MFCP is shown in Figure 6-3, and the associated pseudo code of the procedure is written as follows:

Set k := 0;

Randomly initialize positions and velocities of all particles;

WHILE *(the termination conditions are not met)*

FOR *(each particle d in the swarm)*

Calculate fitness*: **If** cost(X^d)>affordable cost **then** set fitness =-10*

else *Calculate the fitness value of current particle: $f(X^d)$*

Update pbest*: Compare the fitness value of pbest with $f(X^d)$.*

If *$f(X^d)$ is better than the fitness value of pbest,*

then *set pbest to the current position X^d;*

Update gbest*: Find the global best position of the swarm.*

If *$f(X^d)$ is better than the fitness value of gbest,*

then *gbest is set to the position of the current particle X^d;*

Update *R^d: Randomly select a particle from the swarm as R^d;*

Update velocities*: Calculate velocities V^d using Eq. (6-8).*

If *$V^d > Vmax$* ***then*** *$V^d = Vmax$.*

If *$V^d < Vmin$* ***then*** *$V^d = Vmin$;*

Update positions*: Calculate positions X^d using Eq. (6-4);*

END FOR

Set k := k + 1;

END WHILE

where,

fitness $f(X^d)$ is the value of δ_1 in accordance with equation (6-2);

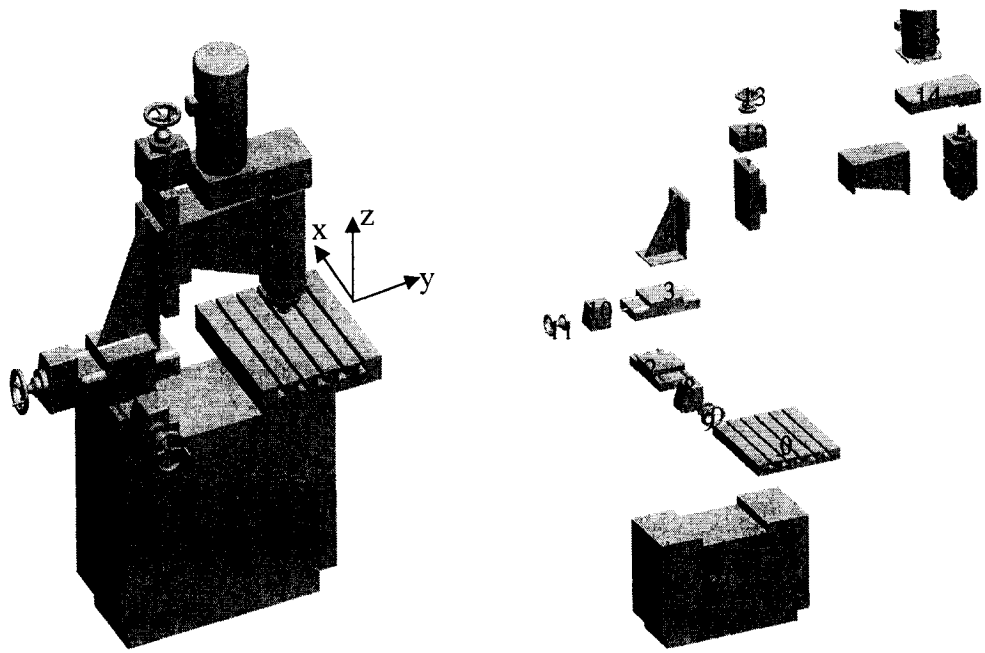
position X^d is the decimal numerical expression of the design's interface vector and is in the range of $[0, 2^{MN-1}]$.

Chapter 7. Case Study

In order to illustrate the methods proposed in previous chapters, the optimization of an RMT design of a boring machine is studied in this chapter.

7.1 The boring machine

The basic design of the boring machine is composed of 16 modules. As an RMT, the number of configurations of the boring machine is more important than its cost and accuracy. Thus, the priority order of the design objectives is: configurations > cost > accuracy.



a) The boring machine b) Modules of the basic design of the boring machine

Figure 7-1 Schematic view of the basic design of a boring machine

Figure 7-1 shows the basic design of a boring machine tool. The topological structure of the boring machine's basic design is shown in Figure 7-2. Attributes of each module in the basic design are shown in Tables A1 and A3 (see Appendix A)

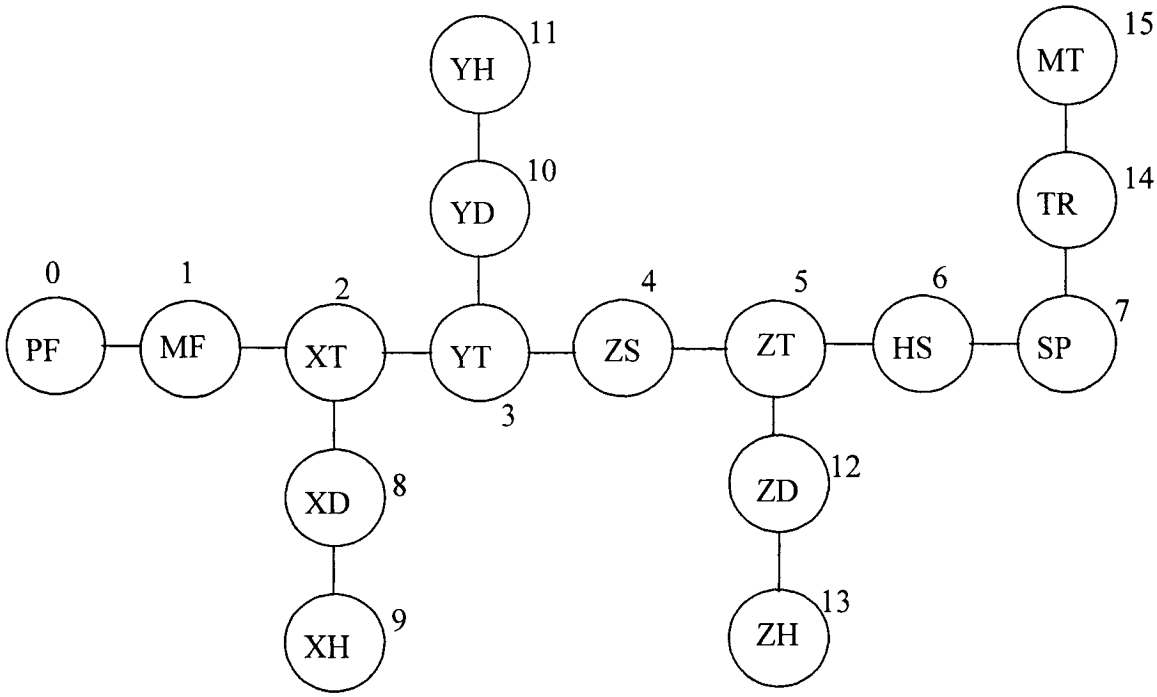


Figure 7-2 Topological structure of the basic design of a boring machine

7.2 Data explanation

In the boring machine design, the spindle and X, Y, Z slide tables are selected from the SETCO Sales Company's product list. The dimensions of these modules also come from that product list. The spindle is a boring spindle of type 4307B (SETCO Sales Company 2004a) and the slide tables are the precision dovetail slide of type M4 10" (SETCO Sales Company 2004b).

Also, in the design, the data concerning purchasing and exchange costs for slide tables, slide drivers and spindle are from SERVO Products Company's product quotation. The costs of the other modules of the boring machine can be reasonably estimated based on the price of a milling/boring machine (without control system) (SERVO Products Company 2003).

Moreover, in the same design, the errors on the x, y and z axes are calculated using the method discussed in Chapter 3.2. Data related to the calculation of those errors for the spindle and slide tables, namely, the dimensions and parallelism of the modules, are provided by SETCO. The data associated with the calculation of those errors for the other modules are from surface machining tolerance standards of machine tools (Cheng, *et al*

1993) in class IV. The surfaces defined in class IV of the standard are either more important than other surfaces or treated as referential surfaces of the machine tools. Dimensions of the referential surfaces and connections of each module are shown in Table A1 in Appendix A.

Given that all modules can be used on other machine tools or sold as second hand products, the affordable cost of an RMT can be estimated at $500,000.

The module conjunction, module instance-requirement and module attribute matrices of the basic design of the boring machine are shown in Tables A2, A3 and A4, respectively, in Appendix A.

After optimization, the module conjunction matrix, module instance-requirement matrix, module attribute matrix and module warehouse of the optimized design are presented in Tables A5, A6, A7 and A8, respectively, of Appendix A.

The modules and topological structure of the optimized design are shown respectively in Figures 7-3 and 7-4.

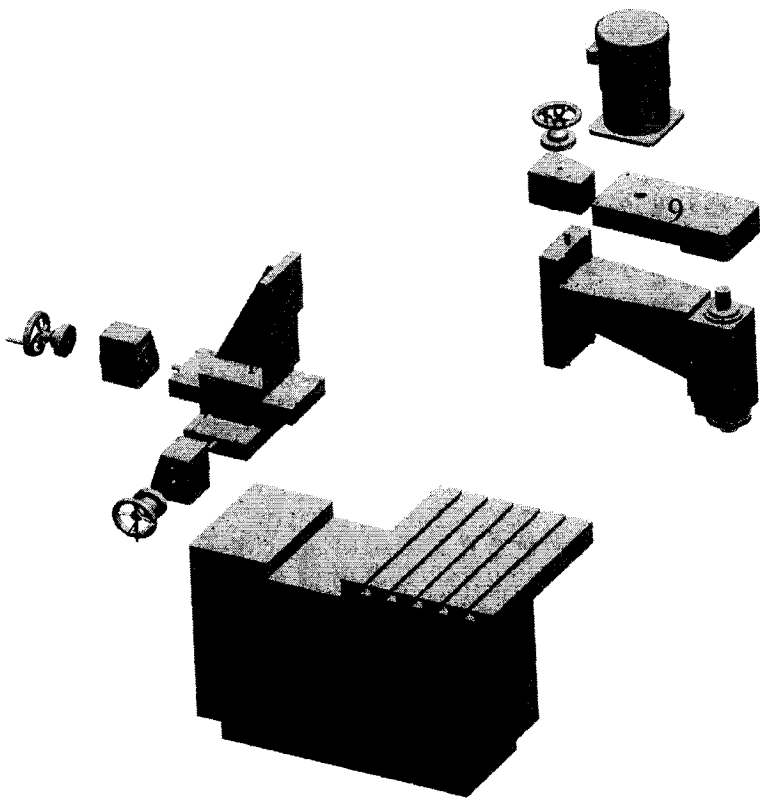


Figure 7-3 Modules of the optimized design of the boring machine

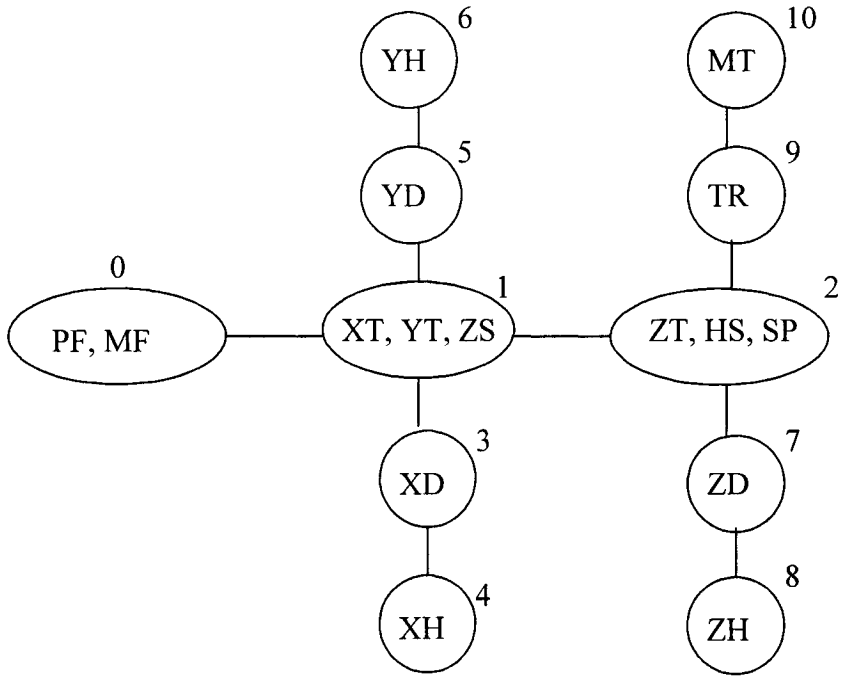


Figure 7-4 Topological structure of the optimized design of the boring machine

The values associated with the three objectives of the design, after optimization, are listed below:

Number of configurations of the optimized design: 256,000

Cost of the optimized design: $141,690

Error on x axis of the optimized design: 0.06785mm

Error on y axis of the optimized design: 0.07875mm

Error on z axis of the optimized design: 0.07995mm

The errors on the x, y and z axes listed above are the maximum values at a maximum range of displacement under the worst installation conditions. Under normal conditions, the errors would be much smaller. In addition, the errors can be significantly reduced by adjusting the interface among modules 0, 1 and 2 in the optimized design or by using compensation technology.

7.3 Selection of parameters of PSOA

The computation process in this case study demonstrates that the selection of PSOA parameters can dramatically affect the convergence rate and the time that PSOA spends on locating the optimal solution. Thus, several sets of parameters are tested in PSOA in order to obtain the best one.

Table 7-1 illustrates the average, maximum and minimum number of iterations that are used to obtain the final result, as well as the times that PSOA was only able to obtain a local optimum (not the best result), by using 15 sets of parameters. These data are derived from tests repeated 100 times for each set of parameters.

Table 7-1 Efficiencies of PSOA with different parameters

Parameter set number	Parameters (ω, c_1, c_2, c_3)	Average number of iterations	Maximum number of iterations	Minimum number of iterations	Number of runs with only local optima
1	(1, 1, 1, 0)	304	1394	7	0
2	(1, 2, 2, 0)	400	2611	0	51
3	(1, 0.5, 0.5, 0.5)	280	1685	6	0
4	(0.8, 1, 1, 0)	151	2969	0	58
5	(1.2, 1, 1, 0)	478	2873	11	0
6	(0.8, 2, 2, 0)	450	2861	2	1
7	(1.2, 2, 2, 0)	492	2137	4	0
8	(0.8, 0.5, 0.5, 0.5)	58	368	0	70
9	(1.2, 0.5, 0.5, 0.5)	339	1768	0	29
10	(0.6, 0.5, 0.5, 0.5)	26	209	0	76
11	1, 0.5, 0.5, 0	294	1620	7	0
12	1, 1, 1, 0.5	310	1695	14	0
13	1.4, 2, 2, 0	585	2845	0	7
14	1, 2, 2, 0.5	333	1625	5	0
15	1, 2, 2, 1	426	1838	8	0

Common condition: particle size=100, maximum allowable number of iterations=3000, number of runs=100

In Table 7-1, the values in "Average iterations" column show the efficiencies of PSOA by using the different parameters. Lower values indicate that the results were obtained relatively quickly. Hence, the lower the value, the higher the efficiency of PSOA. Also, this value reflects the convergence rate. The higher the value, the lower the convergence rate.

In addition, it can be seen from Table 7-1 that the maximum numbers of iterations corresponding to the parameters in sets 8 and 10 are very low. These small numbers of

iterations indicate that the PSOA converged so fast that it often ended at a local optimum. In those situations, the particles fly around the local optimum indefinitely and cannot escape from it.

In contrast, the parameters in sets 2, 4 and 9, whose maximum iterations are relatively large, caused PSOA to converge so slowly that it could not locate an optimum on time. In such cases, the particles always "over fly" the best position, and thus, cannot find the optimal. Consequently, the parameters in sets 2, 4, 8, 9 and 10 will be discarded.

The parameters in set 3 are from a hybrid PSOA with passive congregation (PSOPC) equation (equation (5-4)) suggested by He *et al* (2004). As the performance of PSOA using the parameters in set 3 is the best among the remaining sets of parameters, the data of ω=1, c_1= c_2= c_3=0.5 is therefore selected in order for the PSOA method to find an optimal design.

7.4 Comparison of MFCP and FCP results

The results of using the Fuzzy-Chebyshev Programming (FCP) method, which treats the three objectives (configurations, cost and accuracy) as equally important, are listed below:

Numbers of Configuration of the optimized design: 204,800

Cost of the optimized design: $158,091

Error on x axis of the optimized design: 0.06587mm

Error on y axis of the optimized design: 0.07530mm

Error on z axis of the optimized design: 0.07452mm

Table 7-2 Comparison of the results using MFCP versus FCP

	MFCP	FCP	Change
Configuration of the optimized design	256,000	204,800	25%
Cost of the optimized design ($)	141,690	158,091	-10.37%
Error on x axis of the optimized design (mm)	0.06785	0.06587	3.01%
Error on y axis of the optimized design (mm)	0.07875	0.07530	4.58%
Error on z axis of the optimized design (mm)	0.07995	0.07452	7.28%

While FCP considers all sub-objectives as equally important, MFCP emphasizes the relative importance among sub-objectives. Table 7-2 demonstrates that, compared to the optimized RMT generated from FCP, the RMT generated from MFCP leads to much

higher configurability and lower cost, at the expense of slightly lower accuracy. In addition, the MFCP result reflects the same order of relative importance among the three objectives as given in Chapter 3. Thus, MFCP can accurately perform RMT optimization under a given order of relative importance among objectives.

Chapter 8. Summary and Directions for Future Studies

The research objectives proposed in Chapter 1 have been achieved and the work carried out in this study is summarised below. Recommendations for future work are also presented in this chapter.

8.1 Summary

This study has provided a methodology to optimize RMT design and its module warehouse based on the basic design of an RMT. Unlike most related studies, this methodology considers three important criteria in RMT design according to a given priority order. The three criteria are configurability, cost and accuracy. In summary, the following has been accomplished:

1) The RMT design problem has been formulated as a multi-objective problem incorporating the three design criteria mentioned above. Due to the nature of the RMT design, the order of priorities of the three criteria is pre-specified and has to be respected in solving the multi-objective model.
2) A modified fuzzy-Chebyshev programming (MFCP) method has been proposed in this study to obtain the optimal results. The final results in the above optimization case have demonstrated that the MFCP approach cannot only solve the problem, but also accurately reflect the given order of relative importance of the design objectives.
3) The particle swarm optimization algorithm (PSOA) method has been applied in this study to obtain quick and near optimal solutions to large size RMT design problems. The PSOA method due to its simplicity in parameter selection and implementation is particularly suitable for the RMT design problem since the number of RMT candidate designs rises dramatically with the increase in the number of modules in an RMT basic design.
4) A case study has been conducted using a boring machine design example. The design of the reconfigurable boring machine was optimized using the methodology proposed in this study. Comparison between the MFCP and FCP results is also made which clearly illustrated that the MFCP not only respects the pre-specified order of

importances of the objectives but also leads to much higher configurability and lower cost at the expense of slight worsening of process accuracy.

8.2 Future work

To further improve RMT design, the above study could be extended in the following directions:

1) *Involvement of all RMTs in an RMS*

This study is focused on a single RMT. Extending the study to an entire reconfigurable manufacturing system (RMS) would further improve profitability. One of the reasons for this is that the modules in one RMT can be used in other RMTs in an RMS within the same product family. Incorporation of such interactions among RMTs would be of particular interest to manufacturers.

2) *Extension and enhancement of the MFCP method*

This study has only applied the MFCP method to solving a multi-objective problem with a priority order at the same level. However, MFCP can be further developed to be applied to multi-level multi-objective problems with a pre-specified order of importances of sub-objectives within every level.

3) *Incorporation of control system into RMT design optimization*

This study has mostly concentrated on the aspects of RMT hardware. Many previous studies have been conducted on machine tools and RMS control systems independently. Few studies have integrated the control system into the optimization of RMT design, although the control system is a critical part of an RMT. Therefore, incorporating the control system into the optimization of an RMT design would be an interesting topic for future researchers.

References

Abdi, M.R. and Labib, A.W., 2003, "A design strategy for reconfigurable manufacturing systems (RMS) using analytical hierarchical process (AHP): a case study," *Journal of Intelligent Manufacturing*, **41**(10), pp. 2273-2299.

Abdi, M.R. and Labib, A.W., 2004, "Feasibility study of the tactical design justification for reconfigurable manufacturing systems using the fuzzy analytical hierarchical process," *International Journal of Production Research*, **42**(15), pp. 3055–3076.

Asl, F.M., Ulsoy, A.G. and Koren, Y., 2000, "Dynamic modeling and stability of the reconfiguration of manufacturing systems," *Proc. of the Japan-USA Symposium on Flexible Automation*, Ann Arbor, MI. USA, pp.863-870.

Brucolleri, M., Amico, M. and Perrone, G. 2003, "Distributed intelligent control of exceptions in reconfigurable manufacturing systems," *International Journal of Production Reaserch*, **41**(7), pp. 1393-1412.

Cecil, J., 2001, "Computer-aided fixture design – a review and future trends," *International Journal of Advance Manufacturing Technology*, **18**(11), pp. 790-793.

Chen, B.-C., Tilbury, D.M. and Ulsoy, A.G., 1998, "Modular control for machine tools: cross-coupling control with friction compensation," *Proceedings of the ASME-IMECE Dynamic Systems and Control Division*, Anaheim, California, **64**, pp. 455-462.

Cheng, D., Wang, D., Jiang, Y., Li, C. and Han, X., 1993, "Mechanical design handbook," 3rd Edition, Chemical Engineering Press, Beijing. P.R. China.

Cole, K., 2004, "Reconfigure it out – reconfigurable manufacturing system gain ground in the automotive sector," *Cutting Tool Engineering*, **56**(9), pp. 40-45.

DeGaspari, J., 2002, "All in the family - flexible machining system give manufacturers a hedge on their bets," *Mechanical Engineering Magazine*, February, pp. 56-58.

Diplaris, S.C. and Sfantsikopoulos, M. M., 2000, "Cost-tolerance function: a new approach for cost optimum machining accuracy," *International Journal of advanced Manufacturing Technology*, **16**(1), pp. 32-38.

El-Wahed, W. and Abo-Sinna, M., 2001, "A hybrid fuzzy-goal programming approach to multiple objective decision making problem," *Fuzzy Set and Systems*, **119**(1), pp. 71-85.

Fourie, P.C. and Groenwold, A.A., 2002, "The particle swarm optimization algorithm in size and shape optimization", *Structure Multidisciplinary Optimization*, **23**(4), pp. 259-267.

He, S., Wu, Q.H., Wen, J.Y., Saunders, J.R. and Paton, R.C., 2004, "A particle swarm optimizer with passive congregation," *BioSystems,* **78**(3), pp. 135-147.

Heisel, U. and Meitzner, M., 2003, "Progress in reconfigurable manufacturing systems," *2nd International Conference on Reconfigurable Manufacturing, August 20, Ann Arbor, Michigan,* CD-ROM, **B**11.

Hui, Y.V., Leung, L.C. and Linn, R., 2001, "Optimal machining conditions with costs of quality and tool maintenance for turning," *International Journal of Production Research*, **39**(4), pp. 647-665.

Ignizio, J. P. and Cavalier, T. M., 1994a, *Linear Programming*, Englewood Cliffs, NJ, PRENTICE HALL, pp. 523-527.

Ignizio, J. P. and Cavalier, T. M., 1994b, *Linear Programming*, Englewood Cliffs, NJ, PRENTICE HALL, pp. 541-571.

Juan, H., Yu, S.F. and Lee, B.Y. 2003, "The optimal cutting parameter selection of production cost in HSM for SDK61 tool steels," *International Journal of Machine Tools & Manufacture*, **43**(7), pp. 679-686.

Katz, R. and Moon Y.-M., 2000, "Virtual arch type reconfigurable machine tool design: principles and methodology," *NSF ERC Virtual RMT Report* 41, ERC/RMS , University of Michigan, Ann Arbor, MI.

Kennedy, J. and Eberhart, R., 1995, "Particle swarm optimization," *Proceedings of International Conference on Evolutionary Computation*, Perth, Australia, pp. 303-308.

Kennedy, J. and Spears, W.M., 1998, "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," *Proceedings of International Conference on Evolutionary Computation,* Piscataway, NJ, pp. 77-83.

Koren, Y., Pasek, Z.J., Ulsoy, A.G. and Benchetrit, U., 1996, "Real-time open control architectures and system performance," *Annals of the CIRP*, **45**(1), pp. 377-380.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G. and Brussel, H. Van, 1999, "Reconfigurable manufacturing systems," *Annals of the CIRP*, **48**(2), pp 527-540.

Koren, Y. and Kota, S., 1999, "Reconfigurable machine tools," U.S. Patent, No. 5,943,750.

Koren, Y. and Ulsoy, A. G., 1997, "Reconfigurable manufacturing systems," *Engineering Research Center for Reconfigurable Manufacturing Systems (ERC/RMS) Report*1, ERC/RMS, University of Michigan, Ann Arbor, MI.

Koren, Y. and Ulsoy, A. G., 2002a, "Vision, principles and impact of reconfigurable manufacturing systems," *Powertrain International Magazine*, pp. 14-21.

Koren, Y. and Ulsoy, A. F., 2002b, "Reconfigurable manufacturing system having a production capacity method for designing same and method for changing its production capacity," U.S. Patent, US 6,349,237B1.

Landers, R. G., 2000, "A new paradigm in machine tools: reconfigurable machine tools," *Proceedings of the 2000 Japan-USA Flexible Automation Conference,* July 23-26, 2000, Ann Arbor, Michigan, pp. 361-364.

Landers, R.G., Min, B.–K., and Koren, Y., 2001, "Reconfigurable machine tools," *Annals of the CIRP*, **50**(1), pp. 269–274.

Lian, F.-L., Moyne, J.R. and Tilbury, D.M., 2000, "Implementation of networked machine tools in reconfigurable manufacturing systems," *Proceedings of the Japan-USA Symposium on Flexible Automation*, July 23-26, Ann Arbor, MI, pp.1007-1010.

Maier-Speredelozzi, V. and Hu, S.J., 2002, "Selecting manufacturing system configurations based on performance using AHP," *Transactions of NAMRI*, West Lafayette, Indiana, pp. 1–80.

Martello, S. and Toth, P., 1990, *Knapsack problem: algorithms and computer implementations*, Chichester, JOHN WILEY & SONS.

Mehrabi, M.G., Ulsoy A.G. and Koren, Y., 2000a, "Reconfigurable manufacturing systems: key to future manufacturing," *Journal of Intelligent Manufacturing*, **11**(4), pp. 403-419.

Mehrabi, M.G., Ulsoy, A.G. and Koren, Y., 2000b, "Reconfigurable manufacturing system and their enabling technologies," *International Journal of Manufacturing Technology and Management*, **1**(1), pp. 113-130.

Mehrabi, M.G., Ulsoy, A.G., Koren, Y. and Heytler, P., 2002, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing*, **13**(2), pp. 135-146.

Mellor, C., 2002, "Quick-change artists: why techs must get ready for reconfigurable manufacturing," *The Ontario Technologist*, Jan/Feb, pp. 12-15.

Ming, X.G. and Mak, K.L., 2001, "Intelligent approaches to tolerance allocation and manufacturing operations selection in process planning," *Journal of Material Processing Technology*, **117**(1), pp. 75-83.

Mohanty, B. K., and Vijayaraghavan, T. A. S., 1995, "A multi-objective programming problem and its equivalent goal programming problem with appropriate priorities and aspiration levels: a fuzzy approach," *Computer & Operations Research*, **22**(8), pp. 771-778.

Moon, S -K., Landers, Robert G. and Kota, S., 2000, "Error analysis in reconfigurable machine tool design," ERC/RMS Technical Report, University of Michigan, Ann Arbor, Michigan.

Moon, S-K., Moon Y-M. and Kota, S., 2001, "Screw theory based metrology for design and error compensation of machine tools," *Proceedings of ASME 2001 Design Engineering Technical Conferences*, September 9-21, Pittsburgh, PA, pp. 697-707.

Moon, Y. M. and Kota, S., 2002a, "Generalized kinematic modeling of reconfigurable machine tools," *Journal of Mechanical Design*, **124**(1), pp. 47-51.

Moon, Y. M. and Kota, S., 2002b, "Design of reconfigurable machine tools," *Transactions of the ASME*, **124**(2), pp. 480-483.

Okafor, A. C. and Ertekin, Y. M., 2000, "Derivation of machine tool error models and error compensation procedure for three axes vertical machining center using rigid body kinematics," *International Journal of Machine Tools & Manufacture*, **40**(8), pp. 1199-1213.

Otanez, P.G., Moyne, J.R. and Tilbury, D.M., 2002, "Using deadbands to reduce communication in networked control systems," *Proceedings of the 2002 American Control Conference*, Anchorage, AK, **4**, pp.3015-3020.

Park, E., Tilbury, D.M.and Khargonekar, P.P., 2000, "A modeling and analysis methodology for modular logic controllers of machining systems with auto, hand, and

manual modes," *Proceedings of the American Control Conference*, Chicago, pp. 3158-3164.

Pérez R., Aca, S. J., Valverde T. A., Ahuett, G. H., Molina, G. A. and Riba, R. C., 2004, "A modularity framework for concurrent design of reconfigurable machine tools," *Y. Luo(Ed): CDVE 2004, LNCS3190*, Verlag Berlin Heideberg: Spinger, pp. 87-95.

Rogers, G. G. and Bottaci, L., 1997, "Modular production systems: a new manufacturing paradigm," *Journal of Intelligent Manufacturing*, **8**(2), pp. 147-156.

Schutte, J. F., Reinbolt, J. A., Fregly, B. J., Haftka, R. T. and George, A. D., 2004, "Parallel global optimization with the particle swarm algorithm," *International Journal for Numerical Methods in Engineer*, **61**(13), pp. 2296-2315.

SERVO Products Company, 2003, "Price list- SERVO manual knee mills SV50 and SV54," *SERVO Manual Knee Mills Consumer Price List.*

SETCO Sales Company, 2004a, "SETCO boring/milling spindle," *Product List*, Publication S-0002.

SETCO Sales Company, 2004b, "Precision dovetail slides," *Product List*, Publication M-0001-2.

Shi, Y. and Eberhart, R.C., 1998, "A modified particle swarm optimizer," *Proceedings of International Conference on Evolutionary Computation,* Piscataway, NJ, pp. 69-73.

Slocum, A. H., 1992, "Chapter 2: Principle of accuracy, repeatability, and resolution," *Precision Machine Design*, Englewood Cliffs, NJ, PRENTICE HALL, pp. 61-76.

Walczyk, D. F., Lakshmikanthan, J. and Kirk, D. R., 1998, "Development of a reconfigurable tool for forming aircraft body panels," *Journal of Manufacturing systems*, **17**(4), pp. 287-296.

Wang, P., 2003, "Simultaneously solving process selection, machining parameter optimization and tolerance design problems: a bi-criterion approach," *M.A.Sc. Thesis*, Department of Mechanical Engineering, University of Ottawa, Canada.

Wu, Z., ElMaraghy, W.H. and ElMaraghy, H.A., 1988, "Evaluation of cost-tolerance algorithms for design tolerance analysis and synthesis," *ASME manufacturing Review*, **1**(3), pp. 168-179.

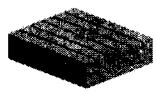
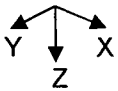
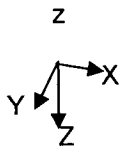
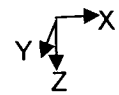
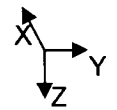
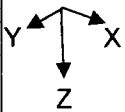
Yeo, S.H. and Hgoi, B.K.A., 1996, "A cost-tolerance model for process sequence optimization," *International Journal of Advanced Manufacturing Technology*, **12**(6), pp. 423-431.

Yigit, Ahmet S., Ulsoy, A. Galip and Allahverdi, Ali, 2002, "Optimizing modular product design for reconfiguring," *Journal of Intelligent Manufacturing*, **13**(4), pp. 309-316.

Zhang, C. and Wang, H.P., 1993, "Optimal process sequence selection and manufacturing tolerance allocation," *Journal of Design and Manufacturing*, **3**, pp. 135-146.

Appendix A - Data of case study

Table A1 Module warehouse of the basic design of a boring machine in case study

	Name	Picture	coordinate	Initial	Index	Instance number	Form tolerance	Dimension of reference surface	Connection dimension L_1xL_2	error on X-axis ($x10^{-5}$ rad)	error on Y-axis ($x10^{-5}$ rad)	error on Z-axis ($x10^{-5}$ rad)
1	Product Fixer			PF	0	2	// 0.025 mm Parallelism	760x500 mm	215x215 mm	4.92	4.13	5.09
2	Main Frame (base)			MF	1	1	// 0.025mm Parallelism	500x305 mm	215x215 mm	6.3	4.92	5.09
3	X-slide Table			XT	2	2	// 0.025 mm in 305x305mm Parallelism	254x254 mm	215x215 mm	8.3	8.3	5.09
4	Y-slide Table			YT	3	2	// 0.025mm in 305x305mm Parallelism	254x254 mm	215x215 mm	8.3	8.3	5.09
5	Z-slide support			ZS	4	1	⊥ 0.020 mm Verticality	254x254 mm	215x215 mm	7.87	0	8.24

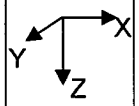
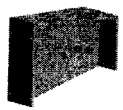
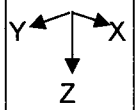
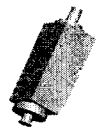
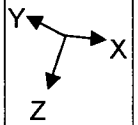
6	Z-slide Table			ZT	5	2	/ / 0.025 mm in 305x305 mm Parallelism	254x254 mm	215x215 mm	8.3	5.09	8.3
7	Head Support			HS	6	1	/ / 0.020mm Parallelism	387x178 mm	215x215 mm	7.87	5.09	7.87
8	Spindle			SP	7	4	↗ ↗ 0.005mm Run-out tolerance	127 mm	279x178 mm	4	6.9	0
9	Transmission		-	TS	14	4	-	-	-	-	-	-
10	Motor		-	MT	15	8	-	-	-	-	-	-
11	X-Slide Driver		-	XD	8	4	-	-	-	-	-	-
	Y-Slide Driver		-	YD	10	4						
	Z-Slide Driver		-	ZD	12	4						
12	X-Slide Hand-Driver		-	XH	9	1	-	-	-	-	-	-
	Y-Slide Hand-Driver		-	YH	11	1						
	Z-Slide Hand-Driver		-	ZH	13	1						

Table A2 Module conjunction matrix of the basic design of a boring machine in the case

	Part fixer	Main frame	X-slide table	Y-slide table	z-support	Z-slide table	Head supporter	Spindle	X-slide driver	X-slide hand driver	Y-slide driver	Y-slide hand driver	Z-slide driver	Z-slide hand driver	Transmission	Motor
Part fixer	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Main frame	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
X-slide table	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
Y-slide table	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0
Z-support	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
Z-slide table	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0
Head supporter	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
Spindle	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
X-slide driver	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
X-slide hand driver	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Y-slide driver	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
Y-slide hand driver	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Z-slide driver	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
Z-slide hand driver	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Transmission	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Motor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table A3 Module instance-requirement matrix of the basic design of a boring machine in the case

	Part fixer	Main frame	X-slide table	Y-slide table	z-support	Z-slide table	Head supporter	Spindle	X-slide driver	X-slide hand driver	Y-slide driver	Y-slide hand driver	Z-slide driver	Z-slide hand driver	Transmission	Motor
Future forecast	1	1	2	2	1	2	1	2	2	1	2	1	2	1	2	2
Shape	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Material	1	1	1	1	1	1	1	1	2	1	2	1	2	1	2	2
Dimension of hole	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	2

Table A4 Module attribute matrix of the basic design of a boring machine in the case

	Part fixer	Main frame	X-slide table	Y-slide table	z-support	Z-slide table	Head supporter	Spindle	X-slide driver	X-slide hand driver	Y-slide driver	Y-slide hand driver	Z-slide driver	Z-slide hand driver	Transmission	Motor
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Initial	PF	MF	XT	YT	ZS	ZT	HS	SP	XD	XH	YD	YH	ZD	ZH	TR	MT
Main_chain index	1	2	3	4	5	6	7	8	0	0	0	0	0	0	0	0
Basic module	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1
Purchasing cost	400	2500	2800	2800	300	2800	300	1622	1075	200	1075	200	1200	300	2000	1200
Storage cost	200	2800	1500	1500	150	1500	200	600	300	250	300	250	350	300	700	300
Exchange cost	20	2000	100	100	100	200	100	55	60	30	150	150	200	30	100	40
Error on x axis 10^{-5} rad	4.92	6.3	8.3	8.3	7.87	8.3	7.87	4	0	0	0	0	0	0	0	0
Error on y axis 10^{-5} rad	4.13	4.92	8.3	8.3	0	5.09	5.09	8.89	0	0	0	0	0	0	0	0
Error on z axis 10^{-5} rad	5.09	5.09	5.09	5.09	8.24	8.3	7.87	0	0	0	0	0	0	0	0	0
Instance number	2	1	2	2	1	2	1	4	4	1	4	1	4	1	4	8

Table A5 Module conjunction matrix of the optimized design of a boring machine in the case

	Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8	Module 9	Module 10
Module 0	0	1	0	0	0	0	0	0	0	0	0
Module 1	1	0	1	1	0	1	0	0	0	0	0
Module 2	0	1	0	0	0	0	0	1	0	1	0
Module 3	0	1	0	0	1	0	0	0	0	0	0
Module 4	0	0	0	1	0	0	0	0	0	0	0
Module 5	0	1	0	0	0	0	1	0	0	0	0
Module 6	0	0	0	0	0	1	0	0	0	0	0
Module 7	0	0	1	0	0	0	0	0	1	0	0
Module 8	0	0	0	0	0	0	0	1	0	0	0
Module 9	0	0	1	0	0	0	0	0	0	0	1
Module 10	0	0	0	0	0	0	0	0	0	1	0

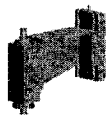
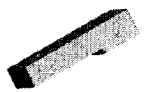
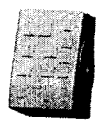
Table A6 Module instance-requirement matrix of the optimized design of a boring machine in the case

	Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8	Module 9	Module 10
Future forecast	1	4	4	2	1	2	1	2	1	2	2
shape	2	1	1	1	1	1	1	1	1	1	1
Material	1	1	1	2	1	2	1	2	1	2	2
Dimension of hole	1	1	2	1	1	1	1	1	1	1	2

Table A7 Module attribute matrix of the optimized design of a boring machine in the case

	Module 0	Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8	Module 9	Module 10
Index	0	1	2	3	4	5	6	7	8	9	10
Initial	PF, MF	XT, YT, ZS	ZT, HS, SP	XD	XH	YD	YH	ZD	ZH	TR	MT
Main_chain index	1	3	6	0	0	0	0	0	0	0	0
Basic module	1	1	1	0	1	0	1	0	1	1	1
Purchasing cost	2900	5900	4722	1075	200	1075	200	1200	300	2000	1200
Storage cost	2940	3030	2180	300	250	300	250	350	300	700	300
Exchange cost	2020	300	355	60	30	150	150	200	30	100	40
Error on X axis 10^{-5} rad	6.3	7.87	4	0	0	0	0	0	0	0	0
Error on Y axis 10^{-5} rad	4.92	0	8.89	0	0	0	0	0	0	0	0
Error on Z axis 10^{-5} rad	5.09	8.24	0	0	0	0	0	0	0	0	0
Instance number	2	4	8	4	1	4	1	4	1	4	8

Table A8 Module warehouse of the optimized design of a boring machine in case study

	Name	Picture	Initial	Index	Instance number	Form tolerance	Dimension of reference surface	Connection dimension L_1xL_2	error on X-axis (x10e-5 rad)	error on Y-axis (x10e-5 rad)	error on Z-axis (x10e-5 rad)
1	Module 0		PF, MF	0	2	// 0.025mm Parallelism	500x305 mm	215x215 mm	6.3	4.92	5.09
2	Module 1		XT, YT, ZS	1	4	⊥ 0.020mm Verticality	254x254 mm	215x215 mm	7.87	0	8.24
3	Module 2		ZT, HS, SP	2	8	↗↗ 0.005mm Run-out tolerance	127 mm	279x178 mm	4	6.9	0
4	Module 9		TS	9	4	-	-	-	-	-	-
5	Module 10		MT	10	8	-	-	-	-	-	-
6	Module 3		XD	3	4	-	-	-	-	-	-
7	Module 5		YD	5	4						
8	Module 7		ZD	7	4						

9	Module 4		XH	4	1	-	-	-	-	-	-
10	Module 6		YH	6	1						
11	Module 8		ZH	8	1						

Appendix B - The LINGO model of example 4.1 by using MFCP

```
MODEL:
MAX = delta1;
delta1<=((3*x1+x2)-60)/(70-60);
delta2<=((2*x1+3*x2)-70)/(110-70);
delta2=((3*x1+x2)-60)*delta1/(70-60);
2*x1+x2<=50;                                    (constraint)
x1<=20;                                         (constraint)
x2<=30;                                         (constraint)
z1=3*x1+x2;                                     (objective 1)
z2=2*x1+3*x2;                                   (objective 2)
w=((3*x1+x2)-60)/(70-60);                       (weight function)
END
```

LINGO output:

```
Global optimal solution found at iteration:           1
  Objective value:                                0.6180340

                     Variable           Value        Reduced Cost
                       DELTA1       0.6180340            0.000000
                           X1        16.18034            0.000000
                           X2        17.63932            0.000000
                       DELTA2       0.3819660            0.000000
                           Z1        66.18034            0.000000
                           Z2        85.27864            0.000000
                            W       0.6180340            0.000000
```

Appendix C - The LINGO model of example 4.1 by using MFCP

```
MODEL:
MIN = delta;
(3*x1+x2)+(70-60)*delta>=70;
(2*x1+3*x2)+(110-70)*delta>=110;
2*x1+x2<=50;
x1<=20;
x2<=30;
z1=3*x1+x2;
z2=2*x1+3*x2;
END
```

LINGO output:

```
 Global optimal solution found at iteration:               3
  Objective value:                                 0.5000000

                      Variable           Value        Reduced Cost
                         DELTA       0.5000000            0.000000
                            X1        15.00000            0.000000
                            X2        20.00000            0.000000
                            Z1        65.00000            0.000000
                            Z2        90.00000            0.000000
```

Appendix D - The LINGO model for example 4.2 by using MFCP

```
model:

max= delta1;

delta1<=((4*x1+x2)-l1)/(u1-l1);
delta2<=((3*x1+2*x2)-l2)/(u2-l2);
delta3<=((x1+2*x2)-l3)/(u3-l3);

((3*x1+2*x2)-l2)*delta1=(u2-l2)*delta2;
((4*x1+x2)-l1)*delta2=(u1-l1)*delta1;
!constraints;
x2+x1<=5;
x2+2*x1<=7;
x2-4<=0;
x1-3<=0;
x1>=0;
x2>=0;
!assignment;
z1=4*x1+x2;                                    (objective z1)
z2=3*x1+2*x2;                                  (objective z2)
z3=x1+2*x2;                                    (objective z3)
u1=13;                                         (up-limit of Z1)
u2=12;                                         (up-limit of Z2)
u3=9;                                          (up-limit of Z3)
l1=8;                                          (low-limit of Z1)
l2=11;                                         (low-limit of Z2)
l3=5;                                          (low-limit of Z3)
w1=((4*x1+x2)-l1)/(u1-l1);                     (weight function)
w2=((3*x1+2*x2)-l2)/(u2-l2);                   (weight function)
end
```

LINGO output:

```
Objective value:                              0.7655644

                    Variable           Value        Reduced Cost
                      DELTA1       0.7655644            0.000000
                          X1        2.413911            0.000000
                          X2        2.172178            0.000000
                          L1        8.000000            0.000000
                          U1        13.00000            0.000000
                      DELTA2       0.5860889            0.000000
                          L2        11.00000            0.000000
                          U2        12.00000            0.000000
                      DELTA3       0.3435002            0.000000
                          L3        5.000000            0.000000
```

U3	9.000000	0.000000
Z1	11.82782	0.000000
Z2	11.58609	0.000000
Z3	6.758267	0.000000
W1	0.7655644	0.000000
W2	0.5860889	0.000000

Appendix E - Lingo model for example 4.2 by using FCP

```
model:          !(this model use the fuzzy-Chebyshev method)

max= delta1;
delta1<=((4*x1+x2)-l1)/(u1-l1);
delta2<=((3*x1+2*x2)-l2)/(u2-l2);
delta3<=((x1+2*x2)-l3)/(u3-l3);
delta1=delta2;
delta2=delta3;

!constraints;
x2+x1<=5;
x2+2*x1<=7;
x2-4<=0;
x1-3<=0;
x1>=0;
x2>=0;

!assignment;
z1=4*x1+x2;
z2=3*x1+2*x2;
z3= x1+2*x2;
u1=13;
u2=12;
u3=9;
l1=8;
l2=11;
l3=5;
end
```

LINGO output:

```
  Global optimal solution found at iteration:            4
  Objective value:                                0.6521739

                      Variable           Value        Reduced Cost
                        DELTA1       0.6521739            0.000000
                            X1        2.130435            0.000000
                            X2        2.739130            0.000000
                            L1        8.000000            0.000000
                            U1        13.00000            0.000000
                        DELTA2       0.6521739            0.000000
                            L2        11.00000            0.000000
                            U2        12.00000            0.000000
                        DELTA3       0.6521739            0.000000
                            L3        5.000000            0.000000
                            U3        9.000000            0.000000
                            Z1        11.26087            0.000000
```

Z2	11.86957	0.000000
Z3	7.608696	0.000000

Appendix F - C++ code of case study

```
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <string.h>

#define requirement_num 4    // number of row in the instance-requirment matrix
#define module_num 16        // number of modules in the basic RMT design
#define population_size 100  // number of particles
#define iteration_num 2000   // maximum iterations number
#define cost_limit 500000    // up-limit of the affordable cost
#define movementx 250        // maximum displacement of tool along x axis
#define movementy 250        // maximum displacement of tool along y axis
#define movementz 250        // maximum displacement of tool along z axis

#define omiga 1              //parameters of PSOA
#define c1 0.5
#define c2 0.5
#define c3 0.5

#define min(A,B) ((A) <= (B)? (A) : (B))
#define max(A,B) ((A) >= (B)? (A) : (B))
#define XOR(A,B) ((A) != (B)? 1 : 0)

int conf_min, conf_max;
float cost_min, cost_max;
float err_x_max, err_x_min;
float err_y_max, err_y_min;
```

```
float err_z_max, err_z_min;

int merge_module_size;
int interface_size=pow(2, module_num-1);
struct interface_struct
  {int att;
   int module1_num;
   int module2_num;
  };
struct interface_struct inter_face[module_num-1];
struct interface_struct temp_interface[module_num-1];

struct PSO_struct
  {int interface_num;
   float fitness;
  };
PSO_struct population[population_size];

struct module_att
  {int index;
   char name[50];
   int main_chain_index;
   bool basic_module;
   float pcost,scost,icost;
   float error_x, error_y, error_z;
   int instance_num;
  };
module_att module[module_num];
module_att temp_module[module_num];

int instance_require_matrix[requirement_num][module_num];
```

```
int temp_instance_require_matrix[requirement_num][module_num];
int module_matrix[module_num][module_num];
int temp_module_matrix[module_num][module_num];

int merge_instance(int ma, int mb)
 {int temp_instance_num=1;
  for (int i=0; i<requirement_num; i++)
   {if (i==0)
     {temp_instance_require_matrix[i][ma]=
temp_instance_require_matrix[i][ma]*temp_instance_require_matrix[i][mb];
      temp_instance_num=temp_instance_num * temp_instance_require_matrix[i][ma];
      for (int j=mb;j<merge_module_size-1; j++)
           {temp_instance_require_matrix[i][j]=temp_instance_require_matrix[i][j+1];}
     }
    else

{temp_instance_require_matrix[i][ma]=max(temp_instance_require_matrix[i][ma],temp_
instance_require_matrix[i][mb]);
      temp_instance_num=temp_instance_num * temp_instance_require_matrix[i][ma];
      for (int j=mb;j<merge_module_size-1; j++)
           {temp_instance_require_matrix[i][j]=temp_instance_require_matrix[i][j+1];}
     }
   }
  return(temp_instance_num);
 }

void merge_module_att(int m1,int m2)
 {temp_module[m1].instance_num=merge_instance(m1,m2);
  temp_module[m1].index=min(temp_module[m1].index, temp_module[m2].index);
  for (int i=m2;i<merge_module_size; i++)
       {temp_module[i].index--;}
```

```
  strcat(temp_module[m1].name,", ");
  strcat(temp_module[m1].name,temp_module[m2].name);
  temp_module[m1].main_chain_index=min(temp_module[m1].main_chain_index,
temp_module[m2].main_chain_index);
  temp_module[m1].basic_module=(temp_module[m1].basic_module ||
temp_module[m2].basic_module);
  temp_module[m1].pcost+=temp_module[m2].pcost;
  temp_module[m1].scost+=temp_module[m2].scost;
  temp_module[m1].icost+=temp_module[m2].icost;
  if (temp_module[m1].main_chain_index< temp_module[m2].main_chain_index)
    {temp_module[m1].error_x=temp_module[m2].error_x;
     temp_module[m1].error_y=temp_module[m2].error_y;
     temp_module[m1].error_z=temp_module[m2].error_z;
    }
   for (int i=m2;i<merge_module_size-1; i++)
        {temp_module[i]=temp_module[i+1];}
  }

 void merge_module_matrix(int a, int b, int end_num)
 {for (int i=0;i<end_num ;i++)

 {temp_module_matrix[a][i]=XOR(temp_module_matrix[a][i],temp_module_matrix[b][i]
 );}
  for (int j=0; j<end_num ;j++)

 {temp_module_matrix[j][a]=XOR(temp_module_matrix[j][a],temp_module_matrix[j][b]
 );}
  for (int m=b; m<end_num ;m++)
    {for (int i=0; i<end_num ;i++)
      {temp_module_matrix[m][i]=temp_module_matrix[m+1][i];}
    }
```

```
 for (int n=b; n<end_num ;n++)
    {for (int j=0; j<end_num ;j++)
      {temp_module_matrix[j][n]=temp_module_matrix[j][n+1];}
    }
}

float cal_error_x_of(int end_num)
{float err_x=0.0;
 for (int i=0; i<end_num; i++)
         {err_x += temp_module[i].error_z* movementy +
temp_module[i].error_y*movementz;          }
 return (err_x);
}

float cal_error_y_of(int end_num)
{float err_y=0.0;
 for (int i=0; i<end_num;i++)
   err_y = err_y + temp_module[i].error_z * movementx +
temp_module[i].error_x*movementz;
 return (err_y);
}

float cal_error_z_of(int end_num)
{float err_z=0.0;
 for (int i=0; i<end_num;i++)
   err_z = err_z + temp_module[i].error_y*movementx +
temp_module[i].error_x*movementy;
 return (err_z);
}

float cal_delta(float conf_num, float cost, float err_x, float err_y, float err_z)
```

```
{float delta, alpha1,alpha2,alpha3,alpha4,alpha5;
 alpha1=(conf_num - conf_min)/(conf_max -conf_min);
 alpha2=(cost_max - cost)/(cost_max - cost_min);
 alpha3=(err_x_max- err_x)/(err_x_max - err_x_min);
 alpha4=(err_y_max- err_y)/(err_y_max - err_y_min);
 alpha5=(err_z_max- err_z)/(err_z_max - err_z_min);
 if (alpha1==0)
   {return (alpha1);}
 if (alpha2==0)
   {return (alpha2);}
 float a= alpha2/alpha1;
 float b= alpha3/alpha2/alpha1;
 float c= alpha4/alpha2/alpha1;
 float d= alpha5/alpha2/alpha1;
 delta= min(alpha1, a);
 delta= min(delta, b);
 delta= min(delta, c);
 delta= min(delta, d);
 return (delta);
}

int cal_conf_num_of(int end_num) //calculate the configuration number of current design
{int bconf_num=1;
 int lconf_num=1;
 int temp_conf_num;
 for (int i=0; i<end_num; i++)
   {if (temp_module[i].basic_module)
     {bconf_num*=temp_module[i].instance_num;}
     else {lconf_num*=(temp_module[i].instance_num + 1);}
   }
 temp_conf_num=bconf_num*lconf_num;
```

```
 return(temp_conf_num);
}

float cal_cost_of(int end_num)   //calculate the cost of current design
{float temp_cost=0;
        for (int i=0; i<end_num; i++)
                {temp_cost= temp_cost+
temp_module[i].pcost*temp_module[i].instance_num +
temp_module[i].scost*(temp_module[i].instance_num-1) +
temp_module[i].icost*(temp_module[i].instance_num-1);}
 return(temp_cost);
}

struct PSO_struct MAX(PSO_struct a,PSO_struct b)
{if (a.fitness> b.fitness)
        {return(a);}
   else
        {return(b);}
}

struct PSO_struct MIN(PSO_struct a,PSO_struct b)
{if (a.fitness<= b.fitness)
        {return(a);}
   else
        {return(b);}
}

void TEMP_COPY()                // make a copy of original condition
{merge_module_size = module_num;
 for (int j=0; j<module_num; j++)
        {temp_module[j]=module[j];
```

```
    for (int i=0; i<requirement_num; i++)
            {temp_instance_require_matrix[i][j]=instance_require_matrix[i][j];}
          for (int m=0; m<module_num; m++)
       {temp_module_matrix[j][m]=module_matrix[j][m];}
   }
}

void convert(int a)//convert a number into binary string and generate a desing according
that string
{merge_module_size = module_num;
 TEMP_COPY();//make a temperay module_att matrix and give it a copy of original one
 for (int j=0; j<(module_num-1); j++)
         {temp_interface[j].att= (a % 2);
    a/=2;//convert 'a' into binary number
          temp_interface[j].module1_num=inter_face[j].module1_num;
          temp_interface[j].module2_num=inter_face[j].module2_num;
          if (temp_interface[j].att==0)

         {merge_module_att(temp_interface[j].module1_num,temp_interface[j].module2_
num);

merge_module_matrix(temp_interface[j].module1_num,temp_interface[j].module2_num,
merge_module_size);
                  merge_module_size-=1;
                 }
        }
}

struct PSO_struct MAX_END(PSO_struct a,PSO_struct b)
{if (a.fitness > b.fitness)
         {return(a);}
```

```
if (a.fitness < b.fitness)
        {return(b);}
if (a.fitness == b.fitness)
      {convert(a.interface_num);
       int conf_a = cal_conf_num_of(merge_module_size);
       float cost_a = cal_cost_of(merge_module_size);
       float errx_a = cal_error_x_of(merge_module_size);
       float erry_a = cal_error_y_of(merge_module_size);
       float errz_a = cal_error_z_of(merge_module_size);
       float ave_err_a= (errx_a + erry_a + errz_a)/3;
       convert(a.interface_num);
       int conf_b = cal_conf_num_of(merge_module_size);
       float cost_b = cal_cost_of(merge_module_size);
       float errx_b = cal_error_x_of(merge_module_size);
       float erry_b = cal_error_y_of(merge_module_size);
       float errz_b = cal_error_z_of(merge_module_size);
       float ave_err_b= (errx_b + erry_b + errz_b)/3;
       if (conf_a > conf_b)
         {return (a);}
       if (conf_a < conf_b)
         {return (b);}
       if (conf_a == conf_b)
         {if (cost_a < cost_b)
            {return (a);}
          if (cost_a > cost_b)
            {return (b);}
          if (cost_a == cost_b)
            {if (ave_err_a < ave_err_b)
               {return (a);}
             else
               {return (b);}
```

```
        }
      }
    }
}

float cal_fitness(int a)// calculate the fitness of a design according a digit number
{convert(a);
 int temp_conf = cal_conf_num_of(merge_module_size);
 float temp_cost = cal_cost_of(merge_module_size);
 float temp_errorx = cal_error_x_of(merge_module_size);
 float temp_errory = cal_error_y_of(merge_module_size);
 float temp_errorz = cal_error_z_of(merge_module_size);
 float fitness = cal_delta(temp_conf,temp_cost,temp_errorx,temp_errory,temp_errorz);
 return (fitness);
}

float cal_cost(int m)
{convert(m);
 float cal_cost=cal_cost_of(merge_module_size);
 return (cal_cost);
}

float cal_conf(int m)
{convert(m);
 float cal_conf=cal_conf_num_of(merge_module_size);
 return (cal_conf);
 }

 void input_data()
 { // get module matrix
  FILE *pf1;
```

```
pf1=fopen("module_matrix.txt","r+t");
for (int i=0; i<module_num; i++)
   {for (int j=0; j<module_num; j++)
     {fscanf(pf1,"%d",&module_matrix[i][j]);}
   }
fclose(pf1);
// input module[] (module attribute matrix)
pf1=fopen("module_att_matrix.txt","r+t");
for (int l=0; l<module_num; l++)
   {fscanf(pf1,"%d", &module[l].index);
    fscanf(pf1,"%s", &module[l].name);
    fscanf(pf1,"%d", &module[l].main_chain_index);
    fscanf(pf1,"%d", &module[l].basic_module);
    fscanf(pf1,"%f", &module[l].pcost);
    fscanf(pf1,"%f", &module[l].scost);
    fscanf(pf1,"%f", &module[l].icost);
    fscanf(pf1,"%f", &module[l].error_x);
    fscanf(pf1,"%f", &module[l].error_y);
    fscanf(pf1,"%f", &module[l].error_z);
    fscanf(pf1,"%d", &module[l].instance_num);
   }
fclose(pf1);
// input instance_require_matrix[][](instances-requirment matrix)
pf1=fopen("instance_require_matrix.txt","r+t");
for (int m=0; m<requirement_num; m++)
   {for (int n=0; n<module_num; n++)
     {fscanf(pf1,"%d",&instance_require_matrix[m][n]);}
   }
fclose(pf1);
int c=0;
for (int a=(module_num-1); a>=0; a--)
```

```
        {for (int b=0; b<a; b++)
                {if (module_matrix[a][b]==1)
                        {inter_face[c].att=1;
                         inter_face[c].module1_num=b;
                         inter_face[c].module2_num=a;
                         c++;
                        }
                }
        }
}

int find_cost_min() // find the design which has the lowest cost by using PSO method
{//calculate the cost_max
 srand(unsigned(time(NULL)));
 int volecity[population_size];
 int interface_size=pow(2,module_num-1);
 int volecity_max= gama*pow(2,module_num-1);
 int quit_index=0;
 int Ldistance, Gdistance,Rdistance;
 int random_ps;
 float Old_Gbest_fitness;
 PSO_struct Lbest[population_size];
 PSO_struct Gbest;
 Gbest.interface_num = 0;
 Gbest.fitness = 100000000;
 for (int i=0; i<population_size; i++)
        {if (rand()%2==0)
       {volecity[i]=(-1)*rand()%volecity_max;}
    else
       {volecity[i]=rand()%volecity_max;}
```

```
        population[i].interface_num=rand()%interface_size;//random generate interface
array
        population[i].fitness = cal_cost(population[i].interface_num);
    random_ps=rand()%population_size;
        Lbest[i] = population[i];
        Gbest= MIN(Lbest[i], Gbest);
        }
  for (int j=0; j<iteration_num; j++)
   {//generation new populations
   for (int i=0; i<population_size; i++)
     {if ((Lbest[i].interface_num - population[i].interface_num)<0)
            {Ldistance =0- rand()%(Lbest[i].interface_num -
 population[i].interface_num);}
      if ((Lbest[i].interface_num - population[i].interface_num)==0)
            {Ldistance = 0;}
          if ((Lbest[i].interface_num - population[i].interface_num)>0)
            {Ldistance = rand()%(Lbest[i].interface_num - population[i].interface_num);}
      if ((Gbest.interface_num -  population[i].interface_num)<0)
        {Gdistance = 0-rand()%(Gbest.interface_num -  population[i].interface_num);}
      if ((Gbest.interface_num -  population[i].interface_num)==0)
            {Gdistance = 0;}
          if ((Gbest.interface_num -  population[i].interface_num)>0)
            {Gdistance = rand()%(Gbest.interface_num -  population[i].interface_num);}
      if ((population[random_ps].interface_num -  population[i].interface_num)<0)
        {Rdistance = 0-rand()%(population[random_ps].interface_num  -
  population[i].interface_num);}
      if ((population[random_ps].interface_num  -  population[i].interface_num)==0)
            {Rdistance = 0;}
          if ((population[random_ps].interface_num  -  population[i].interface_num)>0)
            {Rdistance = rand()%(population[random_ps].interface_num  -
  population[i].interface_num);}
```

```
            volecity[i] = omiga*volecity[i]+c1*Ldistance + c2*Gdistance +c3*Rdistance;
            if (volecity[i]> volecity_max)
            {volecity[i]=volecity_max;}
            else
            {if (volecity[i]<(0-volecity_max))
                  {volecity[i]=(0-volecity_max);}
            }
            population[i].interface_num += volecity[i];
            if (population[i].interface_num> (interface_size-1))
            {population[i].interface_num=interface_size-1;}
            else
            {if (population[i].interface_num<0)
                  {population[i].interface_num=0;}
            }
      population[i].fitness= cal_cost(population[i].interface_num);
      Lbest[i] = MIN(Lbest[i],population[i]);
      Gbest = MIN(population[i],Gbest);
     }
   }
  return (Gbest.interface_num);
}

int find_conf_max() // find the design which has the lowest cost by using PSO method
{//calculate the cost_max
 srand(unsigned(time(NULL)));
 int volecity[population_size];
 int interface_size=pow(2,module_num-1);
 int volecity_max= gama*pow(2,module_num-1);
 int quit_index=0;
 int Ldistance, Gdistance,Rdistance;
 int random_ps;
```

```
float Old_Gbest_fitness;
PSO_struct Lbest[population_size];
PSO_struct Gbest;
Gbest.interface_num = 0;
Gbest.fitness = 0;
for (int i=0; i<population_size; i++)
        {if (rand()%2==0)
      {volecity[i]=(-1)*rand()%volecity_max;}
   else
      {volecity[i]=rand()%volecity_max;}
         population[i].interface_num=rand()%interface_size;//random generate interface
array
         population[i].fitness = cal_conf(population[i].interface_num);
   random_ps=rand()%population_size;
         Lbest[i] = population[i];
         Gbest= MAX(Lbest[i], Gbest);
        }
 for (int j=0; j<iteration_num; j++)
  {//generation new populations
  for (int i=0; i<population_size; i++)
    {if ((Lbest[i].interface_num - population[i].interface_num)<0)
            {Ldistance =0- rand()%(Lbest[i].interface_num -
population[i].interface_num);}
     if ((Lbest[i].interface_num - population[i].interface_num)==0)
            {Ldistance = 0;}
          if ((Lbest[i].interface_num - population[i].interface_num)>0)
            {Ldistance = rand()%(Lbest[i].interface_num - population[i].interface_num);}
     if ((Gbest.interface_num -  population[i].interface_num)<0)
       {Gdistance = 0-rand()%(Gbest.interface_num -  population[i].interface_num);}
     if ((Gbest.interface_num -  population[i].interface_num)==0)
            {Gdistance = 0;}
```

```
            if ((Gbest.interface_num -  population[i].interface_num)>0)
              {Gdistance = rand()%(Gbest.interface_num -  population[i].interface_num);}
        if ((population[random_ps].interface_num -  population[i].interface_num)<0)
          {Rdistance = 0-rand()%(population[random_ps].interface_num -
population[i].interface_num);}
        if ((population[random_ps].interface_num  -  population[i].interface_num)==0)
              {Rdistance = 0;}
            if ((population[random_ps].interface_num  -  population[i].interface_num)>0)
              {Rdistance = rand()%(population[random_ps].interface_num  -
population[i].interface_num);}
            volecity[i] = omiga*volecity[i]+c1*Ldistance + c2*Gdistance +c3*Rdistance;
            if (volecity[i]> volecity_max)
            {volecity[i]=volecity_max;}
            else
            {if (volecity[i]<(0-volecity_max))
                  {volecity[i]=(0-volecity_max);}
            }
            population[i].interface_num += volecity[i];
            if (population[i].interface_num> (interface_size-1))
            {population[i].interface_num=interface_size-1;}
            else
            {if (population[i].interface_num<0)
                  {population[i].interface_num=0;}
            }
        population[i].fitness= cal_conf(population[i].interface_num);
        Lbest[i] = MAX(Lbest[i],population[i]);
        Gbest = MAX(population[i],Gbest);
       }
    }
  return (Gbest.interface_num);
}
```

```
void main()
{int volecity[population_size];
 int interface_size=pow(2,module_num-1);
 int volecity_max= gama*pow(2,module_num-1);
 int quit_index=0;
 PSO_struct Lbest[population_size];
 PSO_struct Gbest;
 PSO_struct Best;

 srand(unsigned(time(NULL)));

 input_data();
 //find the upper and lower limit of cost, configuration number and error on each direction
 int temp=find_cost_min();//only optimize cost
 convert(temp);
 int temp_conf = cal_conf_num_of(merge_module_size);
 float temp_cost = cal_cost_of(merge_module_size);
 float temp_errorx = cal_error_x_of(merge_module_size);
 float temp_errory = cal_error_y_of(merge_module_size);
 float temp_errorz = cal_error_z_of(merge_module_size);
 conf_min=temp_conf;
 cost_min=temp_cost;//the lower limit of the cost
 err_x_max=temp_errorx;
 err_y_max=temp_errory;
 err_z_max=temp_errorz;

 temp= find_conf_max();//only optimize configuration
 convert(temp);
 temp_conf = cal_conf_num_of(merge_module_size);
```

```
temp_cost = cal_cost_of(merge_module_size);
temp_errorx = cal_error_x_of(merge_module_size);
temp_errory = cal_error_y_of(merge_module_size);
temp_errorz = cal_error_z_of(merge_module_size);
conf_max=temp_conf;// the up limit of configuration number
cost_max=temp_cost;
err_x_max=max(temp_errorx,err_x_max);// the upper limit of the error on x direction
err_y_max=max(temp_errory,err_y_max);// the upper limit of the error on y direction
err_z_max=max(temp_errorz,err_z_max);// the upper limit of the error on z direction

convert(0);//only optimize error
temp_conf = cal_conf_num_of(merge_module_size);
temp_cost = cal_cost_of(merge_module_size);
temp_errorx = cal_error_x_of(merge_module_size);
temp_errory = cal_error_y_of(merge_module_size);
temp_errorz = cal_error_z_of(merge_module_size);
conf_min=min(temp_conf,conf_min);//the lower limit ofthe configuration number
cost_max=max(temp_cost,cost_max);// the upper limit of the cost
err_x_min=temp_errorx; //the lower limit of the error on x direction
err_y_min=temp_errory; //the lower limit of the error on y direction
err_z_min=temp_errorz; //the lower limit of the error on z direction

cost_max=min(cost_max, cost_limit);
//find the best design by using PSO method
int random_ps;
int Ldistance, Gdistance, Rdistance;
Gbest.interface_num = -1;
Gbest.fitness = -1;
Best.interface_num = -1;
Best.fitness = -1;
quit_index = 0;
```

```
//generate initial population
 for (int i=0; i<population_size; i++)
        {if (rand()%2==0)
       {volecity[i]=(-1)*rand()%volecity_max;}
    else
       {volecity[i]=rand()%volecity_max;}
         population[i].interface_num=rand()%(interface_size);//random generate interface
array
    if (cal_cost(population[i].interface_num)> cost_max)
     {population[i].fitness=-10;}
     else
          {population[i].fitness = cal_fitness(population[i].interface_num);}
    Lbest[i] = population[i];
         Gbest= MAX(Lbest[i], Gbest);
        }
 // iteration begin
 float Old_Gbest_fitness=Gbest.fitness;
 int count;
 int temp_num;
 for (int j=0; j<iteration_num; j++)
   {Old_Gbest_fitness=Gbest.fitness;//generation new populations
    for (int i=0; i<population_size; i++)
       {if ((Lbest[i].interface_num - population[i].interface_num)<0)
              {Ldistance =0- rand()%(Lbest[i].interface_num -
 population[i].interface_num);}
        if ((Lbest[i].interface_num - population[i].interface_num)==0)
              {Ldistance = 0;}
        if ((Lbest[i].interface_num - population[i].interface_num)>0)
              {Ldistance = rand()%(Lbest[i].interface_num -
 population[i].interface_num);}
        if ((Gbest.interface_num -  population[i].interface_num)<0)
```

```
        {Gdistance = 0-rand()%(Gbest.interface_num -  population[i].interface_num);}
      if ((Gbest.interface_num -  population[i].interface_num)==0)
            {Gdistance = 0;}
      if ((Gbest.interface_num -  population[i].interface_num)>0)
            {Gdistance = rand()%(Gbest.interface_num -  population[i].interface_num);}
      random_ps=rand()%population_size;
      if ((population[random_ps].interface_num -  population[i].interface_num)<0)
        {Rdistance = 0-rand()%(population[random_ps].interface_num -
population[i].interface_num);}
      if ((population[random_ps].interface_num  -  population[i].interface_num)==0)
            {Rdistance = 0;}
      if ((population[random_ps].interface_num  -  population[i].interface_num)>0)
            {Rdistance = rand()%(population[random_ps].interface_num -
population[i].interface_num);}
      volecity[i] = omiga*volecity[i]+c1*Ldistance + c2*Gdistance +c3*Rdistance;
      if (volecity[i]> volecity_max)
            {volecity[i]=volecity_max;}
          else
           {if (volecity[i]<(0-volecity_max))
                 {volecity[i]=(0-volecity_max);}
           }
       population[i].interface_num = population[i].interface_num + volecity[i];
       if (population[i].interface_num> (interface_size-1))
            {population[i].interface_num=interface_size-1;}
          else
           {if (population[i].interface_num<0)
                 {population[i].interface_num=0;}
           }
     if (cal_cost(population[i].interface_num)>cost_max)
       {population[i].fitness=-10;}
     else
```

```
        {population[i].fitness= cal_fitness(population[i].interface_num);}
      Lbest[i] = MAX(Lbest[i],population[i]);
      Gbest = MAX(Lbest[i],Gbest);
     }
   if (Gbest.fitness > Old_Gbest_fitness)
    {count=j;}
 }

convert(Gbest.interface_num);// generate the detail of the best design
//output_data();
int best_conf=cal_conf_num_of(merge_module_size);
float best_cost=cal_cost_of(merge_module_size);
float best_errorX=cal_error_x_of(merge_module_size);
float best_errorY=cal_error_y_of(merge_module_size);
float best_errorZ=cal_error_z_of(merge_module_size);
// out put the data
FILE *pf2;
pf2=fopen("output1.txt","w+t");

// output temp_module_matrix[][]
fprintf(pf2,"Module adjuction matrix:\n");
for (int i=0; i<merge_module_size; i++)
    {for (int j=0; j<merge_module_size; j++)
      {fprintf(pf2," %d",temp_module_matrix[i][j]);}
     fprintf(pf2,"\n");
    }
fprintf(pf2,"\n");
// output module[] (module attribute matrix)
fprintf(pf2,"Module attribute matrix:\n");
for (int l=0; l<merge_module_size; l++)
    {fprintf(pf2," %d", temp_module[l].index);
```

```
    fprintf(pf2," %s", temp_module[l].name);
    fprintf(pf2," %d", temp_module[l].main_chain_index);
    fprintf(pf2," %d", temp_module[l].basic_module);
    fprintf(pf2," %f", temp_module[l].pcost);
    fprintf(pf2," %f", temp_module[l].scost);
    fprintf(pf2," %f", temp_module[l].icost);
    fprintf(pf2," %f", temp_module[l].error_x);
    fprintf(pf2," %f", temp_module[l].error_y);
    fprintf(pf2," %f", temp_module[l].error_z);
    fprintf(pf2," %d", temp_module[l].instance_num);
    fprintf(pf2," \n");
   }
 fprintf(pf2,"\n");
// output instance_require_matrix[][](instances-requirment matrix)
 fprintf(pf2,"Instance-requirement matrix:\n");
 for (int m=0; m<requirement_num; m++)
   {for (int n=0; n<merge_module_size; n++)
     {fprintf(pf2," %d",temp_instance_require_matrix[m][n]);}
    fprintf(pf2,"\n");
   }
 fprintf(pf2,"\n");
 fprintf(pf2,"Interface matrix:\n");
 int u=0;
 for (int v=(merge_module_size-1); v>=0; v--)
       {for (int w=0; w<v; w++)
             {if (temp_module_matrix[v][w]==1)
                   {temp_interface[u].att=1;
                    temp_interface[u].module1_num=w;
                    temp_interface[u].module2_num=v;
                    u++;
                   }
```

```
            }
        }
for (int a=0; a<merge_module_size-1; a++)
  {fprintf(pf2,"  %d",temp_interface[a].att);
   fprintf(pf2,"  %d",temp_interface[a].module1_num);
   fprintf(pf2," %d\n",temp_interface[a].module2_num);
  }
fprintf(pf2,"\n");
fprintf(pf2,"\n%f",Gbest.fitness);
fprintf(pf2,"\n Best Configuration=%d",best_conf);
fprintf(pf2,"\n Best Cost=%f",best_cost);
fprintf(pf2,"\n Best Error on X=%f",best_errorX);
fprintf(pf2,"\n Best Error on Y=%f",best_errorY);
fprintf(pf2,"\n Best Error on Z=%f",best_errorZ);
fprintf(pf2,"\n");
fprintf(pf2,"\n Iteration=%d",count);
fclose(pf2);
}
```