Parameter identification and damage detection using evolutionary programming

Xu, Jianfeng

2005

Xu, J. F. (2005). Parameter identification and damage detection using evolutionary programming. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/12132>

<https://doi.org/10.32657/10356/12132>

Nanyang Technological University

Downloaded on 26 Apr 2023 16:23:46 SGT

PARAMETER IDENTIFICATION AND DAMAGE DETECTION USING EVOLUTIONARY PROGRAMMING

XU JIANFENG

**SCHOOL OF CIVIL AND ENVIRONMENTAL ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY**

2005

Parameter Identification and Damage Detection Using Evolutionary Programming

Xu Jianfeng
B.Eng., M.Eng

School of Civil and Environmental Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirements for the degree of
Doctor of Philosophy

2005

This thesis is dedicated
to my parents,
to my wife,
and to my baby.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Professor Soh Chee Kiong for his profound vision, his excellent guidance, patience and encouragement in conducting this research. It has been a great privilege and honor to work with him. I wish for him and his family all the happiness and joy life has to give.

I would like to acknowledge and thank the colleagues working in the same research group for helpful discussions and suggestions. Specifically, I would like to thank Doctor Yang Yaowen for his helpful ideas, information and supports in this endeavor.

The colleagues in the laboratories have aided me immeasurably. I am very grateful to them for all that they have given me.

The Research Scholarship offered by Nanyang Technological University is gratefully acknowledged.

I would like very much to thank my family for their continuous support. Most of all, I would like to express my deepest appreciation to my wife, Yumin, for her love, understand and encouragement, all of which made this effort possible.

TABLE OF CONTENTS

ACKNOWLEDGMENTS..... ii

TABLE OF CONTENTS iii

SUMMARY viii

LIST OF FIGURES ix

LIST OF TABLES xi

LIST OF SYMBOLS xii

CHAPTER 1 INTRODUCTION 1

1.1 Background..... 1

1.2 Objectives and Contributions 4

1.3 Scope of Work 6

1.4 Organization of Thesis..... 7

CHAPTER 2 EVOLUTIONARY COMPUTATION 9

2.1 Evolutionary Computation Methods 9

2.1.1 Emergence 9

2.1.2 Procedure 10

2.1.3 Advantages and Disadvantages 12

2.1.4 Similarities and Differences 14

2.2 Evolutionary Programming 16

2.2.1 History 17

2.2.2 Development..... 19

2.3 EC applications in Engineering 27

CHAPTER 3 A HYBRID EP ALGORITHM 34

3.1 Introduction 34

3.2	Algorithm.....	35
3.2.1	Representation Structure.....	36
3.2.2	Variation Operators	36
3.2.3	Handling Constraints	39
3.2.4	Selection	41
3.2.5	Procedure	42
3.3	Case Study and Overall Assessment of Algorithm	43
3.3.1	Test Functions	43
3.3.2	Parameters Adopted	45
3.3.3	Results and Comparison	47
3.4	Influence of Different Operators	51
3.5	Summary.....	62

CHAPTER 4 PARAMETER IDENTIFICATION OF COMPOSITE PLATES

		64
4.1	Introduction	64
4.2	Structural Analysis	66
4.2.1	Laminae	66
4.2.2	Laminates.....	69
4.2.3	Energy Functions	71
4.2.4	Shape Functions.....	74
4.2.5	Eigenvalue Equation.....	76
4.3	Parameter Identification	79
4.4	Summary.....	86

CHAPTER 5 INTEGRATED EP AND IMPEDANCE-BASED TECHNIQUE

	FOR DAMAGE DETECTION	87
5.1	Introduction	87
5.2	Traditional NDE Techniques	88
5.2.1	Visual Inspection	88
5.2.2	Eddy Current Methods.....	88
5.2.3	Radiography	89

5.2.4	Sonic/Ultrasonic Methods	90
5.2.5	Thermography	91
5.2.6	Vibration-based methods	92
5.2.7	Other methods.....	93
5.3	Impedance-based Method	94
5.4	Proposed Integrated EP and Impedance-based Technique.....	100
5.4.1	Principle.....	100
5.4.2	Measurement System.....	102
5.5	SUMMARY	104
CHAPTER 6 DAMAGE DETECTION IN BEAM		106
6.1	Introduction	106
6.2	Impedance Model	106
6.2.1	Modeling of PZT Transducer	108
6.2.2	Modeling of Host Structure	114
6.2.3	Model Verification	118
6.3	Damage Detection	121
6.4	Summary	127
CHAPTER 7 DAMAGE DETECTION IN PLATE		129
7.1	Introduction	129
7.2	Impedance Model	129
7.2.1	Modeling of PZT Transducer	129
7.2.2	Modeling of Host Structure	136
7.2.3	Model Verification	141
7.3	Damage Detection	145
7.4	Reliability Analysis	149
7.5	Summary.....	151
CHAPTER 8 CONCLUSIONS AND FUTURE WORK		153
8.1	Summary.....	153
8.2	Conclusions	154

8.3 Future Work.....156

PUBLICATIONS.....159

REFERENCES160

APPENDIX.....178

SUMMARY

The elastic parameters and damages in structures can be backcalculated from their mechanical performance. However, due to the large search space and numerous local optima, the traditional local optimization strategies could be trapped in inaccurate or wrong results.

This study develops a general approach based on evolutionary programming (EP) as a backcalculation algorithm for structural nondestructive evaluation (NDE). Although EP has been proven to be robust in finding the global optimum, the speed of convergence could be slow for certain problems. This thesis presents an EP algorithm hybridized with local search strategies. Numerical experiments show that the algorithm improves the search efficiency of the traditional EP method without loss of effectiveness in finding the global optimum. The high search efficiency and effectiveness enhance the feasibility and practicality of applying an EP-based backcalculation algorithm for structural NDE.

The EP-based backcalculation algorithm is employed to determine the elastic parameters of composite plates based on their natural frequencies. The forward problem is dealt with a Ritz method using polynomials as the shape functions. Experimental data from the literature are used for backcalculation. The results are compared with those reported in the literature. The good match between them demonstrates that the proposed method is promising for practical applications in structural parameter identification.

Moreover, in the traditional NDE methods, the impedance-based technique has shown certain advantages. However, it has some limitations in damage assessment, e.g., it can not simultaneously find the location and quantity of the damage. This thesis integrates the EP-based backcalculation approach and the impedance-based method to develop a new technique for damage detection. A generic impedance model for piezoceramic patch (PZT)-structure interacting system is developed and validated by experiments. A special fitness function is proposed to discriminate the modeling errors from the discrepancies caused by damage. The technique is successfully applied to a lab-sized beam and a lab-sized plate. Both the location and the extent of the damage can be simultaneously identified. The reliability of the technique is also investigated. The technique inherits the advantages of the traditional impedance-based method but overcomes its limitations. Although only beams and plates are analyzed in this study, the proposed impedance model and the NDE technique can be applied to other types of structures.

LIST OF FIGURES

Figure No.	Description	Page
Figure 2.1	Structure of evolutionary computation.....	11
Figure 2.2	Structure of continuous EP	19
Figure 3.1	Mechanism of LBS operator.....	37
Figure 3.2	The repairing scheme for crossover.....	40
Figure 3.3	Flowchart for proposed EP-MMLSX algorithm.....	42
Figure 3.4	Effect of simulated annealing on performance of EP-MMLSX for function f_1	46
Figure 3.5	Performances of the 6 algorithms for test function f_1	54
Figure 3.6	Performances of the 6 algorithms for test function f_2	54
Figure 3.7	Performances of the 6 algorithms for test fiinction f_3	55
Figure 3.8	Performances of the 6 algorithms for test function f_4	55
Figure 3.9	Performances of the 6 algorithms for test function f_5	56
Figure 3.10	Performances of the 6 algorithms for test function f_6	56
Figure 3.11	Performances of the 6 algorithms for test function f_7	57
Figure 3.12	Performances of the 6 algorithms for test function f_8	57
Figure 3.13	Performances of the 6 algorithms for test function f_9	58
Figure 3.14	Performances of the 6 algorithms for test function f_{10}	58
Figure 3.15	Performances ofthe 6 algorithms for test function f_{11}	59
Figure 3.16	Performances of the 6 algorithms for test function f_{12}	59
Figure 4.1	A typical lamina.....	67
Figure 4.2	A laminated composite plate.....	69
Figure 4.3	A deformed Mindlin plate.....	70

Figure 4.4	Evolutionary process for SP1	83
Figure 5.1	Details of E/M impedance-base NDE methods	96
Figure 5.2	1-D model used to represent a PZT-driven dynamic structural system	96
Figure 5.3	Typical impedance-based damage detection results (Chardhry et al. 1995)	98
Figure 5.4	Outline of inEPIB technique	102
Figure 5.5	Depiction of measurement system	103
Figure 5.6	Schematic sketch of measurement system	103
Figure 6.1	Liang's model	107
Figure 6.2	Generic one-dimensional PZT-structure interacting system	108
Figure 6.3	Beam actuated by a pair of PZT patches	113
Figure 6.4	Damaged beam model	115
Figure 6.5	Comparison between experimental measurement and numerical prediction of undamaged beam	119
Figure 6.6	Measured E/M admittance for undamaged and damaged beam	122
Figure 6.7	Comparison between experimental measurement and numerical prediction of damaged beam	123
Figure 6.8	A typical admittance peak shift in beam caused by damage	123
Figure 6.9	Shifts of admittance peaks for beam specimen	125
Figure 6.10	Evolutionary process for damage detection in beam	126
Figure 7.1	Zhou's model	130
Figure 7.2	Generic two-dimensional Structure-PZT interacting system	131
Figure 7.3	Plate bonded with a pair of PZT patches	135
Figure 7.4	Undamaged plate specimen	142
Figure 7.5	Experimentally measured and numerically predicted admittance spectra for undamaged plate	143
Figure 7.6	Experimentally measured and numerically predicted admittance spectra for damaged plate	146
Figure 7.7	Experimentally measured and numerically predicted admittance peak shift in plate	147
Figure 7.8	Evolutionary process for damage detection in plate	149

LIST OF TABLES

Table No.	Description	Page
Table 3.1	Comparison between the best results of test function f_1 obtained by EP-MMLSX and SDEP (Soh and Dong 2001).....	47
Table 3.2	Comparison between performance of EP-MMLSX and SAEP. AEP. CEP/BLO for test functions $f_7 \sim f_9$	49
Table 3.3	Comparison between performance of EP-MMLSX and CEP. FEP. CEP/GMO, CEP/CMO, CEP/MMO and CEP/AMMO for test functions $f_9 \sim f_{12}$	50
Table 3.4	Summary of abbreviations of algorithms.....	52
Table 3.5	Unambiguous ranking of the six algorithms.....	60
Table 4.1	Values of Δ in Eqn. (4.20) for different edge boundary conditions.....	76
Table 4.2	Composite plate specimen data.....	81
Table 4.3	Measured natural frequencies.....	82
Table 4.4	Identified elastic parameters after 100 generation.....	84
Table 4.5	Simulated natural frequencies based on the optimized parameters..	85
Table 6.1	Properties of PZT patch	118
Table 6.2	Properties of beam specimen.....	119
Table 6.3	E/M admittance peaks and natural frequencies(KHz)	120
Table 7.1	Properties of PZT patch	141
Table 7.2	Properties of plate specimen.....	141
Table 7.3	E/M admittance peaks and natural frequencies(KHz) for plate	144
Table 7.4	E/M admittance peaks (KHz) for damaged plate.	146
Table 7.5	Identified damage for various ranges of actuating frequency.....	150

LIST OF SYMBOLS

A_d	=	the damaged area in the plate
a	=	the length of the host structure
a_p	=	the length of the PZT patch
b	=	the width of the host structure
b_p	=	the width of the PZT patch
$\mathbf{C}_k$	=	the eigenvector corresponding to the k -th eigenvalue
$C(0,1)$	=	Cauchy random variable with zero mean and unit scale parameter
D_i	=	the electric displacement
d_{kp}	=	the piezoelectric coupling constant
E_d	=	the Young's modulus of the damage element
E_i	=	the Young's modulus in the i -direction
E_k	=	the applied external electric field in the k -direction
F	=	the feasible region in EP search or the force in structural analysis.
$\bar{F}$	=	the magnitude of the force
f_i	=	the fitness value of the i -th individual
$f(\mathbf{x})$	=	the fitness function
G_{ij}	=	the shear modulus in the i - j plane
$H(x)$	=	step function
h	=	the thickness of the host structure

h_p	=	the thickness of the PZT patch
$\mathbf{K}$	=	the stiffness matrix
k_p	=	the wave number
L_d	=	the length of the damaged beam element
$\mathbf{M}$	=	the mass matrix
M	=	the moment
$\overline{M}$	=	the magnitude of the moment
$N(0,1)$	=	Gaussian random number with zero mean and unit variance
n_g	=	the generation index
p_c	=	the probability of crossover operator
p_L	=	the probability of LBS operator
p_M	=	the probability of MM operator
$\mathbf{Q}$	=	the reduced stiffness matrix for composite lamina
$\overline{\mathbf{Q}}$	=	the transformed stiffness matrix for composite lamina
$\mathbf{q}$	=	vector of unknown coefficients
q	=	the tournament size
$\text{Re}(Y)$	=	the real part of Y
R^n	=	n-dimensional space
r_d	=	the radius of holes in the structure
S	=	the search space
S_p	=	the mechanical strain
$\mathbf{s}$	=	vector of mutation step in EP-MMLSX
$\tilde{s}_{pq}^E$	=	the complex mechanical compliance of the material measured at zero electric field
T	=	the kinetic energy functional
T_0	=	the initial temperature in simulated annealing
T_q	=	the mechanical stress
U	=	the strain energy functional
$U(x_j^L, x_j^U)$	=	random number uniformly distributed between x_j^L and x_j^U

u	=	the mid-plane displacement in the x-direction
$\bar{u}$	=	the displacement in the x-direction
v	=	the mid-plane displacement in the y-direction
$\bar{v}$	=	the displacement in the y-direction
W_i	=	the weight of the i-th individual
w	=	the mid-plane displacement in the z-direction
$\bar{w}$	=	the displacement in the z-direction
$\mathbf{x}$	=	vector of objective variables in EP
$\mathbf{x}'$	=	vector of objective variables in the offspring individual
x_c, y_c	=	the coordinates of the center of the damage
$x_j^i[k]$	=	the j-th variable of the i-th individual at the k-th generation
x_j^L	=	the lower bound of the j-th variable
x_j^U	=	the upper bound of the j-th variable
Y	=	the electrical admittance
Y_p^E	=	the real Young's modulus of the PZT patch
$\tilde{Y}_p^E$	=	the complex Young's modulus of the PZT patch at zero elastic field
Z_p	=	the mechanical impedance of the PZT patch
Z_{str}	=	the mechanical impedance of the host structure
$\Gamma_p(\xi, \eta)$	=	the boundary equation of the p-th supporting edge in the $\xi - \eta$ plane
Π	=	the energy functional
Φ_k	=	the k-th mode shape
Ω	=	the measured natural frequency
β	=	the cooling parameter
γ_{ij}	=	the shear strain in the i-j plane
γ_k	=	the participation factor for the k-th vibration mode
δ	=	the dielectric loss factor
$\delta(x)$	=	the dirac function

ε	=	a small positive value
ε_i	=	the normal strain in the i-direction
$\tilde{\varepsilon}_{ik}^T$	=	the complex permittivity at constant stress
ξ	=	non-dimensionalized coordinate corresponding to x
η	=	non-dimensionalized coordinate for plate or the mechanical loss factor for PZT patch
θ	=	the angle between the global xy axes and the principal 1 2 axes of a composite lamina
$\mathcal{G}_p$	=	the exponential in the shape function
κ	=	the shear correction factor
λ	=	eigenvalue
ρ	=	the mass density of material
ρ_p	=	the mass density of PZT patch
σ	=	vector of strategy parameters in traditional EP
σ_i	=	the normal stress in the i-direction
τ_{ij}	=	the shear stress in the i-j plane
ν_{ij}	=	the Poisson's ratio for the transverse strain in the j -direction when stressed in the i-direction
Φ	=	vector of polynomial shape functions
ψ_x	=	the rotation angle in the x - z plane
ψ_y	=	the rotation angle in the y - z plane
ω	=	the angular frequency of vibration

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND

Each member of a structure is designed to sustain certain loads throughout its entire service life. However, faults caused by different fabrication process may exist in the members, which will weaken the overall performance of the structure. So, the properties of the members have to be tested before putting the structure into service to ensure that they can satisfy the designed objectives. Furthermore, continual health monitoring of the members is also necessary during their service life because damage due to fatigue, impact or corrosion may occur in the members.

Although direct measurement of the mechanical properties of a newly constructed member is possible, generally, the tests are costly, time-consuming and labor-intensive. It is even more difficult to test a structure in service. One solution to this problem is to backcalculate the mechanical properties of the structure or members from their performances. Basically, the process of backcalculation can be viewed as a problem of optimization. The properties of the structure are represented by a set of values, which are searched and optimized so that the predicted performances can best match the measured data.

So far, many optimization techniques have been developed; but, these techniques are not suitable for the backcalculation of structural properties. This is due to the large search space and numerous local optima, which could trap the

Chapter 1: Introduction

traditional local optimization techniques in inaccurate or wrong results. In order to obtain comprehensive and reliable information about the structure, a more effective global optimization technique has to be developed and employed for structural parameter identification and damage detection.

Since the 1960s, there has been an increasing interest in imitating living beings (botany, animal, human) to solve difficult optimization problems. In the natural world, evolution of a species is carried out by means of selection and random changes. Simulating this process on a computer results in stochastic optimization techniques that can often outperform classical methods of optimization when applied to difficult real-world problems (Fogel 1994). Such optimization techniques are called, as a total, evolutionary computation (EC). The best-known techniques in the EC methods are Genetic Algorithms (GA) (Holland 1975; Goldberg 1989), Genetic Programming (GP) (Koza 1992), Evolution Strategies (ES) (Rechenberg 1973; Schwefel 1981) and Evolutionary Programming (EP) (Fogel 1962; Fogel 1992). There are also many hybrid systems, which incorporate various features of the four main paradigms.

EC, as a whole, has been proven to be effective and robust in solving many difficult, real-world problems from a variety of domains, such as social systems, machine learning, operational research, ecology, engineering, immunity systems, economics and management. However, each technique has special merits for specific problems. The standard GAs are better at solving adaptive problems; GP is more suitable for symbolic processing and machine learning; ES for numerical optimization; whereas EP was originally for task prediction and has recently been extended to solve continuous parameter optimization problems (Back 1996).

The continuous EP (Fogel 1992) has been successfully applied to handle numerical optimization problems in broad application fields (McDonnell et al 1995; Fogel et al 1996; Ageline et al 1997; Porto et al 1998). It has been demonstrated that EP may be more efficient and effective in search than the traditional trial-and-error

Chapter 1: Introduction

optimization techniques. This is the principal motivation to employ EP as the backcalculation algorithm in this study.

On the other hand, a lot of nondestructive evaluation (NDE) methods have been developed to detect damage in structures. Among these methods, impedance-based damage detection method has shown certain advantages over the traditional NDE methods, as stated by Park et al (2000). However, traditional impedance-based NDE technique works at very high frequency (typically higher than 35KHz). The high frequency vibration constrains the sensing area in a local region. In addition, at such a high frequency, system identification techniques are difficult, if not impossible, to apply because of the necessity to repeat the structural analysis, which is time consuming. Thus, the damage is assessed by a damage index, which is statistically calculated by comparing the amplitudes of the two spectra (damaged vs. pristine). Root mean square deviation (RMSD), mean absolute percentage deviation (MAPD), covariance change (CC), and correlation coefficient deviation (CCD) have been used as the damage indices. However, it is not enough to use these overall-statistics damage indices for damage assessment, i.e., it is not able to locate and quantify the damage simultaneously, since a severe damage far from the measured point and a minor damage close to the point could have the same damage index. Moreover, various values may need to be determined as criteria to predict the failure for different structures.

In view of these limitations of the impedance-based method, this study integrates EP and the impedance-based method to develop a new NDE technique. The integrated EP and impedance-based (inEPIB) technique will have the advantages of the traditional impedance-based method. Moreover, due to the robustness of EP in finding the global optimum, the inEPIB technique is expected to be able to simultaneously identify the location and extent of the damage, which overcomes the limitations of the traditional impedance-based method. This is the second motivation of this study.

1.2 OBJECTIVES AND CONTRIBUTIONS

The main objective of this study is to develop a new general approach based on EP as a backcalculation algorithm for structural nondestructive evaluation. This approach is employed to determine the elastic constants of composite plates from their natural frequencies to investigate the feasibility of the proposed method.

Another objective of this study is to develop a new convenient and reliable method for structural damage detection based on the developed backcalculation algorithm. By integrating EP and impedance-based NDE method, the inEPIB technique is expected to inherit the advantages of the traditional impedance-based method but overcome its limitations.

In this thesis, the elastic parameters of composite plates are identified from their natural frequencies with the EP-based backcalculation algorithm. The inEPIB technique is developed and applied on beams and plates to simultaneously detect the location and quantity of damage. The effectiveness and reliability of the inEPIB technique has been verified by experiments. Specifically, the present work has the following major contributions:

- 1) A hybrid EP algorithm is developed. Numerical experiments on a number of benchmark problems show that the proposed method is able to find the global optimum effectively and efficiently. The high effectiveness and efficiency enhance the feasibility and practicality of applying EP-based backcalculation algorithm for structural nondestructive evaluation.
- 2) A parameter identification technique for composite plates using the proposed hybrid EP is presented. The elastic constants of several composite plate specimens are backcalculated from experimentally measured natural frequencies. Good matches between the obtained results and those available in the literatures demonstrate that the

proposed EP-based backcalculation algorithm is a promising and practical technique for structural parameter identification.

- 3) The inEPIB NDE technique is developed by integrating EP and the impedance-based method. The inEPIB technique inherits the advantages of the traditional impedance-based method, and uses EP to simultaneously locate and quantify the damage. In addition, it works at low frequency so that it can broaden the sensing region of the traditional impedance-based method.
- 4) Feasibility of the inEPIB technique is investigated by applying it to a one-dimensional (1-D) structure. A generic impedance model for a 1-D piezoelectric patch (PZT)-structure interacting system is proposed. The inEPIB technique uses the proposed impedance model to predict the electromechanical (E/M) impedance spectra and EP algorithm to search the damage. A special fitness function is proposed to reduce the inaccuracy in representing the real structure with a numerical model. The process of the proposed damage detection method is illustrated using a beam structure.
- 5) Application of the inEPIB technique is extended to two-dimensional (2-D) structures. A generic 2-D impedance model has been developed to predict the E/M impedance and admittance (inverse of impedance) for general 2-D structures. As an illustration, damage in a plate structure is detected.
- 6) Experiments are carried out on beams and plates to verify the proposed impedance models and the obtained damage detection results. They show that the impedance models can precisely predict the positions of the admittance peaks and the proposed method is able to practically and reliably locate and quantify the damage simultaneously. The

experiments also indicate that the proposed inEPIB technique is a non-invasive method and has the potential for continuous health monitoring.

1.3 SCOPE OF WORK

The proposed method uses EP as a global optimization technique. The EP algorithm is firstly employed to determine the elastic parameters of composite plates to illustrate its feasibility as a backcalculation algorithm for structural NDE. The EP algorithm is then integrated with the impedance-based NDE method to develop an inEPIB technique for structural damage detection. Although the inEPIB technique is only applied on beam and plate specimens in this research, the impedance models are applicable for any 1-D and 2-D structures. Since the EP algorithm is a general optimization technique, the inEPIB technique has the potential to be conveniently applied to other types of structures with suitable structural analysis method.

The scope of this project includes:

- 1) development of a hybrid EP algorithm to improve the search effectiveness and efficiency;
- 2) structural analysis of composite plates to calculate their natural frequencies;
- 3) implementation of the hybrid EP algorithm to backcalculate the elastic constants of composite plates from the measured natural frequencies;
- 4) illustration of the process of the inEPIB technique by applying it to a beam specimen, which includes the development of a generic impedance model for 1-D PZT-structure interacting system to predict its E/M impedance;
- 5) development of a generic 2-D impedance model and application of the inEPIB technique to a plate specimen to simultaneously locate and quantify the damage;

- 6) experimental verification of the impedance models; and
- 7) demonstration of the practicality of the proposed NDE method.

1.4 ORGANIZATION OF THESIS

This thesis consists of eight chapters.

Chapter 1 gives a brief introduction of the background, objectives, contributions and scope of this study.

In the second Chapter, a general description of EC techniques is presented. The history and development of EP technique are briefly reviewed. A review of the application of EC in civil engineering and other areas is also included in this chapter.

Chapter 3 presents a hybrid EP algorithm by combining a global mutation operator, a local bisection search operator and a crossover operator. Numerical experiments are carried out on a number of benchmark problems. The effect of different operators in the evolutionary computation process is also investigated.

In Chapter 4, the hybrid EP algorithm is employed to backcalculate the elastic constants of composite plates based on their measured natural frequencies. A p-version Ritz method is employed to predict the natural frequencies of the composite plates. The backcalculated results are compared with those available in the literature.

The concept of the proposed inEPIB damage detection technique is introduced in Chapter 5. A comparison of the proposed technique with the other NDE methods is presented. The measurement system used in the inEPIB technique is also described.

Chapter 1 : Introduction

Chapter 6 illustrates the process of the proposed inEPIB technique by applying it to a 1-D structure. A generic impedance model for 1-D PZT-structure interactive system is first developed. By assuming that the PZT active sensor is bonded at its boundaries to the structure and investigating the vibration of the PZT active sensor and the structure, the E/M impedance is predicted. The hybrid EP algorithm is then employed to find the location and quantity of the damage. The impedance model and the damage detection results are verified by the experiments carried out on a lab-sized beam,

Chapter 7 extends the application of the inEPIB technique to 2-D structures. A generic 2-D impedance model is developed. A lab-sized plate is selected to represent the 2-D structure. Experiments show that the inEPIB technique is able to simultaneously find the location and quantity of the damage. The reliability of the proposed method is also investigated.

Finally, Chapter 8 summarizes the work and discusses the conclusions and the areas for future work.

CHAPTER 2 EVOLUTIONARY COMPUTATION

2.1 EVOLUTIONARY COMPUTATION METHODS

2.1.1 Emergence

The ability of human beings to understand the real world is very much limited. In exploring the real world, we are constantly looking for rules that control the appearance and the behavior of the material world. Yet it proves to be too difficult to completely understand the complicated law of the existing world, to say less of mastering it. The mathematical and physical equations which we believe can precisely describe the objective law of the motion of materials are in fact too complicated to be used in dealing with real engineering problems. Realizing this, engineers and scientists are satisfied with getting the "best solution" among the available solutions. Optimization is developed under such a circumstance for various engineering applications, and has ever since been a hot topic of research.

More than 30 years ago, a number of innovative researchers at different places in the United States and Europe independently came up with the idea of mimicking the mechanism of biological evolution in order to develop powerful algorithms for solving problems of adaptation and optimization. Since many optimal structures like the shape of birds' wings or the branching of blood vessels are believed to have emerged through biological evolution, the idea to utilize the underlying mechanism for the solution of optimization problems has motivated a

Chapter 2: Evolutionary Computation

considerable amount of research, resulting in several approaches that have proven their effectiveness and robustness in many difficult, real-world problems from a variety of domains: social systems, machine learning, operational research, ecology, engineering, immunity systems, economics, and management.

During the last three decades, there has been a growing interest in algorithms which are based on the so-called “principle of evolution”. A common term, accepted recently, refers to such techniques as evolutionary computation (EC) methods. There are four mainstream methods in the field of EC. They are Genetic Algorithms (GA), developed by Holland (1975) and received broad extension contributed by Goldberg (1989), Genetic Programming (GP), developed by Koza (1992) and sometimes considered as an extension of GA, Evolutionary Strategies (ES), developed by Rechenberg (1973) and Schwefel (1981), and Evolutionary Programming (EP), initiated by Fogel (1962) and brought to the modern type by Fogel (1992). There are also many hybrid systems which incorporate various features of the above paradigms and consequently are hard to classify but, they can all be generally referred to as EC methods.

2.1.2 Procedure

All of the EC methods share a common conceptual base of simulating the evolution of individuals via processes of selection and variation. These processes depend on the perceived performance (fitness) of the individuals (representing potential solutions to a given optimization problem) as defined in an environment. Thus, a general EC algorithm must have the following five basic components for a particular problem (Michalewicz 1996):

- a genetic representation for potential solutions to the problem, which includes binary encoding in classical GA, parse-tree encoding in GP, and real-value representation in ES and EP;

Chapter 2: Evolutionary Computation

- a way to create an initial population of potential solutions;
- an evaluation function that plays the role of the environment, rating solutions in terms of their “fitness”;
- genetic operators that alter the genetic composition of the children during reproduction, which include mutation operator that introduces innovation into the population by generating variations of individuals, and crossover operator that typically performs an information exchange between different individuals from a population; and
- values for various parameters that the EC algorithm uses, including population size, probabilities of applying genetic operators, etc.

Besides, a selection method or selection operator, which imposes a driving force on the process of evolution by retaining better individuals to survive and reproduce, is also to be supplied by the EC algorithm.

A typical EC algorithm is outlined in Figure 2.1.

```

begin
   $t := 0$ 
  initialize  $P(t)$ 
  evaluate  $P(t)$ 
  while (not termination-condition) do
    begin
       $t := t + 1$ 
      select  $P(t)$  from  $P(t-1)$ 
      alter  $P(t)$ 
      evaluate  $P(t)$ 
    end
  end

```

Fig. 2.1 Structure of evolutionary computation

An evolutionary algorithm typically first initializes a random population of possible solutions, although domain-specific knowledge can be used to bias the

Chapter 2: Evolutionary Computation

search. Then, the individuals are altered according to different operators such as crossover and mutation. Each individual in the population receives a measure of its fitness in the environment. Evaluation may be as simple as computing a fitness function or as complex as running an elaborate simulation. Selection focuses attention on high performance individuals, thus exploiting the available fitness information. Crossover and mutation perturb those individuals, providing general heuristics for exploration. The main loop of the algorithm, including alteration, evaluation and selection, is iterated until a pre-described termination condition is satisfied. Darwinian principle of natural selection, i.e., survival-of-the-fittest, is the driving force of the evolution of individuals. Such a procedure, although simplistic from a biologist's viewpoint, is sufficiently complex to provide robust and powerful adaptive search mechanisms.

2.1.3 Advantages and Disadvantages

The above description shows that EC techniques have some fundamental differences as compared with the traditional optimization methods, such as:

- EC works with the coding of a solution set and not the solutions themselves;
- EC search is based on a population of possible solutions;
- EC uses fitness function (or payoff information), not derivatives or other auxiliary knowledge; and
- EC uses probabilistic transition rules instead of deterministic rules.

Due to these differences, EC has some inherent advantages over the traditional optimization methods, such as conceptual simplicity, broad applicability, outperforming classic methods on real problems, potential to use knowledge and hybridize with other methods, parallelism, robust to dynamic changes, capability for self-optimization, and ability to solve problems that have no known solutions (Fogel 1999). The advantages are summarized below:

- EC techniques are very effective at performing global search (in probability). The traditional approaches perform local search by a convergent stepwise procedure, which compares the values of nearby points and moves to the relative optimal points. In the traditional approaches, global optimum can be found only if the problem possesses certain convexity properties that essentially guarantee that any local optimum is a global optimum (Gen and Cheng 1997).
- EC techniques are widely applicable. EC can be applied to solve those problems with rough, discontinuous, non-differentiable, multi-modal, noisy, and even fractal landscape, which can not be handled by the traditional methods, especially those that need global prediction from local surface analysis. They are applicable even in dynamic situations where the goal or constraints are moving over time or changing, either exogenously or self-induced, and where parameter adjustments and fitness measurements are disturbed. Thus their effectiveness (or robustness) extends to a broader field of applications (Lozano 2002, Back et al 1997).
- The cost to adapt EC to a new optimization problem is relatively low. EC presents a methodological framework that is easy to understand and handle, and is either usable as a black-box method or open to the incorporation of new or old recipes for further sophistication, specialization or hybridization. The same procedures can be applied to diverse problems with relatively little re-programming, so that domain-dependent heuristics can be easily hybridized to make it efficient in the implementation of a specific problem (Lozano 2002, Back et al 1997).
- EC is a parallel algorithm. The greatest potential for the application of EC to difficult real-world problems will come from their implementation on parallel machines, for evolution is an inherently parallel process. Each parent can be varied independently, and each offspring can be evaluated independently. Competition and selection can also be parallelized. Recent

advances in distributed and parallel processing architectures will result in dramatically reduced execution times for simulations that would simply be impractical on current serial computers (Fogel 2000a).

On the other hand, EC have some disadvantages. For instance, there is no guarantee of finding the global optimal solutions and there are no reliable stopping criteria. There is also not enough theoretical basis. Often, these methods are computationally expensive. It is very difficult to carry out a comparison between different algorithms unless experimentally. It is not possible to know how far the solution obtained by the algorithm is from the global optimum. Finally, and probably the worst characteristic, is the fact that the algorithm depends on a set of parameters that have to be tuned experimentally for the problem at hand, with this tuning itself being a difficult optimization problem for some cases. In ES, in particular, this tuning is done using a self-adaptation technique. Nevertheless, the exponential growth of the field is due to the good results obtained for practical real world problems (Lozano, 2002).

2.1.4 Similarities and Differences

Although the processes of all EC techniques are similar at the highest level, they have some subtle differences. The differences touch upon almost all aspects of EC algorithms, including the choices of representation for the individuals, types of selection mechanism used, forms of genetic operators, and measures of performance. Some of the basic differences characterize the EC algorithm into four mainstreams, i.e., GA, GP, ES and EP.

Hoffmeister and Bäck (1991), Back et al (1993), Bäck (1996) and Fogel (2000a) made comparisons between the mainstreams. The similarities and differences for these instances are briefly summarized as follows:

Chapter 2: Evolutionary Computation

- GA and GP simulate the natural evolution mainly on the level of genotype, and thus, they emphasize recombination (crossover) as the most important search operator and apply mutation with very small probability solely as a “background operator”. They also use a probabilistic selection operator (proportional selection). In GA, the potential solutions are generally encoded as binary strings although real-valued representation is also accepted recently, while GP uses parse-trees.
- ES uses normally distributed mutations to modify real-valued vectors and emphasizes mutation and recombination as essential operators for searching in the search space and in the strategy parameter space at the same time. The selection operator is deterministic, and parent and offspring population sizes usually differ from each other.
- EP emphasizes mutation and does not incorporate the recombination of individuals. Similar to ES, when approaching real-valued optimization problems, EP also works with normally distributed mutations and extends the evolutionary process to the strategy parameters. However, the selection operator is probabilistic (tournament selection), and the population size is generally kept as a constant.

In practice, which technique should be chosen for a special problem depends on the characteristics of the problem. Bäck (1996) recommended that standard GA are better at solving adaptive problems; GP is more suitable for symbolic processing and machine learning; ES for numerical optimization; whereas EP was originally for task prediction and has recently been extended to solve continuous parameter optimization problems. Michalewicz et al (1997) also suggested ES and GA with real-valued encoding for numerical optimization, and other versions of GA to handle combinatorial optimization problems. They also believed that “GP are great in discovery of rules given as a computer program, and EP techniques can be used successfully to model a behavior of the system”.

Thus, considering the characteristics of backcalculation problem, this study selects EP as the global optimization algorithm. However, as pointed out by Michalewicz et al (1997), none of the evolutionary techniques is perfect (or even robust) across the problem spectrum; only the whole family of algorithms based on evolutionary computation concepts has this property of robustness. The main key to successful applications is in heuristics methods, which are mixed skillfully with evolutionary techniques.

2.2 EVOLUTIONARY PROGRAMMING

Evolutionary programming (EP), originally conceived by Lawrence J. Fogel in the 1960's, is a stochastic optimization strategy similar to GA. However, EP differs philosophically from other EC techniques such as GA in a crucial manner. EP is a top-down versus GA's bottom-up approach to optimization. It is important to note that (according to neo-Darwinism) selection operates only on the phenotypic expressions of a genotype; the underlying coding of the phenotype is only affected indirectly. The realization that a sum of optimal parts sometimes does not lead to an optimal overall solution is key to this philosophical difference. GA relies on the identification, combination, and survival of 'good' building blocks (schemata) iteratively combining to form larger 'better' building blocks. In GA, the coding structure (genotype) is of primary importance as it contains the set of optimal building blocks discovered through successive iterations. The building block hypothesis (Michalewicz et al 1997) is an implicit assumption that the fitness is a separable function of the parts of the genome. GA's successively iterated local optimization process is different from EP, which is an entirely global approach to optimization. Solutions (or organisms) in an EP algorithm are judged solely on their fitness with respect to the given environment. No attempt is made to partition credit to individual components of the solutions. In EP (and in ES), the variation operator allows for simultaneous modification of all variables at the same time. Fitness, described in terms of the behavior of each population member, is evaluated directly, and is the sole basis for survival of an individual in the population. Thus, a

crossover operation designed to recombine building blocks is not utilized in the general forms of EP (Porto 1997).

2.2.1 History

The book, entitled “Artificial Intelligence Through Simulated Evolution” (Fogel et al 1966), is viewed as the landmark publication for EP applications, although several other papers had appeared earlier in the literature. EP was developed as a method for the simulation of intelligent behavior by computers. As defined by Fogel (2000a), intelligent behaviors are “the capability of a system to adapt its behavior to meet its goals in a range of environments”. For the sake of generality, the environment is described by a sequence of symbols (e.g., $a_1, a_2, \dots, a_n$). The evolving algorithm is supposed to produce, as an output, a new symbol (e.g., a_{n+1}) based on the previous known sequence of symbols. The output symbol should maximize the payoff function, which measures the accuracy of the prediction.

This goal was initially attempted by the use of finite state machines (FSMs). As described by Fogel (2000a), FSMs provide a meaningful representation of behavior based on interpretation of symbols. An FSM is a transducer that can be stimulated by a finite alphabet of input symbols. It can respond in a finite alphabet of output symbols and possesses a finite number of different internal states. The behavior of an FSM is specified by the corresponding input-output symbol pairs and next-state transitions for each input symbol.

EP operates on FSMs in the following way. A population of “parent” FSMs is exposed to the environment, that is, the sequence of symbols thus far observed. For each parent machine, as each input symbol is offered to the machine, each output symbol is compared with the next input symbol. The worth of this prediction is then evaluated with respect to the given payoff function (e.g., all-none, absolute

Chapter 2: Evolutionary Computation

error). After the last prediction is made, a function of the payoff for each symbol (e.g., average payoff per symbol) indicates the fitness of the machine.

Offspring machines are created by randomly mutating the state diagram of each parent machine. For convenience, each parent typically produces one offspring. The mutation can be executed in five different ways: change an output symbol, change a state transition, add a state, delete a state, or change the initial state. The deletion of a state and the changing of the initial state are allowed only when the parent machine has more than one state. Mutations are chosen with respect to a probability distribution, which is typically uniform. The number of mutations per offspring is also chosen with respect to a probability distribution (e.g., Poisson) or it may be a fixed priori. These offspring are then evaluated over the existing environment in the same manner as their parents.

Those machines that provide the greatest payoff are retained to become parents of the next generation. Typically, half of the total machines are saved so that the parent population remains at the same size. This process is iterated until it is required to make an actual prediction of the next symbol (as yet unknown) in the environment. The “best” machine is chosen to generate this prediction, the new symbol is then added to the experienced environment, and the process is repeated.

The original EP procedure was successfully applied to problems in prediction, identification, and automatic control (Fogel et al 1964, 1966; Fogel 1968), and was subsequently extended to simulate co-evolving populations by Fogel and Burgin (1969).

In the mid-1980s, the general EP procedure was further extended to alternative representations including ordered lists for the traveling salesman problem (TSP) (Fogel and Fogel 1986; Fogel 1990, 1993), and to real-valued vectors for continuous function optimization (Fogel and Fogel 1986). This led to other applications in route planning (Fogel and Fogel 1988), optimal subset

selection (Fogel 1989), training neural networks (Fogel and Atmar 1990) and classifications such as game playing (Chellapilla and Fogel 1999; Fogel 2000b).

2.2.2 Development

Modem EP was proposed by Fogel (1992), and worked mainly in continuous domains. The application is very similar to that of ES, and the basic body of the algorithm resembles that of ES too, as depicted in Fig. 2.2. A population of possible solutions is generated randomly to form the first generation. All of the individuals are evaluated and mutated to create new individuals. Generally no crossover operator is employed in the chromosome modification. Each of the offspring individuals is assessed by computing its fitness. Tournament selection (Fogel 1988) is performed to determine the survivals: each individual competes with q randomly selected opponents to get a “win”, and those individuals with higher win will be retained for the next generation. The whole procedure of mutation-evaluation-selection iterates until the termination condition is satisfied.

```

begin
   $t := 0$ 
  initialize  $P(t)$ 
  evaluate  $P(t)$ 
  while (not termination-condition) do
    begin
      mutate  $P(t)$  forming  $P'(t)$ 
      evaluate  $P'(t)$ 
      select  $P(t+1)$  from  $P'(t)$  and  $P(t)$ 
       $t := t + 1$ 
    end
  end

```

Fig. 2.2 Structure of continuous EP

In the early versions of EP applied to continuous parameter optimization (Fogel 1992), the mutation operator is Gaussian with a zero mean and a variance obtained for each component of the object variable vector as the square root of a linear transform of the fitness value $f(x)$:

$$x'_i = x_i + \sqrt{\beta_i(t) \cdot f(x_i + \gamma_i)} + N_i(0,1) \quad (2.1)$$

where x_i and x'_i are the i -th object variable before and after mutation; β_i is the proportionality constant, and γ_i is an offset parameter. Both β_i and γ_i have to be set externally for each problem. $N_i(0,1)$ is the i -th independent sample from a Gaussian distribution with zero mean and unit variance.

In this early version of continuous EP, there is a set of parameters associated with mutation, i.e. β_i and γ_i , to be tuned and determined by the users. Fogel et al (1991) proposed a meta-EP, which tried to minimize the involvement of a human operator. The mutation operation was designed to perturb each component of the parent vector by adding a Gaussian random variable with zero mean and a variance that was an affine function of the parent's error score, i.e., the fitness of the parent. Useful settings for the constant of proportionality and the offset value were determined through experimentation. However, this meta-EP has been applied to only two function optimization problems with minima of zero.

Self-adaptive EP (SAEP) (Fogel 1992, 2000a) successfully overcame the problem of user-determine-parameters. Global convergence may be obtained even in the presence of suboptimal parameterization. In SAEP, each individual $a = (\mathbf{x}, \boldsymbol{\sigma})$ contains a vector of the object variables $\mathbf{x}$ and a vector of the strategy parameters $\boldsymbol{\sigma}$, which control the mutation step. One offspring $a'_i = (\mathbf{x}'_i, \boldsymbol{\sigma}'_i)$ is created from its parent $a_i = (\mathbf{x}_i, \boldsymbol{\sigma}_i)$ as follows:

$$x'_i(j) = x_i(j) + \sigma_i(j) \cdot N_j(0,1) \quad (2.2)$$

$$\sigma'_i(j) = \sigma_i(j) \cdot \exp(\tau \cdot N(0,1) + \tau' \cdot N_j(0,1)) \quad (2.3)$$

where $x_i(j)$, $x'_i(j)$, $\sigma_i(j)$ and $\sigma'_i(j)$ denote the j -th Component of the vectors $\mathbf{x}_i$, $\mathbf{x}'_i$, $\boldsymbol{\sigma}_i$ and $\boldsymbol{\sigma}'_i$, respectively; $N(0,1)$ is a standard normal random variable sampled once for all parameters of the vector $\boldsymbol{\sigma}_i$; $N_j(0,1)$ indicates that the random number is generated anew for each value of j ; the factors τ and τ' are commonly set to

$(\sqrt{2n})^{-1}$ and $(\sqrt{2\sqrt{n}})^{-1}$, respectively; and n is the number of variables (Fogel 1994,

Back and Schwefel 1993). The initial value for the strategy parameters is generally set as 3.0 (Fogel 1991, Back and Schwefel 1993).

Wei et al (1996) tried to improve EP algorithm by using a different selection operator. In their algorithm, each individual in the initialized population was treated as a parent of a family. Each parent generated a set of new solutions (say M), which formed the children of each family. N parents and $M*N$ children competed by using Boltzman machine-like mechanism. The best solution in each family was selected as the parent of the next generation and the better the family, the larger the acceptance number of members of that family. This acceptance number determined the number of children allocated to that family for the next generation. The iteration process was the same as that in the simulated annealing procedures. The authors proved that this process gave a regional guidance to the stochastic search process of EP.

Liang et al (1998a) reported that premature convergence is the major disadvantage of self-adaptive EP due to the rapid drop in the values of the strategy parameters as compared to the corresponding object variables to very low values. They suggested the use of a fixed lower bound on each of the strategy parameters to improve the overall performance of these algorithms. Subsequently, Liang et al (1998b) proposed a dynamic lower bound on the strategy parameters that resulted in performance improvement on some test functions. Swain and Morris (2002) also proposed a differential step lower bound for the strategy parameters, which depended on the distance between the j -th object variable of the i -th individual and the corresponding object variable of the fittest individual in the population pool.

As a primary performance operator in EP, mutation has received much attention to improve the search efficiency. In the early versions of continuous EP and SAEP, the mutation is based on the Gaussian distribution. However, Yao and Liu (1996) and Yao et al (1999) proposed a fast EP (FEP) by replacing the Gaussian distribution in Eqn. (2.2) with a Cauchy distribution:

$$x'_i(j) = x_i(j) + \sigma_i(j) \cdot C_j(0,1) \quad (2.4)$$

where $C_j(0,1)$ is a Cauchy random variable with the scale parameter $t = 1$ and is generated anew for each value of j .

Compared with the Gaussian mutation, the Cauchy mutation is more likely to generate an offspring further away from its parent, and thus FEP has a higher probability of escaping from a local optimum or moving away from a plateau, especially when the “basin of attraction” of the local optimum or the plateau is large relative to the mean step size. Experiments (Yao et al 1999) have shown that FEP performs much better than SAEP for multimodal functions with many local minima while being comparable to SAEP in performance for unimodal and multimodal functions with only a few local minima. However, the Cauchy mutation spends less time in exploiting the local neighborhood of the parent and thus has a weaker fine-tuning ability than the Gaussian mutation in small to mid-range regions. Yao et al (1999) suggested that when the parent was near to the small neighborhood of the global optimum, a smaller scale parameter t for the Cauchy mutation should be selected so that it would have a smaller search step size so as to improve FEP's performance. Rudolph (1997) also made a theoretical analysis on the local convergence rate of evolutionary algorithm with the Cauchy mutation.

Iwamatsu (2002) generalized EP with the Gaussian and Cauchy mutation by replacing the Gaussian random number in Eqn. (2.2) with the general Levy-type random number, which followed the Levy-type distribution characterized by a real number q ($1 \leq q \leq 3$):

$$p_q(x) = \sqrt{\frac{q-1}{\pi\sigma^2}} \frac{\Gamma\left(\frac{1}{q-1}\right) / \Gamma\left(\frac{1}{q-1} - \frac{1}{2}\right)}{\left(1 + (q-1)(x/\sigma)^2\right)^{\frac{1}{q-1}}} \quad (2.5)$$

This distribution function became the Gaussian distribution in the limit $q \rightarrow 1^+$, and became the Cauchy distribution when $q = 2$ and Student's t-distribution when q is rational numbers $q = (3+m)/(1+m)$ ($0 < m < \infty$). Thus, the EP algorithm with Levy-type mutation included both SAEP and FEP, and can realize a more flexible search using the fat-tailed distribution.

Yao et al (1999) also investigated the probability of mixing the Cauchy mutation and the Gaussian mutation. They thought “it would be ideal if the Cauchy mutation is used when search points are far away from the global optimum and the Gaussian mutation is adopted when search points are in the neighborhood of the global optimum”. To realize this idea, they proposed an improved fast EP (IFEP). IFEP generated two candidate offspring from each parent, one by the Cauchy mutation and the other one by the Gaussian mutation. The better one was then chosen as the surviving offspring. It was found that IFEP was capable of performing as well as or better than the better one of SAEP and FEP for most test functions.

In view of the fact that the Cauchy mutations might be more useful in escaping local optima while the Gaussian mutations provide relatively faster local convergence on convex functions, a lot of work has been done to mix the two operators and to exploit their desired properties. Chellapilla and Fogel (1997) and Chellapilla (1998) combined the two operators linearly to develop two new mutation operators, namely, mean mutation operator and adaptive mean mutation operator (MMO and AMMO, respectively). With the MMO, the object variable equation Eqn. (2.2) was replaced by:

$$x'_i(j) = x_i(j) + 0.5\sigma'_{1i}(j) \cdot (N_j(0,1) + C_j(0,1)) \quad (2.6)$$

The MMO was then extended to the AMMO by introducing self-adaptation scheme. The new object variable was generated by:

$$x'_i(j) = x_i(j) + \sigma'_{1i}(j)N_j(0,1) + \sigma'_{2i}(j)C_j(0,1) \quad (2.7)$$

With the variation of $\sigma'_{1i}(j)$ and $\sigma'_{2i}(j)$, the mutation step could be adjusted between the Gaussian and Cauchy mutations. However, the AMMO contained twice as many strategy parameters as the other operators, thus the time taken to find good strategy parameters through self-adaptation was expected to increase.

Yao and Liu (1998) also presented a mixed EP (MEP) to combine the Gaussian mutation and the Cauchy mutation. In MEP, the offspring was generated by the Cauchy mutation or the Gaussian mutation based on a probability determined by the user. They also found that the advantage of FEP over the conventional EP for the low-dimensional problem (when the number of variables $n < 30$) diminished

Chapter 2: Evolutionary Computation

gradually as the dimensionality grew from 100 to 200, especially for the multimodal functions, while “both IFEP and MEP have displayed excellent robustness across dimensionality in their good performance”.

Kim et al(1996) developed an accelerated EP algorithm (AEP), which made full use of the information accumulated during the search process. Each individual consisted of the objective variables, the evolution directions and an “age”, which denoted the duration of the life as an integer type. If the current solution was better than that of the previous generation, then the evolutionary direction is kept for the next generation and a random perturbation is made to generate a new generation. Otherwise the evolutionary direction is kept, with a probability. The “age” parameters controlled the standard deviation (step size) of the Gaussian-distributed random numbers. If an individual fell into a local optimum, its “age” would increase. As a result, the step size was also increased and the individual searched a broader area and got more change to escape from the local optimum. Zhang and Xu (1998) further extended this algorithm for constrained optimization. In their algorithm, if an infeasible solution was generated, it was replaced with the best solution discovered up to the present generation.

Many engineering optimization problems frequently encounter discrete variables as well as continuous variables. Cao et al (2000) developed a mixed-variable evolutionary programming (MVEP) for engineering optimization problems, which contain integer, discrete, zero-one and continuous variables. For the continuous variables, the Gaussian mutation was employed. For the zero-one variables, mutation was conducted by uniformly changing a variable from zero to one and vice versa. For the integer variables, mutation was achieved by changing a variable from an integer value to another with a random step size. And for the discrete variables, mutation was done by changing a variable from a pre-specified discrete value to another.

There are also many other work that has been done to improve the performance of EP on certain objective functions. Choi and Oh (2000) offered a

mutation rule inspired by the neural network backpropagation learning. The offspring was generated by perturbing the parent according to following equations:

$$x_j^i[k+1] = x_j^i[k] + \eta \cdot \Delta x_j^i[k] + \alpha \cdot sx_j^i[k] \quad (2.8)$$

$$\Delta x_j^i[k] = (x_j^{best}[k] - x_j^i[k]) \cdot |N(0,1)| \quad (2.9)$$

$$sx_j^i[k+1] = \eta \cdot acc^i[k] + a \cdot sx_j^i[k] \quad (2.10)$$

where $x_j^i[k]$ is the j -th variable of the i -th individual at the k -th generation; η and a are real-valued constants to be determined empirically; $|\bullet|$ denotes an absolute value; $\Delta x_j^i[k]$ is the amount of change in an individual, which is proportional to the temporal error, and it drives the individual to evolve close to the best individual at the next generation; $sx_j^i[k]$ is the evolution tendency or momentum of previous evolution; and $acc^i[k]$ is defined as 1 if the current update has improved cost or zero for other cases. The momentum term served to accumulate evolved information. The efficiency of the EP algorithm was assessed using several benchmark test functions. The algorithm not only found better solutions in the chosen search space by guiding the evolution direction of individuals, but also speeded up the convergence to the global optimum by using the momentum factor.

Shi and Xu (2001) presented another mutation strategy. It used the principle of Gaussian mutation, however, the strategy parameter was modified by:

$$\sigma_{(k,i)j} = a_{k,j} \cdot \exp(b_{k,j} / f_{k,i}) \quad (2.11)$$

where subscripts i , j , and k indicate the i -th individual, the j -th component and the k -th generation, respectively; $f_{k,i}$ is the fitness score of the i -th individual of the k -th generation; and $a_{k,j}$ and $b_{k,j}$ are the scaling factors. Adaptation of the scaling factors during the process of evolution was also formulated:

$$a_{k,j} = \frac{Y}{(P+k) \left| \sum_{m=1}^{12} X_m - 6 \right|} \quad (2.12)$$

Chapter 2: Evolutionary Computation

$$b_{k,j} = f_k^{\min} \ln \frac{x_{(k,i)j}^{\max} - x_{(k,i)j}^{\min}}{2a_{k,j} \left| \sum_{m=1}^{12} Z_m - 6 \right|} \quad (2.13)$$

where X , Y and Z are random numbers uniformly distributed in $[0,1]$; the superscript max and min represent the maximum and minimum of the corresponding variable; and P denotes the population size. They reported that their algorithm expressed strong self-adaptability, versatility and global optimization capability of escaping local optimum.

Choi (2002) presented a new mutation operator which tried to accommodate information from the other individuals, even from the whole population. In this method, each individual was represented by its objective variables only. The mutation step size of an individual was not decided only on its own fitness, but automatically determined by the contribution of the other individuals. For an individual, the rest of the individuals were grouped into two by their relative fitness. One group consisted of individuals with better fitness and the other with poorer fitness. For an individual $\mathbf{x}_i$, the j -th components of the virtual normalized better-fit individual and poorer-fit individual were represented as:

$$G_{ij} = \frac{\sum_{t=1}^{M_1} x_{tj}}{M_1}, \forall \mathbf{x}_i \text{ such that } f(\mathbf{x}_i) < f(\mathbf{x}_i) \quad (2.14)$$

and

$$B_{ij} = \frac{\sum_{t=1}^{M_2} x_{tj}}{M_2}, \forall \mathbf{x}_i \text{ such that } f(\mathbf{x}_i) > f(\mathbf{x}_i) \quad (2.15)$$

where M_1 and M_2 represent the number of better- and poorer-fit individuals as compared with $\mathbf{x}_i$, respectively. Using the virtual individuals, $\mathbf{x}_i$ was mutated and a new offspring was generated as follows:

$$\mathbf{x}'_i = \mathbf{x}_i + \beta_1[(\mathbf{x}_i - \mathbf{G}_i)N(0,1)] - \beta_2[(\mathbf{x}_i - \mathbf{B}_i)N(0,1)] \quad (2.16)$$

where β_1 and β_2 are empirically determined parameters, and $\mathbf{G}_i$ and $\mathbf{B}_i$ are the virtual individuals. The step size was controlled by the virtual individuals and adaptively changed for individual as the environment changes.

2.3 EC APPLICATIONS IN ENGINEERING

Due to the robustness of finding the global optimum, EC techniques have received extensive applications in engineering areas.

Xie and Steven (1993) developed an evolutionary structural optimization technique (ESO). The concept of ESO states: “by slowly removing inefficient material from a structure, the shape of the structure evolves towards an optimum” (Xie and Steven 1997). ESO is an iterative process where a small amount of material with minimal effects on the gross structural integrity is removed at every iteration, reducing the total volume. The discrete design variables of finite elements are employed to modify the structure. The technique was further extended for topology and shape design with constant volume (Steven et al 2000). The simplicity of the ESO formulation has led to its applications in a wide range of problems, including incorporating displacement and stiffness objectives, natural frequency and buckling optimization and configuration optimization (Kim et al 2002).

Minster et al (1995) employed EP technique to determine the hypocenter of earthquake. In their method, simulated annealing was adopted to prevent premature convergence to a local optimum. They reported that EP was apparently a more effective technique than GA for this problem, yielding faster convergence, particularly when coupled with a simulated annealing schedule.

Weber (1998) proposed a circular binary string to represent a molecule. The molecule composed of certain building blocks of atoms and shared electrons. The possible building blocks were contained in a big library with 20 billion components. GA was thus used to build a specific target molecule by selecting suitable building blocks from the library and combining them in appropriate order. The astonishing result was that GA examined only 3600 peptoids out of the 20 billion and assembled them to the expected molecule. It was found that GA was well suited for this combinatorial optimization problem, and a properly set up GA would converge relatively quickly in 10-15 generations.

Chapter 2: Evolutionary Computation

Gomme and Harrald (1998) applied EP to several instances of the set partitioning problem. For this special problem, the possible solution was encoded to a vector of integers. Mutation was done by randomly choosing some elements in the chromosome and reassigning it to a randomly chosen subset. The element was not deleted if it would produce an invalid solution. Individuals with better fitness were deterministically selected to retain for the next generation.

Karr et al (2000) used the floating-point GA (Michalewicz, 1996) to solve the inverse initial-value and boundary-value (IVBV) problems. The inverse IVBV problem was transformed into a nonlinear optimisation problem characterized by local optima. GA was employed as a search engine and efficiently located the global optimum for three specific examples. The results indicated that GA was viable and effective for solving a wide variety of inverse IVBV problems.

By incorporating with domain-specific knowledge, GA has even found application in fashion design. Kim and Cho (2000) proposed an interactive GA (IGA) to choose suitable clothes according to the consumer's favorites. Different from the conventional GA in which the individual is evaluated by calculating a fitness function, IGA assessed the individuals by users, such that there was an interaction between the users and the computer, and users could choose different style of clothes according to their own preference. In their experiments, IGA has provided user-satisfied results. However, the size of the search space was relatively small (1,880,064 in their experiments), and improvements should be made to decrease the users' burden when the search space is large.

In manufacturing operation, GA was employed as an optimization method to determine the process parameter values (Cook et al 2000). The process parameters were used to represent the genes within the population of chromosomes. A trained neural network model was used as the fitness function for evaluating individual chromosomes. The tool integrated neural network and GA (NN-GA) has successfully determined the process parameter values which could result in the desired product characteristics.

Burczynski et al (2000) investigated the application of GA in engineering mechanics. A floating-point GA was employed to identify voids and cracks in structures. The satisfactory results obtained show that evolutionary techniques were very effective and promising. However, they also pointed out that one disadvantage of the application of GA in these mechanical problems was the time consuming calculation.

Soh and Yang (2000) introduced GP into civil engineering problem solving. With an encoding strategy to map between the real structures and the GP parse-trees, a GP-based approach incorporated with FEM for structural analysis was employed for simultaneous sizing, geometry, and topology optimization of structures. Numerical examples on structural optimization revealed that GP was capable of producing the topology and shape of the desired trusses and the sizing of all the structural components. They also incorporated a fuzzy logic controller into their GP algorithm to improve its search efficiency (Yang and Soh 2000).

Huang et al (2001) integrated GA with a neural-fuzzy technique to predict the permeability in petroleum reservoirs. GA was used to optimize the defuzzification operator parameters. A binary string with fixed length of 96 bits was used to represent the parameters. Case study in a petroleum reservoir in offshore Western Australia showed that the integrated neural-fuzzy-genetic-algorithm improved the predictions by 3-31 % when compared with other similar algorithms. However, it was also pointed out that the cost for improving the performance was the CPU time for the numerical optimization.

Iwamatsu (2001) used EP to study the structural optimization of atomic clusters. The lowest energy structure of silicon clusters has been successfully found. He claimed that the self-adaptation of mutation used in EP was more effective than the traditional scheduled annealing used in simulated annealing method. Moreover, he believed that EP was better than simulated annealing and GA for this parameter optimization problem because it “can avoid both the cumbersome planning of a cooling schedule as in SA and the relaxation of clusters by local optimization as in

Chapter 2: Evolutionary Computation

GA”, and suggested “EP is such a robust algorithm that many optimization problems in physics and chemistry can be handled effectively.”

Soh and Dong (2001) investigated the possibility of using EP to solve inverse problems in civil engineering. In order to prevent the chromosomes from converging prematurely at early generations and stagnating to local optima at later generations, the exponential fitness scaling scheme presented by Gen et al (1996) was adopted in the evaluation and the simulated annealing was employed in the selection of the chromosomes. With the mutation magnitude being dynamically narrowed as the generation increased, the algorithm was more likely to be able to obtain a better chromosome through the mutation operator. Finite element method was incorporated with their EP algorithm (FEMEP) to perform the necessary structural analysis. A square plate under uniaxial tension and a simplified highway pavement model were examined to illustrate the effectiveness of the FEMEP algorithm. They first analyzed the models under static loads using FEM with known mechanical properties. Then the mechanical properties were assumed to be unknown and to be identified. In EP, chromosomes were comprised of variables representing the Young’s moduli of the models, and the fitness of each chromosome was defined as the difference between the calculated displacements at certain nodes and the previously obtained results. By minimizing the fitness function, the Young’s moduli of the models were obtained. Their results have demonstrated that EP is a promising search method for inverse problems in civil engineering.

Li and Wilkinson (2001) utilized the standard EP algorithm in chemical engineering to determine the particle size distribution and sample concentrations from spectral extinction measurements. Simulation tests were successfully performed to recover mono-modal and bi-modal particle size distributions. They found that EP is a useful tool for particle sizing by light attenuation because it has attractive advantages of high stability, insensitivity to measurement noise and distribution shape, low requirement for a priori knowledge and few constraints.

Tsay et al (2001) investigated the possibility of using EP to solve the real-world complicated, nonlinear, non-analytical and discontinuous problems. EP was employed to minimize the overall cost of a cogeneration system with constraints. The first generation was initialized randomly under the constraints. New individuals were generated from the parents by adding a Gaussian random variable. The variance of the random variable was directly proportional to their fitness and inversely proportional to the generation number. They claimed that this mechanism could accelerate the solution process and rapidly escape from the local optimal.

Li et al (2002) employed self-adaptive EP, fast EP and improved fast EP to determine the material constants within the unified creep damage constitutive equations from the experimental data. The successful application to an aluminum alloy demonstrated the search effectiveness and efficiency of EP. The results showed that EP is a robust optimization technique and could deal with complex, non-convex and non-differentiable functions. They claimed that EP algorithms “use real-value vectors and self-adaptation in their chromosome representation and thus perform better than binary genetic algorithms on complex continuous functions”.

Wong and Algie (2002) proposed an EP-based algorithm for determining the optimal combined heat and power dispatch solution of single-area cogeneration system. Domain-specific knowledge was embedded in the algorithm to ensure all of the individuals satisfy the power and heat demands. Expressions for the standard deviation used in the mutation process were designed such that the size of the mutation search range could be controlled and the neighborhood of the best individual in a population be searched. Examples showed that their algorithm has excellent performance when applied to a single-meal cogeneration system.

EP has also been used to train artificial neural networks (ANNs). Compared with the backpropagation algorithm, which is generally used for ANN'S training, EP is more powerful in finding the global minimum. In view of this advantage, Abbass (2002) proposed an evolutionary artificial neural network approach using EP to train ANNs. His approach was employed for the prediction of breast cancer.

Experiments showed that this approach has better generalization and much lower computational cost compared with the standard backpropagation.

Li and Kwan (2003) presented a hybrid GA for the bi-objective public transport driver scheduling problem. A driver schedule contained a set of shifts that covered all the required work, and each shift has an associated cost. The driver scheduling problem was to find a cover with the minimum cost using the minimum number of shifts. These two objectives were formulated as a fuzzified criterion. GA was used to derive a near-optimal weight distribution amongst the fuzzified criteria, so that a single valued weighted evaluation could be computed for each potential schedule. The benchmark experimental results demonstrated that this method is capable of solving large scale real-world driver scheduling problem.

El-Sharkh and El-Keib (2003) presented an EP-based technique to the unified model of the maintenance scheduling problem of power generation and transmission systems. In their method, if an infeasible solution was generated during the evolutionary process, the hill-climbing technique was used to move the infeasible solution to the feasible region. The viability of the approach and its ability to solve the integrated maintenance scheduling problems have been tested on the IEEE 30-bus system.

Pawar and Ganguli (2003) used a genetic fuzzy system to find the location and extent of damage in beams. Using the changes in the natural frequencies, a fuzzy system was generated and the rule-base and membership functions were optimized by GA. The output faults of the fuzzy system were four levels of damage (undamaged, slight, moderate and severe) at five locations along the beam (root, inboard, center, outboard and tip). The genetic fuzzy system developed for a noise level of 0.20 in the data gave a fault isolation success rate of 99.81% when the first eight natural frequencies were used. With noise level of 0.15 and 0.10, accuracy rates of 100% were obtained even when only the first four natural frequencies were used. The fuzzy system also showed excellent robustness with missing measurements and gradual degrades in the presence of faulty sensors/measurements.

Chapter 2: Evolutionary Computation

The genetic fuzzy system also gave very good results for BO-105 hingeless helicopter rotor blade for frequency and mode shape-based data.

Since EC has different merits and each suitable for different problems, some researchers have hybridized different EC techniques such that the desired advantages of the different techniques can be exploited. Howard and D'Angelo (1995) firstly hybridized GP and GA for data modeling. They used it to find a mathematical relationship between physical and biological parameters. Cao et al (1998) used a GA to optimize the parameters of the tree-structured GP individuals to preserve useful structures in evolving better differential equations to fit a given data set. They latter (1999) extended this concept to model higher order differential equations for dynamic systems. Swain and Morris (2003) developed a hybrid GP and EP approach to automatically generate mathematical models fi-om database. In their method, only the parameters in the selected GP individuals were optimized by EP to save computing time.

From the above examples, it is apparent that EC techniques are effective and are promising methods for global optimization. EC can be applied to solve those real-world optimization problems with rough, discontinuous, non-differentiable, multi-modal, noisy, and even fractallandscape, which cannot be handled by the traditional methods. EP is more suitable for continuous parameter optimization problems than the classical GA with binary string. However, researchers also pointed out the drawback of time consuming calculation. For successful application of EC techniques in real-world problem, the search efficiency should be improved.

CHAPTER 3 A HYBRID EP ALGORITHM

3.1 INTRODUCTION

The robustness of EC techniques in finding the global optimum of real-world optimization problems has been well proven. However, the search efficiency, i.e., the speed of convergence to the global optimum, could be slow for certain problems. There have been a large number of papers contributed to improve the search efficiency of EC, as reviewed in the previous chapter. Most of them did it by presenting special mutation strategies. On the other hand, local search using domain-specific knowledge is often employed to speed up the convergence of the EC algorithms on difficult problems (Boschetti et al 1996).

Applying local search operator to the individuals in EC algorithm usually moves the individual from its current location to the nearest optimum. If the evolutionary algorithm was able to get to the basin of attraction for the global optimum, applying local search operator can find the global optimum very quickly. Hart (1994) and Hart and Belew (1996) investigated the effect of local search operators in GA. Chellapilla et al (1998) and Birruet al (1999) embedded various local search operators into the self-adaptive EP and the fast EP to investigate the effectiveness of local search operators in EP. They concluded that "the local search operators can statistically and significantly enhance the performance of EP both in terms of the rate of convergence and the quality of the final solutions obtained".

Embedding local search as an operator into an EC algorithm may speed up the rate of convergence. However, the converged solution could only be a local one. For global optimization, the EP method should be carefully designed so that the algorithm would not be trapped in local optima. The hybrid EP algorithm presented in the next section aims to accelerate the EP search without losing its robustness in finding the global optimum.

3.2 ALGORITHM

An optimization problem can generally be written as follows:

$$\text{minimize } f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \dots, x_n) \in R^n \quad (3.1)$$

where $\mathbf{x} \in S \cap F$, $S \subseteq R^n$ defines the *search space* which is an n -dimensional space bounded by the *parametric constraints*

$$x_j^L \leq x_j \leq x_j^U, \quad j = 1, \dots, n \quad (3.2)$$

The feasible region F is defined by

$$F = \left\{ \mathbf{x} \in R^n \left| \begin{array}{l} g_n(\mathbf{x}) \leq 0, \quad j = 1, \dots, l \\ h_k(\mathbf{x}) = 0, \quad k = 1, \dots, m \end{array} \right. \right\} \quad (3.3)$$

where $g_n(x)$ and $h_k(x)$ represent inequality and equality *constraints*, respectively. The m equality constraints imply that m variables can be represented in terms of the remaining variables, thus can be eliminated. Therefore, the original n -dimensional optimization problem with inequality and equality constraints is equivalent to an $(n-m)$ -dimensional problem with only inequality constraints. The latter is considered hereafter.

When EP is applied to solve this continuous optimization problem, the possible solution is directly encoded as a continuous vector. Sometimes there is also a strategy parameter vector to control the mutation step size. The first population is generated randomly in the feasible domain. New individuals are generated by applying various variation operators to the parents. Better individuals are selected to

form the next generation. After iteration of the loop of mutation-evaluation-selection, a near optimum solution can be obtained.

3.2.1 Representation structure

In the traditional EP method, each chromosome contains a vector of the objective variables $\mathbf{x}$ and a vector of the strategy parameters σ . Such a representation structure is also adopted in this study. However, in this hybrid method, the strategy parameter, denoted as $\mathbf{s}$, represents the mutation step instead of the deviation of a certain distribution in the traditional EP. Thus, the chromosome is represented as $(\mathbf{x}, \mathbf{s})$ in this hybrid algorithm.

3.2.2 Variation Operators

Three kinds of variation operators, namely macro mutation (MM) operator, local linear bisection search (LBS) operator, and crossover operator are proposed. Consequently, the hybrid EP algorithm is named as EP-MMLSX.

1) Local linear bisection search (LBS) operator

The objective of LBS is to exploit the neighbourhood of the parent chromosome. The LBS operator is designed as follows:

For the j -th variable in the chromosome, a variation is applied: $x_j' = x_j \pm s_j$, where $\pm$ is selected according to the gradient g_j . For example, for a maximum problem, if g_j is greater than zero, $x_j' = x_j + s_j$, otherwise, subtraction is adopted. If the mutated chromosome is not better than its parent, s_j reduces to $s_j/2$ until a better solution is found or the mutation step s_j is less than a preset small number which depends

expected accuracy. The mutation step is adjusted as shown in Fig. 3.1(a) for a 1-dimensional maximization problem.

The LBS operator is applied to each variable in the chromosome. The process of local bisection search is similar to hill-climbing, as illustrated in Fig. 3.1 (b) for a 2-dimensional problem. The search starts in x_1 direction to find the ridge of the fitness landscape, and continues to "climb" in x_2 direction from the new point. This process iterates for all directions. It is expected that a point closer to a local optimum can be found by LBS. Bisection method is employed to adjust the mutation step because it requires no specific knowledge about the fitness landscape. And thus, the LBS operator is a general method to be applied to any optimization problem.

Fig. 3.1. Mechanism of LBS operator (a) a one dimensional maximization problem. By adjusting the mutation step from s_j to s_j' , an offspring with better fitness is generated. (b) a two dimensional maximization problem. The individual climbs in x_1 and x_2 directions continuously to approach the peak of the fitness landscape.

The first derivative of the fitness function is needed for the LBS operator. Since the gradient g_j is not available in most cases, it is calculated by an approximate numerical method in this algorithm, i.e.,

$$g_j = \frac{f(\mathbf{x} + \Delta x_j) - f(\mathbf{x})}{\Delta x_j}, \text{ where } \Delta x_j \text{ is set at a small value as } x_j \cdot \varepsilon \text{ for}$$

$x_j \neq 0$, or ε for $x_j = 0$, where ε is a small positive value.

The LBS operator can help the algorithm to find a local optimum very quickly. This local optimum is not necessarily the global one. However, if a solution near to the global optimum is found, the employment of LBS can significantly accelerate the speed of convergence of the algorithm to the global optimum.

2) Macro mutation (MM) operator

As stated previously, the employment of LBS operator can accelerate the convergence speed. However, the risk of being trapped in a local optimum also increases. To overcome this problem, MM operator is designed and employed in the algorithm. The objective of MM operator is to explore the search space and to escape from the local optimum trap. The mechanism of MM is defined as follows:

For a parent chromosome $\mathbf{x} = (x_1, x_2, \dots, x_{n-m})$, a **move** is added to generate an offspring. The **move** is randomly selected within the *search space* S . Thus $move_j = U(x_j^L, x_j^U)$, where $U(x_j^L, x_j^U)$ is a random number uniformly distributed between x_j^L and x_j^U , which are the lower and upper bounds of the *j-th* variable, respectively. The new chromosome $\mathbf{x}' = (x_1 + U(x_1^L, x_1^U), x_2 + U(x_2^L, x_2^U), \dots, x_{n-m} + U(x_{n-m}^L, x_{n-m}^U))$, where the element $x_j' = x_j + U(x_j^L, x_j^U)$ is uniformly distributed in $[x_j + x_j^L, x_j^U]$ or $[x_j^L, x_j + x_j^U]$ depending on $x_j \geq 0$ or $x_j < 0$. It is observed that very weak connection exists between the offspring and the parent lower bound of the variable. Hence, the al

Since the distance between the offspring chromosome to the global optimum is unknown, the mutation step s is set as its upper bound. Thus,

$$x_j' = x_j + U(x_j^L, \bar{x}_j^U), s_j' = x_j^U - x_j^L \quad (3.4)$$

3) Crossover

The mechanism of mutation-selection without crossover in the traditional EP has received considerable criticism from researchers working in the field of GAs (Goldberg 1989). Holland (1975) described evolution based on mutation in the absence of other genetic operators as being an "enumerative search", in large measure equivalent to blindly selecting solutions at random. In this hybrid EP algorithm, an arithmetical crossover (Michalewicz 1996) is employed to explore the search space and to maintain the diversity of the population. The mechanism of crossover is adopted as follows:

$$\begin{aligned} x_j^1' &= \alpha_j x_j^1 + (1 - \alpha_j) x_j^2, s_j^1' = \alpha_j s_j^1 + (1 - \alpha_j) s_j^2 \\ x_j^2' &= (1 - \alpha_j) x_j^1 + \alpha_j x_j^2, s_j^2' = (1 - \alpha_j) s_j^1 + \alpha_j s_j^2 \end{aligned} \quad (3.5)$$

where the superscripts 1 and 2 represent the first and the second parent/offspring, respectively. The quotation mark stands for the offspring chromosome; and α_j is a random number uniformly distributed in $[0,1]$.

3.2.3 Handling Constraints

How to handle the constraints is an important issue in the application of EC techniques. Constrained optimization may be addressed through imposition of the requirement that (i) a penalty function is applied to offsprings lying outside the constraint bounds in such a manner that they do not become part of the next generation or (ii) the mutation operator is formulated to only generate legitimate solutions. For the first method, the penalty function is sometimes difficult to choose. So the second method is adopted in this algorithm to handle the constraints. If the

offspring is beyond the constraints, a repairing scheme will be employed to mend it into the feasible region F .

Generally, F is connected or is a disjoint sum of connected components. For an interior point $\mathbf{x} = (x_1, x_2, \dots, x_{n-m})$ in F , there exists an $\varepsilon > 0$ so that any point $\mathbf{x}' (x_j - \varepsilon < x_j' < x_j + \varepsilon, j = 1, 2, \dots, n-m) \in F$. According to the mechanism of MM, LBS and crossover operator, the offspring is generated by applying a **move** to the parent:

$$\mathbf{x}' = \mathbf{x} + \text{move} \tag{3.7}$$

In case the new chromosome is beyond F , a repairing scheme is employed: the **move** decreases by multiplying a random number between 0 and 1, i.e., $\text{move} = U(0,1) \cdot \text{move}$. Within limited time, $\max(\text{move}_j)$ can be reduced to be less than ε and a feasible offspring can be generated. In case a feasible solution cannot be obtained after $\max(\text{move}_j)$ having been reduced to be less than ε , it indicates that the parent point locates at the boundary of F and the direction of move_j has been going out of F . Then, the direction of move_j is flipped. With this method, a feasible offspring can be ensured. Fig. 3.2 illustrates how the repairing scheme works for crossover, where F includes two regions $F1$ and $F2$. For the mother and father from different regions, the offspring generated by crossover will probably

Fig. 3.2. The repairing scheme for crossover

Chapter 3: A Hybrid EP Algorithm

locate in $F1$ (Offspring1) or $F2$ (Offspring2) or beyond F (Offspring3). According to the repairing scheme, the **move** for Offspring3 will be decreased randomly to **move'**, such that Offspring3 can be repaired to a feasible solution (Offspring3'). With this repairing scheme, a feasible solution can be generated within limited times of repairing.

3.2.4 Selection

A tournament selection is carried out in the hybrid EP-MMLSX method. Each chromosome obtains a tournament "weight" by competition with other q randomly selected opponents. Minster et al (1995) observed that the population tends to stagnate in a neighborhood representing an incorrect solution when pure tournament selection is used. In order to prevent such premature convergence to a local optimum, they combined EP with simulated annealing schedule and found such a combination made the convergence to the correct solution more reliable. This approach is adopted in the hybrid method. The tournament weight W_i of the i -th chromosome is initially set on zero and increased based on the competition as follows:

$$W_i = \begin{cases} W_i + 1 & \text{if } f_i \text{ is equal or better than } f_n \\ W_i + P(\Delta E) & \text{if } f_i \text{ is worse than } f_n \end{cases} \quad (3.8)$$

where f_i is the fitness of the i -th chromosome; f_n is the fitness of a randomly selected opponent; and $P(\Delta E)$ is the probability given by a Gibbs distribution:

$$P(\Delta E) = \exp\left(-\frac{f_n - f_i}{T_0 \beta^{n_g}}\right) \quad (3.9)$$

where T_0 is the initial temperature and is set at $T_0 = (f_{\max} - f_{\min})/2$ so that $(f_n - f_i)/T_0$ is within $[0, 23]$; n_g is the present generation index; and β is the cooling parameter, which is less than 1 to get the similar effect of annealing.

3.2.5 Procedure

Similar to the other evolutionary computation algorithms, the proposed algorithm first generates a pool of possible solutions. Then in the following generations, each parent chromosome is assigned a random number uniformly distributed from 0 to 1. Depending on which range this number is located in, i.e., $[0, p_L]$, $(p_L, p_L + p_c)$ or $[p_L + p_c, 1]$, LBS or crossover or MM operator is applied to the parent to generate a new chromosome, where p_L and p_c are the probabilities of LBS and crossover operators, respectively. After this, the tournament selection is performed on all parent and offspring chromosomes to form a new generation. The flowchart of the proposed algorithm is shown in Fig. 3.3.

Fig. 3.3. Flowchart for proposed EP-MMLSX algorithm.

3.3 CASE STUDY AND OVERALL ASSESSMENT OF ALGORITHM

3.3.1 Test Functions

The following 12 benchmark problems are selected to assess the performance of the proposed algorithm.

$$1) \max f_1(x) = \sum_{i=1}^n x_i \sin(i\pi x_i); \quad \sum_{i=1}^n x_i^2 \leq 100.0.$$

$$2) \max f_2(x) = \frac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} + \frac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3},$$

$$x_1 + x_2 - x_3 \leq 1, \quad -x_1 + x_2 - x_3 \leq -1, \quad 12x_1 + 5x_2 + 12x_3 \leq 34.8,$$

$$12x_1 + 12x_2 + 7x_3 \leq 29.1, \quad -6x_1 + x_2 + x_3 \leq -4.1, \quad 0 \leq x_i, \quad i = 1, 2, 3.$$

The global solution is $x^* = (1, 0, 0)$, and $f(x^*) = 2.471429$.

$$3) \max f_3(x) = -x \sin(10\pi x) + 1; \quad -2.0 \leq x \leq 1.0.$$

$$4) \max f_4(x, y) = 0.5 + \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001(x^2 + y^2))^2}; \quad -100 \leq x, y \leq 100.$$

$$5) \min f_5(x, y) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10y - 0.5 \sum_{i=1}^5 x_i^2$$

$$6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 \leq 56.5, \quad 10x_1 + 10x_3 + y \leq 20,$$

$$0 \leq x_i \leq 1, \quad 0 \leq y.$$

The global solution is $(x^*, y^*) = (0, 1, 0, 1, 1, 20)$, and $f(x^*, y^*) = -213$.

$$6) \min f_6(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{i=1}^{10} x_i} \right)$$

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} = 2, \quad x_4 + 2x_5 + x_6 + x_7 = 1,$$

$$x_3 + x_7 + x_8 + 2x_9 + x_{10} = 1, \quad x_i \geq 0.000001.$$

$$\text{where } c_1 = -6.089, \quad c_2 = -17.164, \quad c_3 = -34.054,$$

Chapter 3: A Hybrid EP Algorithm

$$\begin{aligned}
c_4 &= -5.914, & c_5 &= -24.721, & c_6 &= -14.986, \\
c_7 &= -24.100, & c_8 &= -10.708, & c_9 &= -26.662, \\
c_{10} &= -22.179.
\end{aligned}$$

$$7) \min f_7(x) = \sum_{i=1}^n x_i^2; \quad n=3; \quad -6 \leq x_i \leq 6, \quad i=1, \dots, n.$$

$$8) \min f_8(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2; \quad -5.12 \leq x_i \leq 5.12, \quad i=1, 2.$$

$$\begin{aligned}
9) \min f_9(x) &= \frac{1}{\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6}} - 0.9980038, \\
&-65 \leq x_i \leq 65, \quad i=1, 2;
\end{aligned}$$

where

$$\begin{bmatrix} a_{1j} \\ a_{2j} \end{bmatrix} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \dots & 32 & 32 & 32 \end{bmatrix}.$$

$$10) \min f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e;$$

$$n=30; \quad -30 \leq x_i \leq 30, \quad i=1, \dots, n.$$

$$11) \min f_{11}(x) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i));$$

$$n=30; \quad -5.12 \leq x_i \leq 5.12, \quad i=1, \dots, n,$$

$$12) \min f_{12}(x) = \sum_{i=1}^n \frac{x_i^2}{4000} + 1 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right);$$

$$n=30; \quad -600 \leq x_i \leq 600, \quad i=1, \dots, n.$$

These 12 problems have been studied by other researchers. f_1 and f_2 have been studied by Soh and Dong (2001) with their improved EP method (SDEP). $f_3 \sim f_6$ have been studied by Michalewicz (1996) using a GA with varying population size (GAVaPS) and a GA for numerical optimization for constrained problems (GENOCOP). Choi and Oh (2000) studied $f_7 \sim f_9$ using the self-adaptive EP (SAEP), the accelerated EP (AEP), and a conventional EP with a

Chapter 3: A Hybrid EP Algorithm

mutation operator inspired by neural network backpropagation learning (CEPBLO). $f_9 \sim f_{12}$ have been studied by Yao et al (1999) using conventional EP (CEP) and fast EP (FEP). Chellapilla (1998) has also studied functions $f_{10} \sim f_{12}$ by using conventional EP with Gaussian mutation operator (CEP/GMO), Cauchy mutation operator (CEP/CMO), mean mutation operator (CEP/MMO) and adaptive mean mutation operator (CEP/AMMO). f_{10} with a larger parameter bound has been studied by Chellapilla and Fogel (1997).

In SDEP (Soh and Dong 2001), the population size is tuned to 1000 and 400 for f_1 and f_2 , respectively. In the other methods, the parameters are not explicitly tuned. The population size varies from 45 (Choi and Oh 2000), 50 (Chellapilla 1998, Chellapilla and Fogel 1997), 70 (Michalewicz 1996) to 100 (Yao et al 1999). Although the population size is different in the previous methods, a number of 10 is generally adopted as the tournament size in the EP-based methods.

3.3.2 Parameters Adopted

There are several parameters to be pre-determined by the users. Following typical implementations (Chellapilla and Fogel 1997, Chellapilla 1998), the numerical experiments are carried out using a population size of 50 and a tournament size $q = 10$ in this study. Probabilities for MM, LBS and crossover operator are set as $p_M = 0.6$, $p_L = 0.2$ and $p_C = 0.2$, respectively. For direct comparison, the program terminates when the number of function evaluations reaches the number reported in the corresponding literatures.

The simulated annealing scheme adjusts the selection pressure in the early generations. The simulated annealing is expected to take effect in the first N generations, which means that after N generations, $P(\Delta E)$ is small enough to be neglected. If q opponents are selected for competition, $P(\Delta E) = \frac{1}{q}$ can be viewed

as a small value. Since $(f_n - f_i)/T_0 \in [0, 2]$, we use the average value of 1.0 and from Eqn. (3.9), the cooling parameter β is calculated as $\sqrt[N]{1/\ln q}$.

The value of the cooling parameter is determined by experiments. Fig. 3.4 illustrates the influence of β value on the performance of EP-MMLSX algorithm based on f_1 as an example. Different values of β are selected, i.e, 0, 0.91998, 0.95916, 0.97258, 0.97937, 0.98346 and 0.98994, corresponding to $N=0, 10, 20, 30, 40, 50$ and 60, respectively. When $N=0$, it indicates that the simulated annealing is not activated in the algorithm. Fig. 3.4 shows that the algorithm has better performance when N is between 30–50, which is consistent with the recommendations of Minster et al (1995). In this study, $\beta = 0.97258$ is selected as the cooling parameter.

Fig. 3.4. Effect of simulated annealing on performance of EP-MMLSX for function f_1

3.3.3 Results and Comparison

For test function f_1 , the best known function values for cases when the dimension of the function is 3, 5, 7 and 9, respectively, have been reported by Soh and Dong (2001). Table 1 shows a comparison between the best results obtained by EP-MMLSX and SDEP. From Table 3.1, it is observed that EP-MMLSX can find a better function value. The relative increase can be up to more than 5 percent when the dimension is 9.

Table 3.1 Comparison between the best results of test function f_1 obtained by EP-MMLSX and SDEP (Soh and Dong 2001)

Dimension of the Test Functions	Number of Function Evaluations	Best results obtained by SDEP	Best results found by EP-MMLSX	Relative Increase (%)	Solutions
3	150,000	17.2445	17.2445	0.00	(-6.4955,5.2498,-5.49982)
	300,000	17.2445	17.2445	0.00	(-6.4955,5.2498,-5.49982)
5	150,000	22.1569	22.3194	0.73	(-4.50937,-4.25292,4.83412,-4.62556,4.10045)
	300,000	22.1579	22.3194	0.73	(-4.50937,-4.25292,4.83412,-4.62556,4.10045)
7	150,000	25.8743	26.3133	1.70	(-4.49526,3.24768,-3.49835,4.12449,-3.69964,3.7496,3.49978)
	300,000	25.9472	26.3136	1.41	(-4.49468,3.24789,-3.49865,4.12424,-3.69978,3.74986,3.4999)
9	150,000	27.9138	29.6309	6.15	(2.53722,3.2389,2.8306,4.12285,3.6988,3.41647,2.64187,3.56221,3.61106)
	300,000	28.027	29.6407	5.76	(2.53066,3.24205,2.8315,4.12359,3.69924,3.41629,2.64261,3.56176,3.61089)

Chapter 3: A Hybrid EP Algorithm

For the other eleven test functions $f_2 \sim f_{12}$, 20 independent trial runs of EP-MMLSX are conducted on each of them. The average results are correspondingly compared with those reported in the previous works with the same number of function evaluations, which takes into account the computing cost of both macro and local search.

Soh and Dong (2001) have also studied the test function f_2 . They reported that a run of 16,000 function evaluations with their improved EP method obtained the known global optimum and the convergence was faster than all of the compared methods. With EP-MMLSX, not only the best trial, but all of the 20 runs obtain the global optimum within 10,000 function evaluations.

Michalewicz (1996) studied f_3 and f_4 using GAVaPS. In 20 independent runs, after 3,000 function evaluations, the best values for f_3 and f_4 were found to be 2.841 and 0.970 respectively. With EP-MMLSX the average results are also better than the above results, which are 2.950 and 0.987, respectively.

Michalewicz (1996) studied f_5 and f_6 using GENOCOP, and reported that GENOCOP found the known global optimum (-213.0) of f_5 in 70,000 function evaluations. In comparison, EP-MMLSX obtains the global optimum in less than 30,000 function evaluations in the trials. For the test function f_6 , the global optimum is unknown. Michalewicz (1996) reported -47.760765 as the best solution ever known. The number of function evaluations was not given. However, EP-MMLSX found a point with a better function value of -47.7611. This point is

$$x = (0.0406707, 0.147737, 0.783142, 0.00141423, 0.485246, 0.000693211, \\ 0.027400559, 0.0179473, 0.037316026, 0.096878089)$$

All of the 20 independent runs of EP-MMLSX converge to this point within 10,000 function evaluations.

$f_7 \sim f_9$ have been studied by Choi and Oh (2000) using SAEP, AEP and CEP/BLO. Comparison between these methods and EP-MMLSX is shown in Table

3.2, where MBF signifies the mean best fitness and NFE the number of function evaluation. It should be mentioned that the data from Choi and Oh (2000) are obtained from their figures instead of a table, so there may be some minor errors, but it will not affect the overall assessment. For the test function f_7 , all four methods converge very quickly and their performances are comparable. For the test function f_8 , EP-MMLSX is as good as SAEP and AEP, but CEP/BLO converged faster than all of them. Both EP-MMLSX and CEP/BLO have a much better performance than SAEP and AEP for the test function f_9 .

Table 3.2 Comparison between performance of EP-MMLSX and SAEP, AEP, CEP/BLO for test functions $f_7 \sim f_9$

Test functions	SAEP		AEP		CEP/BLO		EP-MMLSX	
	NFE	MBF	NFE	MBF	NFE	MBF	NFE	MBF
f_7	4200	1E-10	6000	1E-10	1200	1E-10	4178	5.4E-11
f_8	45000	9E-5	45000	5E-6	5850	1E-10	45000	5.9E-4
f_9	45000	5E0	45000	1E-5	2000	4E-8	13954	3.8E-8

Note: NFE means the number of function evaluations.

MBF means the mean best fitness.

CEP, FEP, CEP/GMO, CEP/CMO, CEP/MMO and CEP/AMMO were employed to study the test functions $f_9 \sim f_{12}$ by Yao et al (1999), Chellapilla (1998) and Chellapilla and Fogel (1997). Table 3.3 shows a comparison between these methods and EP-MMLSX on these four test functions. With the same number of function evaluations, EP-MMLSX statistically found results better than CEP with various operators. Compared with FEP, EP-MMLSX algorithm has better performance for f_9 and f_{12} but worse for f_{10} and f_{11} .

Chapter 3: A Hybrid EP Algorithm

Table 3.3. Comparison between performance of EP-MMLSX and CEP, FEP, CEPIGMO, CEPICMO, CEP/MMO and CEP/AMMO for test functions f_9 – f_{12}

	f_9		f_{10}		f_{11}		f_{12}	
	NFE	MBF	NFE	MBF	NFE	MBF	NFE	MBF
CEP (Yao et al 1999)	10000	0.66	150000	9.2	500000	89.0	200000	8.6E-2
FEP (Yao et al 1999)	10000	0.22	150000	1.8E-2	500000	4.6E-2	200000	1.6E-2
CEP/GMO (Chellapilla and Fogel 1997)	N/A	N/A	150000	14.9973	N/A	N/A	N/A	N/A
CEP/CMO (Chellapilla and Fogel 1997)	N/A	N/A	150000	6.9304	N/A	N/A	N/A	N/A
CEP/MMO (Chellapilla and Fogel 1997)	N/A	N/A	150000	5.3377	N/A	N/A	N/A	N/A
CEP/AMMO (Chellapilla and Fogel 1997)	N/A	N/A	150000	6.0053	N/A	N/A	N/A	N/A
CEP/GMO (Chellapilla 1998)	N/A	N/A	150000	10.47	250000	113.72	250000	6.05
CEP/CMO (Chellapilla 1998)	N/A	N/A	150000	4.52	250000	55.82	250000	7.08
CEP/MMO (Chellapilla 1998)	N/A	N/A	150000	3.75	250000	46.96	250000	3.84
CEP/AMMO (Chellapilla 1998)	N/A	N/A	150000	3.41	250000	52.59	250000	1.85
EP-MMLSX	13954	3.8E-8	149868	2.14125	249955	21.3419	249561	1.045E-2

Note: NFE means the number of function evaluations.

MBF means the mean best fitness.

N/A means not available.

According to the comparison, the EP-MMLSX algorithm obtained a better value and/or converged faster to the global optimum than SDEP, GAVaPS and

GENOCOP for the test functions $f_1 \sim f_6$. The proposed algorithm also performed better than SAEP and AEP for the test functions $f_7 \sim f_9$. Moreover, it was able to statistically obtain a better value than those conventional EP with various mutation rules for the test functions $f_{10} \sim f_{12}$. It was also better than FEP for f_9 and f_{12} . However, the EP-MMLSX algorithm was not as good as FEP for f_{10} and f_{11} and not efficient as CEP/BLO for f_8 . In general, the numerical experiments on the 12 benchmark problems show that the EP-MMLSX algorithm has a higher effectiveness and efficiency for the majority of the test functions.

3.4 INFLUENCE OF DIFFERENT OPERATORS

As shown above, the EP-MMLSX algorithm is able to find the global optimum effectively and efficiently. However, the influence of different operators, i.e., MM, LBS and crossover, on the performance of the algorithm is not clear. In this section, their effects are investigated by enabling certain operators, e.g., EP-MM, implying that only MM operator is enabled in the algorithm. Considering the fact that LBS operator is designed to speed up the convergence to local optima, it is not reasonable to compare the results of EP-LBS (the algorithm with only LBS operator) and EP-MM for those problems with a large number of local optima. The effect of LBS operator is analyzed by comparing the results of EP-MM and EP-MMLS (both MM and LBS operators are enabled). Similarly, the effect of crossover is investigated by the comparison between the results of EP-MMLS and EP-MMLSX.

Moreover, some other local search operators developed by previous researchers can be used to replace the proposed LBS operator for certain problems. Hence, the effectiveness of (1+1) ES (van Kemenade 1995), Solis and Wets method (Solis and Wets 1981) and Davidon-Fletcher-Powell method (Davidon 1959, Fletcher and Powell 1963) are also investigated with the combination of MM operator. The corresponding algorithms are denoted as EP-MMES, EP-MMSW and

EP-MMDFP, respectively. A summary of the abbreviations for the various versions of the algorithm is listed in Table 3.4. The objective of the comparison is to give some guidelines for selecting suitable local search operators for certain optimization problems.

Table 3.4. Summary of abbreviations of algorithms

Abbreviations in Text	Abbreviations in Table 3.5 and Figs. 3.5~3.16	Descriptions
EP-MM	M	The algorithm employs MM operator only.
EP-MMDFP	D	The algorithm employs MM and DFP operators.
EP-MMES	E	The algorithm employs MM and (1+1) ES operators.
EP-MMLS	L	The algorithm employs MM and LBS operators.
EP-MMLSX	LX	The algorithm employs MM, LBS and crossover operators.
EP-MMSW	S	The algorithm employs MM and SW operators.

The selected local search operators are briefly described as follows:

- 1) (1+1)-ES: A (1+1) Evolutionary Strategy was used as a local optimizer by van Kemenade (1995). The offspring is generated based on the Gaussian mutation from the parent. The better of the parent and offspring is selected to survive. The procedure iterates to find a nearby optimum. This local optimizer makes as few assumptions as possible regarding the actual shape of fitness landscape it is traveling through, and thus, suitable for any kind of problems.
- 2) Solis and Wets method: A new point is generated by adding the normally distributed zero mean deviates to the current point. If the function value at the new point is worse than the current point, the algorithm examines the opposite point generated by taking a step in the opposite direction. If neither

of these steps generates a better point, a new point is then generated in the next iteration.

- 3) Davidon-Fletcher-Powell gradient search method: As a quasi-Newton method, it can have quadratic termination that needs only the first derivatives and thus is economical on function evaluations. For the trial solution x_0 and the initial matrix A_0 , it repeats the following steps:

1. $s_k = -A_k \cdot g_k$, $x_{k+1} = x_k + s_k$;
2. $y_k = g_{k+1} - g_k$, $z_k = A_k \cdot y_k$;
3. $A_{k+1} = A_k + (s_k \cdot s_k^T) / (s_k \cdot y_k^T) - (z_k \cdot z_k^T) / (y_k \cdot z_k^T)$.

where the subscript k represents the number of iterations and g is the gradient vector at point x .

In EP-MMES and EP-MMSW, the initial value for the strategy parameter is set at one-tenth of the parametric bound, The matrix A_0 in EP-MMDFP is initially set as an identity matrix. The probabilities of the local search operators are set at 0.2, and the local search has 5 iterations for all of the algorithms. f_1 is studied with dimension 3 only, for simplicity. 20 independent runs are performed for each algorithm on each problem. The performances of the algorithms for the 12 test functions are illustrated in Figs.3.5~3.16. The comparison is based on the number of function evaluations so that the computing cost for local search is also considered.

Fig.3.5. Performances of the 6 algorithms for test function f_1

Fig.3.6. Performances of the 6 algorithms for test function f_2

Fig. 3.7. Performances of the 6 algorithms for test function f_3

Fig. 3.8. Performances of the 6 algorithms for test function f_4

Fig. 3.9. Performances of the 6 algorithms for test function f_5

Fig. 3.10. Performances of the 6 algorithms for test function f_6

Fig. 3.11. Performances of the 6 algorithms for test function f_7

Fig. 3.12. Performances of the 6 algorithms for test function f_8

Fig. 3.13. Performances of the 6 algorithms for test function f_9

Fig. 3.14. Performances of the 6 algorithms for test function f_{10}

Fig. 3.15. Performances of the 6 algorithms for test function f_{11}

Fig. 3.16. Performances of the 6 algorithms for test function f_{12}

Chapter 3: A Hybrid EP Algorithm

An unambiguous rank order based on the results is shown in Table 3.5. The rank is based on the mean best function value obtained by the algorithm at the end of the trial. For those cases where the value is the same, the speed of convergence is then considered. The algorithm with a higher convergence rate is given a higher rank. For example, the entry "M, (E, L, S, LX), D" for the results of f_1 indicates that the algorithm EP-MM generates solutions that have the best mean function value and that these values are unambiguously statistically and significantly better than those generated by the other algorithms. The keyword unambiguously means the results of EP-MM are statistically and significantly better in all comparisons with the other algorithms. The parentheses grouping E, L, S and LX indicate that the results for EP-MMES, EP-MMLS, EP-MMSW and EP-MMLSX are not all statistically and/or significantly different.

Table 3.5 Unambiguous ranking of the six algorithms

Test Functions	Dimension	Number of Function Evaluations	Characteristics	Unambiguous rank
f_1	3	150000	Multimodal	M, (E, L, S, LX), D
f_2	3	14000	Linear function with linear constraints	M, (S, E), (LX, L), D
f_3	1	1500	Multimodal	(M, S, E, L, LX, D)
f_4	2	15000	Multimodal	(LX, E), L, S, M, D
f_5	6	50000	Nonlinear function with linear constraints	L, LX, M, (S, E), D
f_6	10	100000	Multimodal	(D, L, LX), M, E, S
f_7	3	30000	Strictly convex	D, LX, L, S, E, M
f_8	2	40000	Global optimum resides inside a long, narrow, and parabolic-shaped flat valley.	E, D, LX, L, M, S
f_9	2	20000	Multimodal, with multiple deep local minima	E, LX, L, M, S, D
f_{10}	30	100000	Multimodal	LX, S, L, M, E, D
f_{11}	30	250000	Multimodal	(LX, L), (E, S), D, M
f_{12}	30	250000	A large convex domain exists in the neighborhood of the global optimum.	D, LX, L, S, E, M

Chapter 3: A Hybrid EP Algorithm

The following observations are obtained from Table 3.5 and Figs. 3.5~3.16.

- 1) *Effect of LBS operator*: It is observed that EP-MM performs better or as well as EP-MMLS for test functions f_1 , f_2 and f_3 . This is probably attributed to the small search space of the test functions. The number of independent variables is less than 3. The variables are constrained in $[-10, 10]$, $[0, 3]$ and $[-2, 1]$ for the three problems, respectively. The small search space can be explored by MM operator effectively. The gain of convergence acceleration by LBS can not balance the cost of fitness evaluation. Thus, the introduction of local search operators in the algorithm decreases the search efficiency. However, the EP algorithm with enabled LBS operator outperforms EP-MM for the other test functions. This demonstrates that the LBS operator can improve the convergence rate in EP searches, especially for problems with large search space.

- 2) *Effect of crossover*: The effect of crossover can be observed by comparing the performance of EP-MMLS and EP-MMLSX. In all cases except f_5 , the results of EP-MMLSX are better than or at least as good as those of EP-MMLS. Figs. 3.8, 3.11, 3.12, 3.13, 3.14, and 3.16 clearly show that when EP-MMLS stagnated in the later generations, EP-MMLSX can continue finding a better solution. This indicates that crossover can prevent the evolution from trapping in the local optima. Thus it is recommended that crossover should be employed in this EP algorithm.

- 3) *Some guidelines for selection of local search operators*: Different local search operators have different advantages over the others for solving various problems. For example, DFP method is more suitable for f_6 , f_7 and f_{12} . Both the algorithms EP-MMLS and EP-MMLSX have the similar performance as EP-MMDFP for f_6 . The EP-MMDFP algorithm has the best performance for function f_7 and f_{12} because f_7 is a spherical function which is strictly convex, and f_{12} has a relatively large convex domain

$\left[-\frac{\pi}{2}\sqrt{i}, \frac{\pi}{2}\sqrt{i}\right]$ in the neighborhood of the global optimum. On the other hand, EP-MMLS has better performance than EP-MMES and EP-MMSW for functions f_5 , f_6 , f_7 , f_{11} and f_{12} . These functions have at least 6 variables, except f_7 which is a strictly convex problem. Statistically, EP-MMES or EP-MMSW outperform EP-MMLS for problems with less than 3 variables i.e., f_2 , f_4 , f_8 and f_9 , except f_{12} , but they are defeated by EP-MMLS for the problems with more variables. Therefore, it is recommended that for problems with large convex domain in the neighborhood of the global optimum, DFP should be selected; for problems with fewer variables, ES and SW are more suitable; and for problems with more variables and numerous local optima, LBS is the best choice.

3.5 SUMMARY

When applying EP to real-world problems solving, one important consideration is the search efficiency. The proposed algorithm integrated with macro mutation, local linear bisection search and crossover operators aims to improve the performance of the conventional EP method. Numerical experiments on 12 benchmark problems have shown that the proposed method can search the global optimum effectively and efficiently, thus it could be an alternative search and optimization algorithm.

The influence of the operators on the overall performance of the algorithm is also investigated. The employment of LBS operator can improve the search efficiency of EP for problems with large search space. Crossover has also demonstrated its effectiveness in the search by preventing the evolution from being trapped in the local optima. Moreover, some other local search operators are also investigated. Some practical guidelines to select suitable local search operator are provided: DFP is suitable for problems with large convex domain in the neighborhood of the global optimum; ES and SW show their advantage to deal with

Chapter 3: A Hybrid EP Algorithm

problems with fewer variables; and LBS is recommended for problems with more variables and a large number of local optima.

The high effectiveness and efficiency of EP-MMLSX algorithm enhances the feasibility and practicality of applying EP-based backcalculation algorithm for structural nondestructive evaluation.

CHAPTER 4 PARAMETER IDENTIFICATION OF COMPOSITE PLATES

4.1 INTRODUCTION

Composite materials have certain advantages over the conventional materials, such as high strength to weight ratios, high stiffness to weight ratios, improved corrosion resistance, low specific gravity of fibers, low maintenance costs, etc. More importantly, composite materials can be tailored to meet certain design requirements for specific applications. Due to these advantages, composite materials have received extensive applications in the design of high-performance structural systems.

There are many methods of manufacturing laminated composite components (Lubin 1982, Schwartz 1983). Different manufacturing and curing processes will produce different mechanical properties for the components. To ensure reliability of the composite structural systems, the actual behaviors of the constituted laminated composite parts in service must be accurately predicted and carefully monitored. The material properties determined from the standard specimens tested in the laboratory may deviate significantly from those of the actual laminated composite components manufactured in the factory. Furthermore, laminated composite structures subjected to dynamic loads or operated in severe environments may experience progressive stiffness reduction or material degradation, which would finally lead to failure of the structures (Wang and Kam 2001). Therefore, it is

Chapter 4: Parameter Identification of Composite Plates

necessary to develop nondestructive technique to accurately determine the current mechanical properties of laminated composite structures.

Bar-Cohen (1986) reviewed the previous nondestructive evaluation (NDE) techniques for the determination of the material properties of laminated composite parts. These techniques have their limitations or specific difficulties in the identification of elastic constants, especially for structures in service. A number of researchers tried to identify the elastic parameters of laminated composites from the measured vibration performances. Kam and Liu (1998) developed a method to identify the element bending stiffness of beam structures using measured natural frequencies and mode shapes. Wilde and Sol (1987), Soares et al (1993), and Rikards et al (1999) used natural frequencies to identify the elastic properties of laminated composite plates. Ip et al (1998) used modal analysis and Rayleigh-Ritz method to characterize orthotropic shells. Wang and Kam (2001) also identified the elastic parameters of shear deformable laminated composite plates by measuring their static deformations subjected to point loads. Zou et al (2000) reviewed the vibration-based model-dependent damage identification and health monitoring methods for composite materials.

In the above process of parameter identification, an optimization technique has to be employed to minimize the differences between the predicted performances and the measured natural frequencies, mode shapes or static deformations. It is noted that a large number of local minima may exist in the feasible region, thus the conventional local optimization algorithms may not be suitable for this minimization problem. In this study, the hybrid EP algorithm proposed in the previous chapter is employed to identify the elastic parameters of laminated composite plates from their natural frequencies.

Chapter 4: Parameter Identification of Composite Plates

4.2 STRUCTURAL ANALYSIS

Several approximate methods have been proposed for free vibration analysis of angle-ply plates. Ashton and Anderson (1969) firstly employed the Rayleigh-Ritz method to obtain the natural frequencies of laminated rectangular plates using series of expansion of beam mode shape functions. Mohan and Kingsbury (1971) used the Galerkin method to study the vibration of laminated plates. Sivakumaran (1987) chose a set of complete power series as the admissible functions in the Rayleigh-Ritz method and obtained the approximate natural frequencies of symmetrically laminated rectangular plates. Liew (1990) proposed a set of 2-D polynomials as the shape functions in the Rayleigh-Ritz method to analyze the vibration of plates. The method was then extended to obtain the natural frequencies and mode shapes of symmetrically (Chow et al 1992) and unsymmetrically (Liew et al 1999) laminated plates. They claimed that polynomial shape function has certain advantages as it can be accurately integrated and differentiated. Moreover, it can be adapted for plates with different shapes and various boundary conditions (Liew et al 1998). Thus, in this study, the Ritz method with polynomial shape function is adopted for the vibration analysis of laminated composite plates.

4.2.1 Laminae

A lamina is the basic building block in a laminated fiber-reinforced composite. As defined by Jones (1975), “A lamina is a flat (sometimes curved as in a shell) arrangement of unidirectional fibers or woven fibers in a matrix”. A typical lamina is shown in Fig. 4.1, where directions 1-, 2- and 3- are the principal material directions with 1-axis parallel to the orientation of the fibers and 2-3 plane transversely isotropic.

The strain-stress relationship for transversely isotropic material (for a 2-3 symmetry plane) is given by (Jones 1975):

Chapter 4: Parameter Identification of Composite Plates

Fig. 4.1. A typical lamina

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{Bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{12} & 0 & 0 & 0 \\ S_{12} & S_{22} & S_{23} & 0 & 0 & 0 \\ S_{12} & S_{23} & S_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(S_{22} - S_{23}) & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{55} \end{bmatrix} \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{Bmatrix} \quad (4.1)$$

where $S_{11} = \frac{1}{E_1}$, $S_{22} = \frac{1}{E_2}$, $S_{12} = -\frac{\nu_{12}}{E_1}$, $S_{23} = -\frac{\nu_{23}}{E_2}$, and $S_{55} = \frac{1}{G_{13}}$; E_i , $i = 1, 2, 3$

is the Young's modulus in the i -direction; G_{ij} , $i, j = 1, 2, 3$ is the shear modulus in

the i - j plane; and ν_{ij} , $i, j = 1, 2, 3$ is the Poisson's ratio for the transverse strain in the

j -direction when stressed in the i -direction, $\frac{\nu_{ij}}{E_i} = \frac{\nu_{ji}}{E_j}$.

By neglecting the effect of transverse normal stress σ_3 in the lamina, Eqn.

(4.1) can be reduced to:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 \\ 0 & 0 & Q_{44} & 0 & 0 \\ 0 & 0 & 0 & Q_{55} & 0 \\ 0 & 0 & 0 & 0 & Q_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{Bmatrix} = \mathbf{Q} \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{Bmatrix} \quad (4.2)$$

Chapter 4: Parameter Identification of Composite Plates

The reduced stiffness matrix $\mathbf{Q}$ depends on the elastic properties:

$$Q_{11} = \frac{E_1}{1 - \nu_{12}\nu_{21}} \quad (4.3a)$$

$$Q_{12} = \frac{\nu_{12}E_2}{1 - \nu_{12}\nu_{21}} = \frac{\nu_{21}E_1}{1 - \nu_{12}\nu_{21}} = \frac{\nu_{12}E_2}{1 - \nu_{12}\nu_{12}E_2/E_1} \quad (4.3b)$$

$$Q_{22} = \frac{E_2}{1 - \nu_{12}\nu_{21}} \quad (4.3c)$$

$$Q_{44} = G_{23} \quad (4.3d)$$

$$Q_{55} = G_{13} = G_{12} \quad (4.3e)$$

$$Q_{66} = G_{12} \quad (4.3f)$$

For a lamina whose principal material directions do not coincide with the coordinate directions that are geometrically natural to the solution of the problem, a transformation process can be employed to obtain the constitutive equation in the global coordinate system xyz :

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{Bmatrix} = \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} & 0 & 0 & \bar{Q}_{16} \\ \bar{Q}_{12} & \bar{Q}_{22} & 0 & 0 & \bar{Q}_{26} \\ 0 & 0 & \bar{Q}_{44} & \bar{Q}_{45} & 0 \\ 0 & 0 & \bar{Q}_{45} & \bar{Q}_{55} & 0 \\ \bar{Q}_{16} & \bar{Q}_{26} & 0 & 0 & \bar{Q}_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix} = \bar{\mathbf{Q}} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix} \quad (4.4)$$

where the stiffness coefficients $\bar{Q}_{ij}$, $i, j = 1, 2, 4, 5, 6$ depend on the fiber orientation in the layer:

$$\bar{Q}_{11} = Q_{11} \cos^4 \theta + 2(Q_{12} + 2Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{22} \sin^4 \theta \quad (4.5a)$$

$$\bar{Q}_{12} = (Q_{11} + Q_{22} - 4Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{12} (\sin^4 \theta + \cos^4 \theta) \quad (4.5b)$$

$$\bar{Q}_{22} = Q_{11} \sin^4 \theta + 2(Q_{12} + 2Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{22} \cos^4 \theta \quad (4.5c)$$

$$\bar{Q}_{16} = (Q_{11} - Q_{12} - 2Q_{66}) \sin \theta \cos^3 \theta + (Q_{12} - Q_{22} + 2Q_{66}) \sin^3 \theta \cos \theta \quad (4.5d)$$

$$\bar{Q}_{26} = (Q_{11} - Q_{12} - 2Q_{66}) \sin^3 \theta \cos \theta + (Q_{12} - Q_{22} + 2Q_{66}) \sin \theta \cos^3 \theta \quad (4.5e)$$

$$\bar{Q}_{66} = (Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{66} (\sin^4 \theta + \cos^4 \theta) \quad (4.5f)$$

$$\bar{Q}_{44} = Q_{44} \cos^2 \theta + Q_{55} \sin^2 \theta \quad (4.5g)$$

Chapter 4: Parameter Identification of Composite Plates

$$\bar{Q}_{45} = (Q_{55} - Q_{44}) \cos \theta \sin \theta \quad (4.5h)$$

$$\bar{Q}_{55} = Q_{55} \cos^2 \theta + Q_{44} \sin^2 \theta \quad (4.5i)$$

where θ is the angle started from the global xy axes to the principal 12 axes, as shown in Fig. 4.1.

4.2.2 Laminates

A laminate is a stack of laminae with various orientations of principal material directions in the laminae. The layers of a laminate are usually bound together by the same matrix material that is used in the laminae. The principal material directions of each layer can be oriented differently so that the constructed laminates can satisfy the necessary strength and stiffness as a structural element.

A rectangular laminated composite plate of area axb and constant thickness h is considered in this study, as shown in Fig. 4.2. The x - and y - coordinates are taken in the middle surface of the plate.

Fig. 4.2 A laminated composite plate

Chapter 4: Parameter Identification of Composite Plates

Based on the Mindlin plate theory (Mindlin 1951), i.e., a straight line normal to the mid-plane of the laminate remains straight and inextensible after deformation, the displacement field in a laminated plate can be expressed as:

$$\bar{u} = u + z\psi_x \quad (4.6a)$$

$$\bar{v} = v + z\psi_y \quad (4.6b)$$

$$\bar{w} = w(x, y, t) \quad (4.6c)$$

where $\bar{u}$, $\bar{v}$ and $\bar{w}$ represent the displacements in the x -, y - and z -directions, respectively; u , v and w are the associated mid-plane displacements; and ψ_x and ψ_y are the rotation angles in the x - z and y - z planes, respectively, as defined in Fig. 4.3.

Fig. 4.3. A deformed Mindlin plate

The geometric equation can thus be expressed as:

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u}{\partial x} + z \frac{\partial \psi_x}{\partial x} \\ \frac{\partial v}{\partial y} + z \frac{\partial \psi_y}{\partial y} \\ \psi_y + \frac{\partial w}{\partial y} \\ \psi_x + \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + z \frac{\partial \psi_x}{\partial y} + z \frac{\partial \psi_y}{\partial x} \end{Bmatrix} \quad (4.7)$$

Chapter 4: Parameter Identification of Composite Plates

4.2.3 Energy functions

The strain energy functional U due to bending of the Mindlin plate is given by:

$$U = \frac{1}{2} \int \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} dV = \frac{1}{2} \int \boldsymbol{\varepsilon}^T \overline{\mathbf{Q}} \boldsymbol{\varepsilon} dV$$

where V is the plate volume.

The Mindlin plate theory yields a constant value of the transverse shear strain through the thickness of the plate. Thus, shear correction factors have to be introduced to account for the discrepancy between the constant state of shear strains in the Mindlin theory and the quadratic or higher-order distribution of shear strains in the elasticity theory (Ochoa and Reddy 1992). The potential energy of the laminated composite plate can then be calculated as:

$$\begin{aligned}
 U &= \frac{1}{2} \int \left\{ \begin{matrix} \varepsilon_x & \varepsilon_y & \gamma_{xy} \end{matrix} \right\} \begin{bmatrix} \overline{Q}_{11} & \overline{Q}_{12} & \overline{Q}_{16} \\ \overline{Q}_{12} & \overline{Q}_{22} & \overline{Q}_{26} \\ \overline{Q}_{16} & \overline{Q}_{26} & \overline{Q}_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \\
 &\quad + \left\{ \begin{matrix} \tau_{yz} & \tau_{xz} \end{matrix} \right\} \begin{bmatrix} \kappa_{44}^2 \overline{Q}_{44} & \kappa_{45}^2 \overline{Q}_{45} \\ \kappa_{45}^2 \overline{Q}_{45} & \kappa_{55}^2 \overline{Q}_{55} \end{bmatrix} \begin{Bmatrix} \gamma_{yz} \\ \gamma_{xz} \end{Bmatrix} \right\} dV \\
 &= \frac{1}{2} \int \left[\overline{Q}_{11} \left(\frac{\partial u}{\partial x} + z \frac{\partial \psi_x}{\partial x} \right)^2 + 2 \overline{Q}_{12} \left(\frac{\partial u}{\partial x} + z \frac{\partial \psi_x}{\partial x} \right) \left(\frac{\partial v}{\partial y} + z \frac{\partial \psi_y}{\partial y} \right) \right. \\
 &\quad + \overline{Q}_{22} \left(\frac{\partial v}{\partial y} + z \frac{\partial \psi_y}{\partial y} \right)^2 + 2 \overline{Q}_{16} \left(\frac{\partial u}{\partial x} + z \frac{\partial \psi_x}{\partial x} \right) \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + z \frac{\partial \psi_x}{\partial y} + z \frac{\partial \psi_y}{\partial x} \right) \\
 &\quad + 2 \overline{Q}_{26} \left(\frac{\partial v}{\partial y} + z \frac{\partial \psi_y}{\partial y} \right) \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + z \frac{\partial \psi_x}{\partial y} + z \frac{\partial \psi_y}{\partial x} \right) \\
 &\quad + \overline{Q}_{66} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + z \frac{\partial \psi_x}{\partial y} + z \frac{\partial \psi_y}{\partial x} \right)^2 + \kappa_{44}^2 \overline{Q}_{44} \left(\psi_y + \frac{\partial w}{\partial y} \right)^2 \\
 &\quad \left. + 2 \kappa_{45}^2 \overline{Q}_{45} \left(\psi_x + \frac{\partial w}{\partial x} \right) \left(\psi_y + \frac{\partial w}{\partial y} \right) + \kappa_{55}^2 \overline{Q}_{55} \left(\psi_x + \frac{\partial w}{\partial x} \right)^2 \right] dV \quad (4.9)
 \end{aligned}$$

Chapter 4: Parameter Identification of Composite Plates

where κ_{ij}^2 , $i, j = 4, 5$ is the shear correction factor. Different formulations have been proposed for the shear correction factor based on various simplifications. Mindlin (1951) recommended that κ^2 vary from 0.76 to 0.91 depending on the Poisson's ratio for an isotropic plate. Reissner (1945) indicated that $\kappa^2 = \frac{5}{6}$.

Wittrick (1987) proposed a formula as $\kappa^2 = \frac{5}{6 - \nu}$. For simplicity,

$\kappa_{ij}^2 = \frac{5}{6}$, $i, j = 4, 5$ is adopted in this study.

The integration of Eqn. (4.9) over the plate thickness yields:

$$\begin{aligned}
 U = \frac{1}{2} \int_A \left\{ A_{11} \left(\frac{\partial u}{\partial x} \right)^2 + 2B_{11} \frac{\partial u}{\partial x} \frac{\partial \psi_x}{\partial x} + D_{11} \left(\frac{\partial \psi_x}{\partial x} \right)^2 \right. \\
 + 2A_{12} \frac{\partial u}{\partial x} \frac{\partial v}{\partial y} + 2B_{12} \left(\frac{\partial u}{\partial x} \frac{\partial \psi_y}{\partial y} + \frac{\partial v}{\partial y} \frac{\partial \psi_x}{\partial x} \right) + 2D_{12} \frac{\partial \psi_x}{\partial x} \frac{\partial \psi_y}{\partial y} \\
 + A_{22} \left(\frac{\partial v}{\partial y} \right)^2 + 2B_{22} \frac{\partial v}{\partial y} \frac{\partial \psi_y}{\partial y} + D_{22} \left(\frac{\partial \psi_y}{\partial y} \right)^2 \\
 + 2A_{16} \frac{\partial u}{\partial x} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + 2B_{16} \left[\frac{\partial \psi_x}{\partial x} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + \frac{\partial u}{\partial x} \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right) \right] \\
 + 2D_{16} \frac{\partial \psi_x}{\partial x} \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right) + 2A_{26} \frac{\partial v}{\partial y} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\
 + 2B_{26} \left[\frac{\partial v}{\partial y} \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right) + \frac{\partial \psi_y}{\partial y} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] + 2D_{26} \frac{\partial \psi_y}{\partial y} \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right) \\
 + A_{66} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 + 2B_{66} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right) + D_{66} \left(\frac{\partial \psi_x}{\partial y} + \frac{\partial \psi_y}{\partial x} \right)^2 \\
 \left. + A_{44} \left(\psi_y + \frac{\partial w}{\partial y} \right)^2 + 2A_{45} \left(\psi_x + \frac{\partial w}{\partial x} \right) \left(\psi_y + \frac{\partial w}{\partial y} \right) + A_{55} \left(\psi_x + \frac{\partial w}{\partial x} \right)^2 \right\} dA
 \end{aligned} \tag{4.10}$$

where A is the plate area, $dA = dx dy$, and

$$\langle A_{ij}, B_{ij}, D_{ij} \rangle = \int_{-h/2}^{h/2} \langle 1, z, z^2 \rangle \bar{Q}_{ij} dz, \quad i, j = 1, 2, 6 \tag{4.11a}$$

Chapter 4: Parameter Identification of Composite Plates

$$A_{ij} = \kappa_{ij}^2 \int_{-h/2}^{h/2} \bar{Q}_{ij} dz, \quad i, j = 4, 5 \quad (4.11b)$$

The kinetic energy T of the vibrating Mindlin plate is given by:

$$\begin{aligned} T &= \frac{1}{2} \int \rho (\dot{u}^2 + \dot{v}^2 + \dot{w}^2) dV \\ &= -\frac{1}{2} \rho \omega^2 \int [(u + z\psi_x)^2 + (v + z\psi_y)^2 + w^2] dV \end{aligned} \quad (4.12)$$

where "." denotes the derivative of time and ω is the vibration frequency of the plate.

Without loss of generality, a transformation relationship is introduced:

$$x = \frac{a}{2}(\xi + 1) \quad (4.13a)$$

$$y = \frac{b}{2}(\eta + 1) \quad (4.13b)$$

so that the plate with various dimension in the x - y plane can be mapped to a unit square plate in the ξ - η plane. The energy functions, Eqns. (4.10) and (4.12) can be converted to:

$$\begin{aligned} U &= \frac{1}{2} \int_{-1}^1 \int_{-1}^1 \left[\frac{4A_{11}}{a^2} \left(\frac{\partial u}{\partial \xi} \right)^2 + \frac{8B_{11}}{a^2} \frac{\partial u}{\partial \xi} \frac{\partial \psi_x}{\partial \xi} + \frac{4D_{11}}{a^2} \left(\frac{\partial \psi_x}{\partial \xi} \right)^2 \right. \\ &\quad + \frac{8A_{12}}{ab} \frac{\partial u}{\partial \xi} \frac{\partial v}{\partial \eta} + \frac{8B_{12}}{ab} \left(\frac{\partial u}{\partial \xi} \frac{\partial \psi_y}{\partial \eta} + \frac{\partial v}{\partial \eta} \frac{\partial \psi_x}{\partial \xi} \right) + \frac{8D_{12}}{ab} \frac{\partial \psi_x}{\partial \xi} \frac{\partial \psi_y}{\partial \eta} \\ &\quad + \frac{4A_{22}}{b^2} \left(\frac{\partial v}{\partial \eta} \right)^2 + \frac{8B_{22}}{b^2} \frac{\partial v}{\partial \eta} \frac{\partial \psi_y}{\partial \eta} + \frac{4D_{22}}{b^2} \left(\frac{\partial \psi_y}{\partial \eta} \right)^2 \\ &\quad + \frac{4A_{16}}{a} \frac{\partial u}{\partial \xi} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right) \\ &\quad + 2B_{16} \left[\frac{2}{a} \frac{\partial \psi_x}{\partial \xi} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right) + \frac{2}{a} \frac{\partial u}{\partial \xi} \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right) \right] \\ &\quad \left. + \frac{4D_{16}}{a} \frac{\partial \psi_x}{\partial \xi} \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right) + \frac{4A_{26}}{b} \frac{\partial v}{\partial \eta} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right) \right] \end{aligned}$$

Chapter 4: Parameter **Identification** of Composite Plates

$$\begin{aligned}
 & + 2B_{26} \left[\frac{2}{b} \frac{\partial v}{\partial \eta} \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right) + \frac{2}{b} \frac{\partial \psi_y}{\partial \eta} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right) \right] \\
 & + \frac{4D_{26}}{b} \frac{\partial \psi_y}{\partial \eta} \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right) + A_{66} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right)^2 \\
 & + 2B_{66} \left(\frac{2}{b} \frac{\partial u}{\partial \eta} + \frac{2}{a} \frac{\partial v}{\partial \xi} \right) \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right) + D_{66} \left(\frac{2}{b} \frac{\partial \psi_x}{\partial \eta} + \frac{2}{a} \frac{\partial \psi_y}{\partial \xi} \right)^2 \\
 & + A_{44} \left(\psi_y + \frac{2}{b} \frac{\partial w}{\partial \eta} \right)^2 + 2A_{45} \left(\psi_x + \frac{2}{a} \frac{\partial w}{\partial \xi} \right) \left(\psi_y + \frac{2}{b} \frac{\partial w}{\partial \eta} \right) \\
 & + A_{55} \left(\psi_x + \frac{2}{a} \frac{\partial w}{\partial \xi} \right)^2 \left\} \frac{ab}{4} d\xi d\eta \quad (4.14)
 \end{aligned}$$

and

$$\begin{aligned}
 T &= -\frac{1}{2} \rho \omega^2 \int_V \left[(u + z \psi_x)^2 + (v + z \psi_y)^2 + w^2 \right] dV \\
 &= -\frac{1}{2} \rho \omega^2 \int_{-1}^1 \int_{-1}^1 \left[(u^2 + v^2 + w^2) h + (\psi_x^2 + \psi_y^2) \frac{h^3}{12} \right] \frac{ab}{4} d\xi d\eta \quad (4.15)
 \end{aligned}$$

4.2.4 Shape functions

The displacements are approximated as combinations of a set of polynomial shape functions:

$$u = \sum_{i=0}^{N_1} \sum_{l=0}^i a_m \phi_m^u = \sum_{m=0}^{P_1} a_m \phi_m^u = \mathbf{a}^T \boldsymbol{\Phi}^u \quad (4.16a)$$

$$v = \sum_{i=0}^{N_2} \sum_{l=0}^i b_m \phi_m^v = \sum_{m=0}^{P_2} b_m \phi_m^v = \mathbf{b}^T \boldsymbol{\Phi}^v \quad (4.16b)$$

$$w = \sum_{i=0}^{N_3} \sum_{l=0}^i c_m \phi_m^w = \sum_{m=0}^{P_3} c_m \phi_m^w = \mathbf{c}^T \boldsymbol{\Phi}^w \quad (4.16c)$$

$$\psi_x = \sum_{i=0}^{N_4} \sum_{l=0}^i d_m \phi_m^x = \sum_{m=0}^{P_4} d_m \phi_m^x = \mathbf{d}^T \boldsymbol{\Phi}^x \quad (4.16d)$$

$$\psi_y = \sum_{i=0}^{N_5} \sum_{l=0}^i e_m \phi_m^y = \sum_{m=0}^{P_5} e_m \phi_m^y = \mathbf{e}^T \boldsymbol{\Phi}^y \quad (4.16e)$$

Chapter 4: Parameter Identification of Composite Plates

where N_s , $s = 1, 2, 3, 4, 5$ are the degree of the mathematically complete two-dimensional polynomial space; a_m, b_m, c_m, d_m and e_m are the unknown coefficients; ϕ^k , $k = u, v, w, x, y$ are the shape function vectors; the subscript m is determined by:

$$m = \frac{i(i+1)}{2} + i1$$

and P_s , $s = 1, 2, 3, 4, 5$ are the total numbers of a_m, b_m, c_m, d_m , and e_m , which are dependent on the degree set of the polynomial space N_s , given by:

$$P_s = \frac{(N_s + 1)(N_s + 2)}{2}, s = 1, 2, 3, 4, 5 \quad (4.18)$$

Although N_s can be different for the displacements u, v, w, ψ_x and ψ_y , they are set the same in this study for convenience.

The 2-D polynomial shape functions are defined as (Liew 1990):

$$\phi_m^k = \xi^{i-i1} \eta^{j1} \phi_0^k, k = u, v, w, x, y \quad (4.19)$$

and the basic functions are given as:

$$\phi_0^k = \prod_p [\Gamma_p(\xi, \eta)]^{\mathcal{G}_p^k} \quad (4.20)$$

where Γ_p is the boundary equation of the p -th supporting edge in the $\xi - \eta$ plane and $\mathcal{G}_p$ is employed to ensure that the shape functions can satisfy at least the geometric boundary conditions. The common edge conditions for the Mindlin plate are given as follows:

$$\text{Free edge (F): } M_{nn} = M_{ns} = Q_n = 0 \quad (4.21a)$$

$$\text{Hard type simply supported edge (S): } M_{nn} = \psi_s = w = 0 \quad (4.21b)$$

$$\text{Soft type simply supported edge (S*): } M_{nn} = M_{ns} = w = 0 \quad (4.21c)$$

$$\text{Clamped edge (C): } \psi_n = \psi_s = w = 0 \quad (4.21d)$$

where the subscripts n and s represent the normal and tangential directions of the edge, respectively; M_{nn} and M_{ns} are the moments in the n - z and n - s plane, respectively; and Q , is the shear force in the z -direction.

Chapter 4: Parameter Identification of Composite Plates

With the polynomial shape functions, the hard type simply supported boundary condition (S) cannot be implemented if the edge is not parallel or perpendicular to the coordinial axes. However, this drawback can be remedied by introducing point restraints to ensure the rotation along the edge be zero $\psi_s = 0$ (Liew et al 1998). For the other kinds of boundary conditions, the values of $\mathcal{G}_p$ for various boundary conditions are listed in Table 4.1.

Table 4.1. Values of $\mathcal{G}_p$ in Eqn. (4.20) for different edge boundary conditions

Supporting condition for p -th edge	$\mathcal{G}_p$				
	ϕ_0^u	ϕ_0^v	ϕ_0^w	ϕ_0^x	ϕ_0^y
Clamped (C)	1	1	1	1	1
Free (F)	0	0	0	0	0
Simply supported (S*)	0	0	1	0	0
Simply supported (S) with edge parallel to ξ axis	0	1	1	1	0
Simply supported (S) with edge parallel to η axis	1	0	1	0	1

4.2.5 Eigenvalue equation

The governing eigenvalue equation can be derived by minimizing the energy functional $\Pi = U + T$, which is obtained by setting the first derivatives of the energy functional to the unknown coefficients as zero:

$$\frac{\partial \Pi}{\partial a_m} = \frac{\partial \Pi}{\partial b_m} = \frac{\partial \Pi}{\partial c_m} = \frac{\partial \Pi}{\partial d_m} = \frac{\partial \Pi}{\partial e_m} = 0 \tag{4.22}$$

Thus, the eigenvalue equation is formulated as:

$$(\mathbf{K} - \mathbf{M}\omega^2)\mathbf{q} = \mathbf{0} \tag{4.23}$$

where $\mathbf{q}$ is the vector of unknown coefficients, $\mathbf{K}$ and $\mathbf{M}$ are the stiffness and mass matrices, respectively.

$$\mathbf{q}^T = \{\mathbf{a}^T \quad \mathbf{b}^T \quad \mathbf{c}^T \quad \mathbf{d}^T \quad \mathbf{e}^T\} \tag{4.24}$$

Chapter 4: Parameter Identification of Composite Plates

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^{aa} & \mathbf{K}^{ab} & \mathbf{K}^{ac} & \mathbf{K}^{ad} & \mathbf{K}^{ae} \\ & \mathbf{K}^{bb} & \mathbf{K}^{bc} & \mathbf{K}^{bd} & \mathbf{K}^{be} \\ & & \mathbf{K}^{cc} & \mathbf{K}^{cd} & \mathbf{K}^{ce} \\ & \text{Sym} & & \mathbf{K}^{dd} & \mathbf{K}^{de} \\ & & & & \mathbf{K}^{ee} \end{bmatrix} \quad (4.25)$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^{aa} & \mathbf{M}^{ab} & \mathbf{M}^{ac} & \mathbf{M}^{ad} & \mathbf{M}^{ae} \\ & \mathbf{M}^{bb} & \mathbf{M}^{bc} & \mathbf{M}^{bd} & \mathbf{M}^{be} \\ & & \mathbf{M}^{cc} & \mathbf{M}^{cd} & \mathbf{M}^{ce} \\ & \text{Sym} & & \mathbf{M}^{dd} & \mathbf{M}^{de} \\ & & & & \mathbf{M}^{ee} \end{bmatrix} \quad (4.26)$$

The components of the stiffness and mass matrices can be given as:

$$K_{mn}^{aa} = \frac{\partial^2 U}{\partial a_m \partial a_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{11} \frac{4}{a^2} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^u}{\partial \xi} + A_{16} \frac{4}{ab} \left(\frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^u}{\partial \eta} + \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^u}{\partial \xi} \right) + A_{66} \frac{4}{b^2} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^u}{\partial \eta} \right] \frac{ab}{4} d\xi d\eta \quad (4.27a)$$

$$K_{mn}^{ab} = \frac{\partial^2 U}{\partial a_m \partial b_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{12} \frac{4}{ab} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^v}{\partial \eta} + A_{16} \frac{4}{a^2} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^v}{\partial \xi} + A_{26} \frac{4}{b^2} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^v}{\partial \eta} + A_{66} \frac{4}{ab} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^v}{\partial \xi} \right] \frac{ab}{4} d\xi d\eta \quad (4.27b)$$

$$K_{mn}^{ac} = \frac{\partial^2 U}{\partial a_m \partial c_n} = 0 \quad (4.27c)$$

$$K_{mn}^{ad} = \frac{\partial^2 U}{\partial a_m \partial d_n} = \int_{-1}^1 \int_{-1}^1 \left[B_{11} \frac{4}{a^2} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^x}{\partial \xi} + B_{16} \frac{4}{ab} \left(\frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^x}{\partial \xi} + \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^x}{\partial \eta} \right) + B_{66} \frac{4}{b^2} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^x}{\partial \eta} \right] \frac{ab}{4} d\xi d\eta \quad (4.27d)$$

$$K_{mn}^{ae} = \frac{\partial^2 U}{\partial a_m \partial e_n} = \int_{-1}^1 \int_{-1}^1 \left[B_{12} \frac{4}{ab} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^y}{\partial \eta} + B_{16} \frac{4}{a^2} \frac{\partial \phi_m^u}{\partial \xi} \frac{\partial \phi_n^y}{\partial \xi} + B_{26} \frac{4}{b^2} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^y}{\partial \eta} + B_{66} \frac{4}{ab} \frac{\partial \phi_m^u}{\partial \eta} \frac{\partial \phi_n^y}{\partial \xi} \right] \frac{ab}{4} d\xi d\eta \quad (4.27e)$$

Chapter 4: Parameter Identification of Composite Plates

$$K_{mn}^{bb} = \frac{\partial^2 U}{\partial b_m \partial b_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{22} \frac{4}{b^2} \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^v}{\partial \eta} + A_{26} \frac{4}{ab} \left(\frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^v}{\partial \eta} + \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^v}{\partial \xi} \right) + A_{66} \frac{4}{a^2} \frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^v}{\partial \xi} \right] \frac{ab}{4} d\xi d\eta \quad (4.27f)$$

$$K_{mn}^{bc} = \frac{\partial^2 U}{\partial b_m \partial c_n} = 0 \quad (4.27g)$$

$$K_{mn}^{bd} = \frac{\partial^2 U}{\partial b_m \partial d_n} = \int_{-1}^1 \int_{-1}^1 \left[B_{12} \frac{4}{ab} \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^x}{\partial \xi} + B_{16} \frac{4}{a^2} \frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^x}{\partial \xi} + B_{26} \frac{4}{b^2} \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^x}{\partial \eta} + B_{66} \frac{4}{ab} \frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^x}{\partial \eta} \right] \frac{ab}{4} d\xi d\eta \quad (4.27h)$$

$$K_{mn}^{be} = \frac{\partial^2 U}{\partial b_m \partial e_n} = \int_{-1}^1 \int_{-1}^1 \left[B_{22} \frac{4}{b^2} \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^y}{\partial \eta} + B_{26} \frac{4}{ab} \left(\frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^y}{\partial \eta} + \frac{\partial \phi_m^v}{\partial \eta} \frac{\partial \phi_n^y}{\partial \xi} \right) + B_{66} \frac{4}{a^2} \frac{\partial \phi_m^v}{\partial \xi} \frac{\partial \phi_n^y}{\partial \xi} \right] \frac{ab}{4} d\xi d\eta \quad (4.27i)$$

$$K_{mn}^{cc} = \frac{\partial^2 U}{\partial c_m \partial c_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{44} \frac{4}{b^2} \frac{\partial \phi_m^w}{\partial \eta} \frac{\partial \phi_n^w}{\partial \eta} + A_{45} \frac{4}{ab} \left(\frac{\partial \phi_m^w}{\partial \xi} \frac{\partial \phi_n^w}{\partial \eta} + \frac{\partial \phi_m^w}{\partial \eta} \frac{\partial \phi_n^w}{\partial \xi} \right) + A_{55} \frac{4}{a^2} \frac{\partial \phi_m^w}{\partial \xi} \frac{\partial \phi_n^w}{\partial \xi} \right] \frac{ab}{4} d\xi d\eta \quad (4.27j)$$

$$K_{mn}^{cd} = \frac{\partial^2 U}{\partial c_m \partial d_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{45} \frac{2}{b} \frac{\partial \phi_m^w}{\partial \eta} \phi_n^x + A_{55} \frac{2}{a} \frac{\partial \phi_m^w}{\partial \xi} \phi_n^x \right] \frac{ab}{4} d\xi d\eta \quad (4.27k)$$

$$K_{mn}^{ce} = \frac{\partial^2 U}{\partial c_m \partial e_n} = \int_{-1}^1 \int_{-1}^1 \left[A_{44} \frac{2}{b} \frac{\partial \phi_m^w}{\partial \eta} \phi_n^y + A_{45} \frac{2}{a} \frac{\partial \phi_m^w}{\partial \xi} \phi_n^y \right] \frac{ab}{4} d\xi d\eta \quad (4.27l)$$

$$K_{mn}^{dd} = \frac{\partial^2 U}{\partial d_m \partial d_n} = \int_{-1}^1 \int_{-1}^1 \left[D_{11} \frac{4}{a^2} \frac{\partial \phi_m^x}{\partial \xi} \frac{\partial \phi_n^x}{\partial \xi} + D_{16} \frac{4}{ab} \left(\frac{\partial \phi_m^x}{\partial \xi} \frac{\partial \phi_n^x}{\partial \eta} + \frac{\partial \phi_m^x}{\partial \eta} \frac{\partial \phi_n^x}{\partial \xi} \right) + D_{66} \frac{4}{b^2} \frac{\partial \phi_m^x}{\partial \eta} \frac{\partial \phi_n^x}{\partial \eta} + A_{55} \phi_m^x \phi_n^x \right] \frac{ab}{4} d\xi d\eta \quad (4.27m)$$

$$K_{mn}^{de} = \frac{\partial^2 U}{\partial d_m \partial e_n} = \int_{-1}^1 \int_{-1}^1 \left[D_{12} \frac{4}{ab} \frac{\partial \phi_m^x}{\partial \xi} \frac{\partial \phi_n^y}{\partial \eta} + D_{16} \frac{4}{a^2} \frac{\partial \phi_m^x}{\partial \xi} \frac{\partial \phi_n^y}{\partial \xi} + D_{26} \frac{4}{b^2} \frac{\partial \phi_m^x}{\partial \eta} \frac{\partial \phi_n^y}{\partial \eta} \right]$$

Chapter 4: Parameter Identification of Composite Plates

$$+ D_{66} \frac{4}{ab} \frac{\partial \phi_m^x}{\partial \eta} \frac{\partial \phi_n^y}{\partial \xi} + A_{45} \phi_m^x \phi_n^y \left] \frac{ab}{4} d\xi d\eta \quad (4.27n)$$

$$K_{mn}^{ee} = \frac{\partial^2 U}{\partial d_m \partial d_n} = \int_{-1}^1 \int_{-1}^1 \left[D_{22} \frac{4}{b^2} \frac{\partial \phi_m^y}{\partial \eta} \frac{\partial \phi_n^y}{\partial \eta} + D_{26} \frac{4}{ab} \left(\frac{\partial \phi_m^y}{\partial \xi} \frac{\partial \phi_n^y}{\partial \eta} + \frac{\partial \phi_m^y}{\partial \eta} \frac{\partial \phi_n^y}{\partial \xi} \right) \right. \\ \left. + D_{66} \frac{4}{a^2} \frac{\partial \phi_m^y}{\partial \xi} \frac{\partial \phi_n^y}{\partial \xi} + A_{44} \phi_m^y \phi_n^y \right] \frac{ab}{4} d\xi d\eta \quad (4.27o)$$

and

$$M_{mn}^{aa} = -\frac{1}{\omega^2} \frac{\partial^2 T}{\partial a_m \partial a_n} = \rho h \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \phi_m^u \phi_n^u d\xi d\eta \quad (4.28a)$$

$$M_{mn}^{bb} = -\frac{1}{\omega^2} \frac{\partial^2 T}{\partial b_m \partial b_n} = \rho h \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \phi_m^v \phi_n^v d\xi d\eta \quad (4.28b)$$

$$M_{mn}^{cc} = -\frac{1}{\omega^2} \frac{\partial^2 T}{\partial c_m \partial c_n} = \rho h \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \phi_m^w \phi_n^w d\xi d\eta \quad (4.28c)$$

$$M_{mn}^{dd} = -\frac{1}{\omega^2} \frac{\partial^2 T}{\partial d_m \partial d_n} = \rho \frac{h^3}{12} \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \phi_m^x \phi_n^x d\xi d\eta \quad (4.28d)$$

$$M_{mn}^{ee} = -\frac{1}{\omega^2} \frac{\partial^2 T}{\partial e_m \partial e_n} = \rho \frac{h^3}{12} \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \phi_m^y \phi_n^y d\xi d\eta \quad (4.28e)$$

$$M_{mn}^{ab} = M_{mn}^{ac} = M_{mn}^{ad} = M_{mn}^{ae} = M_{mn}^{bc} = M_{mn}^{bd} = M_{mn}^{be} = M_{mn}^{cd} = M_{mn}^{ce} = M_{mn}^{de} = 0 \quad (4.28f)$$

where $m, n = 0, 1, 2, \dots, \frac{(N+1)(N+2)}{2} - 1$.

With these, the eigenvalue equation is fully defined and the natural frequencies of a laminated composite plate can be predicted by solving Eqn. (4.23).

4.3 PARAMETER IDENTIFICATION

The problem of parameter identification of a composite laminated plate based on natural frequencies can be formulated as a minimization problem. Mathematically, it can be stated as:

Chapter 4: Parameter Identification of Composite Plates

$$\text{Minimize } f(\mathbf{X}) = \sum_i (\Omega_i - \omega_i(\mathbf{X}))^2 \quad (4.29)$$

where Ω and ω represent the measured and the predicted natural frequencies, respectively; and $\mathbf{X}$ is a vector of the unknown parameters to be determined.

As observed in Eqn. (4.3), the elastic property of a composite plate can be determined by five parameters:

$$\mathbf{X} = (E_1, E_2, G_{12}, G_{23}, \nu_{12}) \quad (4.30)$$

There are some restrictions on the elastic constants (Jones 1975):

$$-1 < \nu < 0.5 \text{ and generally } \nu > 0 \quad (4.31a)$$

$$E_1, E_2, G_{12}, G_{23} > 0 \quad (4.31b)$$

$$\text{and } E_1 > E_2, G_{12}, G_{23} \quad (4.31c)$$

The upper bound for the Young's moduli E_1, E_2 and the shear moduli G_{12}, G_{23} are unknown. In the process of parameter identification, previous researchers (Rikards et al 1999, Wang and Kam 2001) selected the boundary of the parameters based on experience. However, a robust nondestructive evaluation technique should try to reduce the effect of human factors as much as possible. To overcome this problem, the Young's moduli and shear moduli in Eqn. (4.30) are divided by E_1 . The first variable $\frac{E_1}{E_1} = 1 = \text{const}$ is discarded from the objective

variables. Thus, only four variables are contained in the objective vector:

$$\mathbf{X} = \left(\frac{E_2}{E_1}, \frac{G_{12}}{E_1}, \frac{G_{23}}{E_1}, \nu_{12} \right) \quad (4.32)$$

The restriction for the four variables can be determined by:

$$0 < \frac{E_2}{E_1}, \frac{G_{12}}{E_1}, \frac{G_{23}}{E_1} < 1 \quad (4.33a)$$

and

$$0 < \nu_{12} < 0.5 \quad (4.33b)$$

Chapter 4: Parameter Identification of Composite Plates

After the replacement of the Young's moduli E_1 , E_2 and the shear moduli G_{12} , G_{23} with their ratios to E_1 , the mass matrix $\mathbf{M}$ remains unchanged, but the stiffness matrix $\mathbf{K}$ converts to $\mathbf{K}'$. The eigenvalue equation Eqn. (4.23) is hence modified as:

$$(\mathbf{K}' - \mathbf{M}\omega'^2)\mathbf{q} = 0 \tag{4.34}$$

However, from Eqns, (4.3), (4.5), (4.11) and (4.27), it is noticed that

$$\mathbf{K} = E_1 \mathbf{K}' \tag{4.35}$$

Thus, the following relationship can be obtained:

$$\omega' = \frac{1}{\sqrt{E_1}} \omega \tag{4.36}$$

where ω' and ω are the vectors containing the eigenvalues of Eqn. (4.23) and Eqn. (4.34), respectively.

In this study, the eigenvalue equation, Eqn. (4.34), is solved using Matlab. The Young's modulus E_1 is determined by comparing the solved first nonzero eigenvalue with the measured first natural frequency. All of the other eigenvalues are consequently adjusted based on the calculated E_1 . The adjusted eigenvalues are then compared with the other measured natural frequencies to calculate the value of the fitness function Eqn. (4.29).

Table 4.2, Composite plate specimen data

Specimen	SP1	SP2	SP3	SP4	SP5
Material	Aluminium	Glass/epoxy	Glass/epoxy	Glass/epoxy	Carbon/epoxy
a(mm)	193	268	209	180	260
b(mm)	281	150	192	119.9	137
h(mm)	1.94	1.43	2.59	10.13	2.34
$\rho(\text{kg/m}^3)$	2688	1932	2120	1852	1586
No of plies of equal thickness	1	16	16	-	16
Fiber orientation	-	$[0_{16}]$	$[0,-40,40,90,40,0,90,-40]_s$	$[0]$	$[0_{16}]$

Chapter 4: Parameter Identification of Composite Plates

Table 4.3. Measured natural frequencies

Specimens		SP1	SP2	SP3	SP4	SP5
Modes						
Natural Frequencies (Hz)	Mode 1	112.6	72.3	172.5	757.8	136.7
	Mode 2	127.9	97.6	250.2	1484	312.5
	Mode 3	267.9	176.7	300.6	1804	329.1
	Mode 4	286.6	197.4	437.9	2107	419.7
	Mode 5	334.5	238.3	443.6	2290	431.8
	Mode 6	392.1	277.5	760.3	3703	707.8
	Mode 7	493.8	348.0	766.2	3960	877.0
	Mode 8	560.1	-	797.4	4422	910.6
	Mode 9	720.7	-	872.6	4696	963.4
	Mode 10	-	-	963	5130	1015.1
	Mode 11	-	-	-	5951	-
	Mode 12	-	-	-	6332	-
	Mode 13	-	-	-	7240	-
	Mode 14	-	-	-	7674	-

Five different plate specimens are selected from the available literatures (Rikards et al 1999, Soares et al 1993) for study. The specimen data and measured natural frequencies are quoted from the references and listed in Tables 4.2 and 4.3, respectively.

The technique described above is applied to the five specimens to identify the elastic parameters. Fig. 4.4 illustrates the detailed evolutionary process for Specimen 1 as an illustration example. After 100 generations, the optimized parameter values for the five specimens are presented in Table 4.4. The results obtained by other methods are also listed in Table 4.4 for direct comparison.

Chapter 4: Parameter Identification of Composite Plates

Fig. 4.4 Evolutionary process for SP1
(a) Fitness (b) E_1, E_2, G_{12} and G_{23} (c) ν_{12}

Chapter 4: Parameter Identification of Composite Plates

It is observed that the identified parameter values for Young's moduli E_1 and E_2 , plane shear modulus G_{12} and Poisson's ratio ν_{12} agree well with the solutions obtained by other methods. The exception is the transverse shear modulus G_{23} . The reason for the discrepancies is that for the cases of very thin plates, i.e. SP 1, 2, 3, and 5, in which the side/thickness ratio is about 100, the fitness function Eqn. (4.29) is rather insensitive to the transverse shear modulus. For specimen SP4, which is relatively thicker, the identified value is more acceptable. Rikards et al (1999) recommended that the transverse shear modulus "can be safely determined only from the experiments with thick plate".

Table 4.4. Identified elastic parameters after 100 generation

Specimen	Method	E_1 (GPa)	E_2 (GPa)	G_{12} (GPa)	G_{23} (GPa)	ν_{12}
SP1	Pedersen (1989)	69.5	67.8	24.5	-	0.34
	Soares et al (1993)	68.7	68.1	24.6	26.9	0.34
	Rikards et al (1999)	71.3	68.8	24.5	-	0.32
	This study	69.5	67.9	24.5	13.74	0.34
SP2	Sol (1986)	45.6	16.9	6.9	-	0.22
	Frederiksen (1992)	45.4	17.3	7.3	-	0.255
	Rikards et al (1999)	44.2	17.7	7.2	4.9	0.195
	This study	43.5	16.9	7.25	1.82	0.36
SP3	Pedersen (1989)	61.3	21.4	9.8	-	0.28
	Soares et al (1993)	57.2	21.4	11.3	11.8	0.3
	Rikards et al (1999)	56.5	20.8	11.8	29.83	0.349
	This study	59.0	19.0	11.5	1.7E-4	0.37
SP4	Frederiksen (1992)	42.4	11.6	4.68	4.55	0.305
	Tension test (Frederiksen, 1997)	43.4	10.2	-	-	0.32
	Rikards et al (1999)	42.7	11.6	4.69	6.07	0.324
	This study	42.1	11.6	4.67	4.03	0.3
SP5	Pedersen (1989)	127.0	10.6	6.06	-	0.29
	Soares et al (1993)	126.3	10.5	6.1	6.5	0.24
	This study	127.5	10.6	6.08	12.8	0.22

Chapter 4: Parameter Identification of Composite Plates

Besides the smaller side/thicknessratio for SP4, it is also noticed that higher vibration modes have been measured for parameter identification. Since the shear modulus has more contribution to the higher vibration modes, this could be another reason for the better solution for SP4. Although higher mode natural frequencies are difficult to measure, it could be viewed as a complementary technique to determine the transverse shear modulus of thin plates.

Table 4.5. Simulated natural frequenciesbased on the optimized parameters

Specimens		SP1	SP2	SP3	SP4	SP5
Modes						
Natural Frequencies (Hz)	Mode 1	112.4	72.2	172.0	757.6	136.5
	Mode 2	127.3	97.1	249.8	1480.9	315.8
	Mode 3	263.4	176.5	300.7	1802.8	334.6
	Mode 4	285.9	197.4	439.7	2102.7	420.4
	Mode 5	334.1	238.7	441.0	2290.5	429.1
	Mode 6	391.8	277.6	760.0	3698.0	720.6
	Mode 7	496.1	347.6	765.9	3958.0	877.0
	Mode 8	559.2	-	802.3	4416.3	912.9
	Mode 9	718.8	-	874.4	4685.9	958.6
	Mode 10	-	-	960.8	5130.6	1003.7
	Mode 11	-	-	-	5933.5	-
	Mode 12	-	-	-	6340.2	-
	Mode 13	-	-	-	7242.2	-
	Mode 14	-	-	-	7670.9	-

The simulated natural frequencies based on the optimized elastic parameters are listed in Table 4.5 as a comparison with the measured results. Good match between the two sets of data has demonstrated the correctness of the optimized results. However, there are still discrepancies between them. This is because the simulation is based on the laminated theory of composite plate which has a few assumptions, such as laminates are perfectly bonded, a straight line normal to the mid-plane of the laminate remains straight and inextensible after deformation, etc.

Chapter 4: Parameter Identification of Composite Plates

The fact that the totally free boundary condition assumed in the simulation cannot be satisfied in the experiments has also increased the discrepancies.

4.4 SUMMARY

The hybrid EP algorithm is employed as a search and optimization technique to identify the elastic constants of laminated composite plates. A special process is proposed to eliminate the influence of human factors on the identification results. The evolutionary process can converge within 50 generations. Good agreement is observed between the identified major elastic properties and those given in the available literatures. Some discrepancies have been found in the identification of the transverse shear modulus G_{23} . Causes of the discrepancies are analyzed and some techniques to overcome this problem are discussed.

The successful application of EP for structural parameter identification shows EP's potential for solving other inverse problems in civil engineering. In the following chapters, the application is extended to structural damage detection.

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

CHAPTER 5 INTEGRATED EP AND IMPEDANCE-BASED TECHNIQUE FOR DAMAGE DETECTION

5.1 INTRODUCTION

Most structures are damaged to a certain extent at some phase of their operational life. The damage may be due to creep, fatigue, impact, crack growth, delamination, corrosion, or other causes. The damages can result in the failure of the structural components, including fracture in tension or instability under compression. Consequently, the structural system's integrity and safety can be jeopardized. For proper functioning of the system, the damage should be located and rectified if possible.

The interest in the ability to monitor a structure and detect damage at the earliest possible stage is pervasive throughout the civil, mechanical and aerospace engineering communities. Various nondestructive evaluation (NDE) methods have been developed for different applications. Yoseph (1999) analyzed the NDE methods for in-service aerospace structures. Goch et al(1999) gave a survey of the state-of-the-art of NDE measuring methods for the assessment of surface integrity. Zou et al (2000) and Farraret al (2001) presented comprehensive literature reviews about vibration-based structural damage identification. McCann and Forde (2001) summarized NDE methods in the assessment of concrete and masonry structures.

Chapter 5: Integrated EP and Impedance-Based Techniques for Damage Detection

5.2 TRADITIONAL NDE TECHNIQUES

5.2.1 Visual inspection

Visual inspection technology has been the most popular damage detection method for centuries. It is the oldest and most common form of NDE technique used to detect surface defects. Visual inspection can be done in many ways either directly or remotely, by using borescopes, fiberscopes or video cameras. Although visual inspection is speedy, inexpensive and can be easily carried out to detect various types of surface defects, its reliability depends heavily upon the ability and experience of the inspectors. The inspectors must know how to search for critical flaws and how to recognize areas where failure could occur. Moreover, the surface to be inspected must be relatively clean. Cleaning the surface ranges from wiping with a cloth to blast cleaning and treatment with chemicals to show the surface details. The main disadvantage of visual inspection is that it can only detect the damages at the surfaces that are accessible to either the naked eye or to an optical aid.

5.2.2 Eddy current methods

Eddy current techniques offer low-cost methods for high speed, large scale inspection of conductive materials. Eddy current methods can be used for both quantitative measurements of material properties and sorting of objects by dimensions, properties or presence of discontinuities.

Eddy current methods for nondestructive, non-contact evaluation of electrically conducting materials rely on the principles of magnetic induction. When a coil excited by an alternating current is brought in close proximity to a conducting material, eddy currents are induced in the specimen. These currents set up a magnetic field that tends to oppose the original magnetic field. The impedance

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

the coil in close proximity to the specimen is affected by the presence of the induced eddy currents in the specimen. When the eddy currents in the specimen are distorted by the presence of flaws or material variations, the impedance in the coil is altered. The changes in magnitude, time lag, phase angle and flow patterns of eddy currents within the material under test are detected by measuring the resultant magnetic field with another set of sensing coils or solid state magnetic field detectors, and displayed in a manner that indicates the types of flaw or material condition.

The disadvantage of eddy current methods is that they require large amounts of power and the produced data are complicated to interpret, and the analysis involves solutions of elaborate inverse problem to deduce the presence of damage. Moreover, these techniques can only be applied to electrically conductive materials thus they are only commonly used in the aerospace, automotive, marine and manufacturing industries.

5.2.3 Radiography

Radiographic inspection is primarily used to find sub-surface flaws in materials. This technique chooses a radiation source, such as high-voltage X-ray machine to produce X-rays or radioactive isotopes to produce gamma rays, and places it close to the material to be inspected. The radiation passes through the material and is then captured either on film or digitally. The permeability or density of the structure is indicated by the magnitude of the passed radiation.

Although these techniques are relatively simple to implement and interpret, they require large and costly equipment that is difficult to use on large structural components without removing them from the site. In addition, these techniques require both sides of the structure to be accessible to emit and collect the radiation rays, which is sometimes not practical. Finally, X-rays and gamma rays are very hazardous. Special precautions must be taken when performing radiography. Therefore the method is undertaken under controlled conditions, inside a protective

enclosure or after assessment with appropriate barriers and warning systems to ensure that there are no hazards to personnel.

5.2.4 Sonic/Ultrasonic methods

Nondestructive sonic and ultrasonic testing methods are non-invasive and have been used for the past thirty years in the assessment of civil engineering structures and materials. These techniques refer to the transmission and reflection of mechanical stress waves through a medium at sonic and ultrasonic frequencies. Equipment to initiate the waves ranges from a simple force hammer to an ultrasonic angular insonification. Transducers are used to collect the transmitted or reflected waves. The traveling time, frequency and energy of the collected signals can all be used to detect cracks, voids or other damages that are inside the structures. The mechanism is similar to a doctor listening to a patient's heart and lungs during health examination.

The frequency of the initiated wave depends on the required depth of penetration and the vertical and lateral resolution. For example, the smallest vertical dimension that can be detected is normally $\lambda/4$, with λ the dominant wavelength. In the case of an impulse radar survey, the shallowest target that can be identified below the surface of a structure is $\lambda/3$ and within a structure the minimum size of target is a value of $\lambda/2$ (McCann and Forde, 2001).

If ultrasonic waves are employed, good coupling of the transducer to the surface is required. Typically liquid is used as a couplant, which affects mostly field applications. Consequently, various fixtures have been developed, e.g. a dripless bubbler (Patton and Hsu, 1998). Alternative dry coupling methods were also developed including the use of air-coupled piezoelectric transducers, electromagnetic transducers and laser ultrasonics (Green 1997).

Chapter 5: Integrated EP and impedance-Based Technique for Damage Detection

Although sonic and ultrasonic testing is a non-invasive and continuous monitoring method for in-service structures, such technique needs additional equipment to initialize the propagating waves. The existence of multiple numbers of travel paths from the source to the sensor can make signal identification difficult (Bray and McBride 1992). In addition, noises may arise from friction (including bearing wear), crack growth, turbulence (including leakage) and material changes such as corrosion. Furthermore, because the interpretation of the results is done on-line according to the echo signals appearing on the device monitor, this work needs cumulative experience, scientific and technical knowledge and intuition in order to accurately carry out the scanning procedure to obtain end-up with reliable scanning results (Abd and Farag 2002).

5.2.5 Thermography

Thermography is a non-contact optical method to detect damages in the structures. This technique uses specialized scanning camera to convert the temperature distribution on the structural surfaces into thermal images. It is observed that structures with defects emit differing amounts of infrared radiation. If there are any cracks or delaminations in the structure, the surface will heat up faster under solar irradiation in these areas and hot spots will be observed in the thermal record. The most attractive feature of thermography is the capability to cover large areas in a single frame. This method has been proven to be most effective as a reconnaissance tool for the rapid assessment of large buildings particularly high-rise apartment blocks (Stanley and Balendran 1995). Unfortunately, this method is found unreliable when testing bonded joints with a narrow gap between the unbonded surfaces.

Recent advances in infrared technology, specifically the development of high-density imaging sensors, made thermographic NDE technique more sensitive and effective than before. Transient thermography uses quartz lamps or hot-air heating to apply a short energy pulse to the structure, and measures the gradient of the thermal maps so that the flaws can be identified due to their different thermal

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

conductivity and diffusivity. This technique can rapidly (within a few seconds) make quantitative measurements of less than 1% material loss for various regions in the image (Favro and Thomas 1995; Han et al 1996).

5.2.6 Vibration-based methods

Damage detection by the changes of dynamic properties or response of structures is a subject that has received considerable attention (Shen and Taylor 1991; Liang et al 1992; Vestroni and Capecchi 1996, 2000). The basic idea of vibration-based NDE methods is that modal parameters (notably frequencies, mode shapes, and modal damping) are functions of the physical properties of the structure (mass, damping, and stiffness). Therefore, changes in the physical properties will cause changes in the modal properties. In these techniques, the effect of damage is simulated by changing the stiffness, mass and damping of the structural model in the damaged zones. The modal parameters can be predicted based on the damaged model. These predicted data are compared with the experimentally measured results. A system identification technique is employed to minimize the difference between them, so that the location and quantity of the damage can be determined.

This group of methods uses the lower frequencies of the vibration modes and can best describe the global behavior of the structure. Therefore, they hold promise for global nondestructive inspection of a variety of structures, because surface measurements of a vibrating structure can provide information about the health of the internal members. Moreover, it is not necessary to dismantle the member to be inspected from the structure, hence the techniques can be applied to in service structures.

The effectiveness of the whole group of vibration-based global NDE techniques depends heavily on the extend of the measurement errors and modeling errors. Usually 5% measurement errors can be expected, and this severely hinders damage identification (Juneja et al 1997). An important issue in these methods is their ability to discriminate between the model/data discrepancies caused by

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

modeling errors and the discrepancies that are a result of structural damage (Doebeling et al. 1996). Moreover, measurement of the vibration modal parameters is sometimes complicated and difficult, e.g., it is usually difficult to obtain the obvious and accurate mode shapes on some stiff civil structures (Wang et al 2001).

5.2.7 Other methods

With the same principle of vibration-based NDE methods, structural damage can also be detected by static measurements (Sanayei and Scampoli 1991, Sanayei and Onipede 1991). Compared with dynamic testing, the equipment for static testing is cheaper and more accurate measurement is possible. Some researchers also studied the algorithms using the dynamic and static test data together (Hajela and Soeiro 1989, Wang et al 2001).

A number of researchers have proposed damage detection algorithms that bypass the tedious system identification usually employed in traditional vibration-based NDE method. The measured data may include the natural frequencies, mode shapes, mode shape curvature or transfer function (Rickles and Kosmatka 1992, Pandey et al 1991). These researchers analyzed the variations of the measured data for the undamaged and damaged structure to identify the damage.

Kim and Bartkowicz (1994) used a two-step approach, in which the first step uses the optimal update method to find a general area of damage, and the second step used the sensitivity technique to locate the damaged element. The sensitivity technique computed the sensitivity of the modal parameters to changes in selected model properties. It then found the optimum set of corrections that minimized the error between the test and analysis. Zimmerman et al (1994) showed that using multiple approaches not only provided a level of confidence in damage assessment, but some types of damage were more easily revealed using one approach as opposed to others.

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

Juneja et al (1994) presented an algorithm based on contrast maximization. The focus of this approach was to employ contrast maximization vectors containing maximum information about the location of the damage. The feasibility of this approach was tested “numerically” on a 132-degree-of-freedom structure equipped with three active members. The results indicated that contrast maximization had a high probability of identifying a single damaged member, and could be used for real-time applications.

A number of researchers have used neural networks for damage identification (Rhim and Lee 1995, Sensmeier et al 1995). The damage identification problem is divided into two phases: (1) Pattern generation/training phase; and (2) the pattern recognition phase. The measured data may include the natural frequencies, mode shapes or the transfer function, or contrast maximization vectors.

Expert system has also been employed for structural health monitoring and repair. Schuetze et al (1994) used an expert system to diagnose whether a building satisfied an ambient vibration regulation. The expert system helped the engineer to select sensors based on given knowledge of the structure, suggested where the sensors should be located, and then organized the data after the sensors were installed. Roddis et al (1995) developed a rule-based expert system, which was later modified into a case-based expert system (Roddis and Bocox 1997), to give suggestions on the remedial actions for fabrication errors in steel bridge members. The developed expert system is presently in operation at the bridge offices of the Kansas Department of Transportation.

5.3 IMPEDANCE-BASED METHOD

Recently, an impedance-based NDE method capable of in-service, on-line, incipient damage detection in complex structures has been proposed at the Center for Intelligent Material Systems and Structures (CIMSS) of Virginia Polytechnic

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

Institute and State University (Sun et al. 1995). The basic concept of this method is to monitor the variations in the structural mechanical impedance caused by the presence of damage in the structure. Since structural mechanical impedance measurements can be difficult to obtain, this NDE technique utilizes the electromechanical coupling property of piezoelectric materials. The piezoelectric transducers acting in the direct manner produce an electrical charge when mechanically stressed. Conversely, a mechanical strain is produced when an electrical field is applied. By this coupling property, the electrical impedance of piezoelectric transducer is directly related to the mechanical impedance of the structure being bonded, and will also be affected by the presence of damage. Through monitoring the measured impedance and comparing it to a baseline impedance measurement, it can be qualitatively determined that structural damage has occurred or is imminent. In order to ensure high sensitivity to incipient damage, the electrical impedance is measured at high frequencies, at which the wavelengths of the structural vibration are shorter than the damage to be detected.

The electromechanical (E/M) impedance method utilizes the direct as well as the converse E/M properties of piezoelectric materials. Hence, one PZT patch can be used for both actuation and sensing of the structure response. The piezoelectric transducers used in the E/M impedance technique are thin Lead (Pb) Zirconate Titanate (PZT) ceramic wafers. The PZT patch can be intimately bonded to an existing structure, or embedded into a new composite construction, as shown in Fig. 5.1. When an alternating electric field is applied to the PZT transducer, a local strain produced by the patch creates strain waves in the host structure. In response, the structure presents to the transducer a mechanical point impedance, which modifies the effective electrical impedance of the piezoelectric transducer through the coupling property of the PZT patch.

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

Fig. 5.1 Details of E/M impedance-base NDE methods

The interaction of a PZT transducer with the host structure can be described by a simple impedance model. Fig. 5.2 shows the diagrammatic representation. The PZT is considered as a thin bar undergoing axial vibration in response to the applied alternating voltage. Solving the equation for the PZT bar connected to the external mechanical point impedance of the structure leads to the following equation for the frequency-dependent electrical impedance (Sun et al 1995)

$$Y(\omega) = j\omega a \left[\epsilon_{33}^T (1 - j\delta) - \frac{Z_{str}(\omega)}{Z_{str}(\omega) + Z_p(\omega)} d_{31}^2 \tilde{Y}_{11}^E \right]$$

where Y is the electrical admittance (inverse of impedance); j symbolizes the imaginary part; Z_p and Z_{str} are the PZT's and the structure's mechanical

Fig. 5.2 1-D model used to represent a PZT-driven dynamic structural system

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

impedance, respectively; $\tilde{Y}_{11}^E$ is the complex Young's modulus of the PZT at zero electric field; d_{31} is the piezoelectric coupling constant in the I - direction at zero stress; ϵ_{33}^T is the dielectric constant at zero stress; δ is the dielectric loss tangent of the PZT; and a is the geometric constant of the PZT. This equation indicates that the electrical impedance of the PZT bonded onto the structure is directly related to the mechanical impedance of the host structure. When the host structure is damaged the parameters, such as mass, stiffness or damping of the structure would be changed. Consequently, the electrical impedance of the transducer will be modified.

The application of the E/M impedance-based method includes an array of piezoelectric PZT transducers bonded to the monitored structure, and a high precision impedance analyzer coupled to a data-acquisition computer (Giurgiutiu and Rogers 1997). The variation of the E/M impedance of the PZT transducer is monitored over a large frequency spectrum in the high KHz frequency band. Fig. 5.3a shows the frequency response diagrams obtained during a typical laboratory demonstration of the E/M impedance technique. The variation in the electrical impedance of a PZT transducer over a frequency range is analogous to the frequency response but has much higher resolution and is more easily obtained.

As soon as the impedance spectrum of the structure is measured, it is compared to the previously measured baseline so that meaningful interpretation can be extracted pertinent to the modifications that have taken place in the structure, including the appearance of defects. The health of the structure can be qualitatively estimated through a damage index. The damage index is a scalar quantity that serves as a metric of the damage taking place in the structure. For example, the root mean square deviation damage index is defined as:

$$DIndex = \sum_{i=1}^N [\text{Re}(Y_i^1) - \text{Re}(Y_i^0)]^2$$

where N is the number of sample points in the impedance spectrum; and the superscripts 0 and 1 signify the initial and the present state of the structure. Mean absolute percentage deviation, covariance change, and correlation coefficient deviation have also been used as the damage index. Fig. 5.3b presents a typical

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

damage index diagram that distinguishes between damaged and undamaged locations during the health-monitoring process.

Fig. 5.3 Typical impedance-based damage detection results (Chardhry et al. 1995)

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

Since the development of impedance-based NDE technique, it has been widely applied to various engineering disciplines, including aerospace and aircraft structures (Chaudhry et al 1994, 1995; Giurgiutiu 2000; Giurgiutiu et al 2001; Ghoshal et al 2001), and civil structures (Ayres, et al 1996; Soh et al 2000, Park et al 2000, 2001; Bhalla et al 2001). Some of the applications were even under significant varying temperature conditions (Park et al 1999).

Park et al (2000) compared the impedance-based NDE technique with the other traditional NDE methods, and showed that this technique has some principle advantages: (1) the technique is not based on my model, and thus can be easily applied to complex structures; (2) the technique uses small non-intrusive actuators to monitor inaccessible locations; (3) the sensor (PZT) exhibits excellent features under normal working conditions, having a large range of linearity, fast response, lightweight, high conversion efficiency, and long term stability; (4) the technique, because of high frequency, is very sensitive to minor changes; (5) the measured data can be easily interpreted; (6) the technique can be implemented for on-line health monitoring; and (7) the continuous monitoring provides a better assessment of the current status of the structure, which can eliminate scheduled base inspections. Finally they concluded that the impedance-based technique is able to provide an effective mean to qualitatively detect incipient damage in complex structures, and has the potential to be developed into a completely autonomous monitoring system.

Despite the advantages listed above, the impedance-based NDE technique has some limitations. In the previous applications, the electrical impedance is measured at very high frequency (typically higher than 30KHz). The excited very high mode of the structure are essentially the local modes, since damping is much more significant at high ultrasonic frequencies (Sun et al 1995). Thus, the sensing region of the PZT is localized to region close to the sensor/actuator. Esteban (1996) did an extensive theoretical modeling study based on the wave-propagation approach to identify the effective sensing region of the impedance-based NDE method. Based on the knowledge acquired through various case studies, Park et al (2000) estimated that the effective sensing area of a single PZT could vary from

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

0.4m to 2m depending on the material and density of the structure. Other than the limited sensing area, the damage indices used for damage assessment is also not able to simultaneously locate and quantify the damage, since a severe damage far from the measured point and a minor damage close to the point could have the same damage index. Moreover, different threshold values of damage index may needed for different structures to determine the danger or to predict the failure.

5.4 PROPOSED INTEGRATED EP AND IMPEDANCE BASED TECHNIQUE

5.4.1 Principle

Rytter (1993) defined four levels of classification for the NDE methods:

- to determine whether damage has occurred in the structure;
- to determine the geometric location of the damage;
- to quantify the severity of the damage; and
- to predict the remaining service life of the structure.

The remaining life of the structure can be estimated based on fracture mechanics, fatigue life analysis or structural design assessment as long as the location and quantity of damage have been determined. Thus, only the first three steps will be considered in this study. Doebling et al (1996) replenished that a robust NDE method should have two more features:

- to be well-suited to automation; and
- not to be dependent on the engineering judgment of the user or an analytical model of the structure to the greatest extent.

In view of these requirements, each of the current damage detection techniques has its merits as well as limitations. The traditional experimental methods can obtain detailed information of damage, but they need to know the vicinity of the damage a priori, and that the portion of the structure to be inspected

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

is readily accessible. Vibration-based methods can obtain the global information of the structure but sometimes difficult to operate. The traditional impedance-based approaches can overcome the difficulty in operation but have some problems in global inspection and damage assessment. A less ambitious, but more attainable, goal of this study would be to develop a new technique that has the features listed above.

It is noticed that the vibration-based methods and impedance-based techniques have complementary advantages, i.e., the former can identify damages in the entire structure and the latter is convenient in measurement. These features inspire us to integrate the two groups of methods to develop a new NDE technique.

Since impedance-based methods are convenient in operation, the experimental process will be used in the new technique. In order to break the bounds on sensing region in traditional impedance-based methods, the integrated technique will work in lower frequencies so that global vibrations are excited, thus, the health of the entire structure can be monitored. System identification algorithm used in vibration-based methods will be employed in the integrated technique for damage assessment. Considering that there are numerous local optima in the search space which would trap the traditional local optimization strategies, the hybrid evolutionary programming (EP) algorithm presented in Chapter 3 will be employed to identify the location and quantity of the damage. Thus, the proposed new NDE technique is an integration of EP for system identification and impedance-based model for system characterization. The integrated EP and impedance-based (inEPIB) technique will not only have the advantages of both vibration- and impedance-based methods but also overcome their limitations.

The process of the inEPIB technique is illustrated in Fig. 5.4. It contains an experimental subroutine to measure the electrical impedance of PZT transducers and a numerical subroutine to identify the damages.

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

In order to minimize the effect of modeling errors and to enhance the practicability of the inEPIB technique, a substep of baseline comparison is employed which will be illustrated in the following two chapters.

Fig. 5.4 Outline of inEPIB technique

5.4.2 Measurement system

To implement the inEPIB technique, an experimental system similar to that of impedance-based methods is developed to measure the electrical impedance of the PZT transducer. The system contains an impedance analyzer (Model: HP-4192A), a multiplex terminal and a personal computer equipped data acquisition software and a PC-analyzer interface cable. The whole system is depicted in Fig. 5.5 and its setup is shown in Fig. 5.6.

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

Figure 5.5 Depiction of measurement system

Fig. 5.6 Schematic sketch of measurement system

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

For a structure to be continuously monitored, small-size PZT transducers are permanently bonded to it. Prior to the bonding process, the surface of the structural substrate is cleansed to achieve good surface adherence. High strength epoxy adhesive is recommended in the process of bonding to ensure that the PZT actuator/sensor is fully attached to the structure. Thus, the electrical impedance of the PZT transducer contains the information of the structure and can be used for continuous health monitoring.

Two electrical wires are soldered to the two electrodes of the PZT transducer, and subsequently connected to the multiplex terminal. The wires should be twisted together to minimize electrical interfaces. The multiplex terminal consists of a forty-channel N2260A multiplexer module housed inside the 3499B switch control system designed and built by Agilent Technologies. A program, functional in the HP-VEE 3.0 software environment, is run to control the operations of the impedance analyzer. The control and acquisition of the experimental data is achieved through the multiplex terminal and the GPIB interface card installed in the computer. When an excitation electrical field with certain frequency is applied to the PZT transducer, the electrical impedance or admittance can be recorded and stored in the personal computer as a function of the exciting frequency.

5.5 SUMMARY

After review of the existing NDE methods, a new NDE technique is proposed by combining the vibration-based methods and the impedance-based methods. A system for data collection is also developed. The proposed technique is expected to have the advantages of both groups of methods and can satisfy all of the requirements for a robust NDE method.

So far, the expectations of the proposed technique are only based on theoretical analysis. The practicability has to be tested with real applications. Thus,

Chapter 5: Integrated EP and Impedance-Based Technique for Damage Detection

in the next chapters, the proposed technique is tested on a lab-sized beam and a lab-sized plate to verify its feasibility for real-life application.

CHAPTER 6 DAMAGE DETECTION IN BEAM

6.1 INTRODUCTION

The integrated evolutionary programming and impedance-based (inEPIB) technique proposed in the previous chapter is first implemented for one-dimensional (1-D) structures. Prior to the inverse problem for system identification, the forward problem must be solved to predict the electromechanical (E/M) impedance of the PZT transducer bonded on a damaged 1-D structure.

6.2 IMPEDANCE MODEL

The forward problem is defined to predict the electrical impedance of the PZT transducer with the properties of the transducer and the host structure known.

Despite its wide acceptance as an effective technique for structural health monitoring, damage detection and failure prevention, the impedance-based methods have so far evolved through experimental discovery and proof-of-concept demonstrations. Little work has been done to determine the analytical model which can correctly predict the impedance response for a given structure and its change with the progression of damage (Giurgiutiu and Rogers 1999).

Chapter 6: Damage Detection in Beam

For the modeling of piezoelectric elements integrated with host structures, several static-based modeling techniques have been developed to determine the induced loading (Bailey and Hubbard 1985, Crawley and de Luis 1987, Fanson and Caughey 1987, Dimitriadis et al 1989, Cudney et al 1990, Kim and Jones 1991, Wang and Rogers 1991). It is typically assumed in the static models that piezoelectric actuators do not significantly alter the inertial mass and stiffness of the host structures. Static analysis usually leads to the conclusion that the amplitude of the actuation strain of the PZT actuator is frequency independent. Furthermore, dynamic analysis based on the static strain excitation is not accurate in predicting the dynamic response of the integrated PZT/substrate systems (Zhou et al 1996). Lalande (1995) and Fairweather (1998) have provided comprehensive literature reviews on the abovementioned static-based models.

The first impedance-type model for PZT-structure interacting system was proposed by Liang et al (1993, 1996). They formulated the force output of a piezoelectric actuator-driven one-degree-of freedom (1-DOF) spring-mass-damper system (Fig. 6.1). In Liang's model, the PZT patch was assumed to be fixed on one end and connected to the host structure on the other end. Experiments showed that Liang's model could favorably predict a good velocity response of a circular ring (Rossi et al. 1993). However, no closed form solution for the electrical impedance of the transducer was determined.

Fig. 6.1. Liang's model

Chapter 6: Damage Detection in Beam

Giurgiutiu and Zagari (2001) characterized the PZT transducer with free, clamped and elastically constrained boundary conditions. Their model was used to predict the E/M impedance response. However, the predicted real part of the E/M admittance (inverse of impedance) for a free beam structure was more than ten times smaller than the experimentally measured data in low frequencies.

For the successful implementation of the inEPIB NDE technique, this study developed a generic impedance model for the PZT-structure interacting system, such that the E/M impedance can be precisely predicted for the undamaged and damaged structures.

6.2.1 Modeling of PZT transducer

A generic 1-DPZT-structure interacting system is illustrated in Fig. 6.2. The length, width, and thickness of the PZT patch are denoted as a_p , b_p and h_p , respectively. The subscript p denotes the parameters of the PZT patch. The effect of the host structure on the PZT patch is represented by its force impedances at the points where the PZT patch and the host structure are bonded. Since force impedance is point-dependant, different values are assumed for the force impedance at the two ends. The active forces applied to the PZT patch can be expressed as:

$$F|_{x_p=0} = -Z'_{str} \dot{u}|_{x_p=0} \quad (6.1a)$$

$$F|_{x_p=a_p} = -Z_{str} \dot{u}|_{x_p=a_p} \quad (6.1b)$$

Fig. 6.2 Generic one-dimensional PZT-structure interacting system

Chapter 6: Damage Detection in Beam

where Z_{str} and Z'_{str} are the force impedances of the host structure, as shown in Fig.6.2, u is the displacement of the PZT patch, and "." denotes the derivative of time. The minus sign indicates that the structural reactions are equal and opposite to the output force of the PZT actuator.

Under small field conditions, the constitutive relations for a piezoelectric material are (IEEE Standard 1987):

$$S_p = \tilde{s}_{pq}^E T_q + d_{kp} E_k \quad (6.2a)$$

$$D_i = d_{iq} T_q + \tilde{\epsilon}_{ik}^T E_k \quad (6.2b)$$

where $p, q = 1, 2, \dots, 6$, $i, k = 1, 2, 3$, S_p is the mechanical strain; T_q is the mechanical stress; $\tilde{s}_{pq}^E = s_{pq}^E (1 + j\eta)^{-1}$ is the complex mechanical compliance of the material measured at zero electric field; η is the mechanical loss factor; j symbolizes the imaginary part; d_{kp} is the induced strain coefficient, i.e., mechanical strain per unit electric field; $\tilde{\epsilon}_{ik}^T = \epsilon_{ik}^T (1 - j\delta)$ is the complex permittivity at constant stress; δ is the dielectric loss factor; E_k is the applied external electric field in the k -direction; and D_i is the electric displacement.

When the PZT patch is bonded to the structure for damage detection, the electrical field is generally applied in the z -direction, and for 1-D problem, only the deformation in the x -direction is considered, thus, Eqn. (6.2) can be simplified as:

$$S_1 = \frac{1}{\tilde{Y}_p^E} T_1 + d_{31} E_3 \quad (6.3a)$$

$$D_3 = d_{31} T_1 + \tilde{\epsilon}_{33}^T E_3 \quad (6.3b)$$

where $\tilde{Y}_p^E = Y_p^E (1 + j\eta)$ is the complex Young's modulus at zero electric field; Y_p^E is the real Young's modulus; the subscript p indicates that the variable is for PZT patch, and $E_3 = \frac{V}{h_p}$ is the applied electrical field in the 3-direction. For harmonic

Chapter 6: Damage Detection in Beam

motion, $E_3 = \bar{E}e^{j\omega t}$, where ω is the input angular frequency. Generally, $\frac{\partial E_3}{\partial x} = 0$ is assumed.

Considering the force equilibrium of an infinitesimal beam element, the governing equation for the longitudinal vibration of the PZT patch can be described by:

$$\tilde{Y}_p^E \frac{\partial^2 u}{\partial x^2} = \rho_p \ddot{u} \quad (6.4)$$

where ρ_p is the mass density of the PZT patch.

The solution of Eqn. (6.4) can be expressed as:

$$u = (\alpha \sin k_p x_p + \beta \cos k_p x_p) e^{j\omega t} \quad (6.5)$$

where $k_p = \omega \sqrt{\rho_p / \tilde{Y}_p^E}$ is the wave number; and α and β are the coefficients to be determined from the boundary conditions.

At the two ends of the PZT patch, the induced stresses can be expressed in term of the forces applied on the patch. At the right end, the force provides tension:

$$T_1|_{x_p=a_p} = \frac{F}{b_p h_p} = -\frac{Z_{str} \dot{u}|_{x_p=a_p}}{b_p h_p} \quad (6.6a)$$

while at the left end, the force provides compression:

$$T_1|_{x_p=0} = -\frac{F'}{b_p h_p} = \frac{Z'_{str} \dot{u}|_{x_p=0}}{b_p h_p} \quad (6.6b)$$

Substituting Eqn. (6.5) and the constitutive Eqn. (6.3a) into the boundary conditions Eqn. (6.6), the following equations are obtained:

$$\begin{aligned} k_p (\alpha \cos k_p a_p - \beta \sin k_p a_p) - d_{31} \bar{E} \\ = -\frac{Z_{str} j \omega}{\tilde{Y}_p^E b_p h_p} (\alpha \sin k_p a_p + \beta \cos k_p a_p) \end{aligned} \quad (6.7a)$$

Chapter 6: Damage Detection in Beam

$$k_p \alpha - d_{31} \bar{E} = \frac{Z'_{str} j \omega}{\tilde{Y}_p^E b_p h_p} \beta \quad (6.7b)$$

Consequently, the coefficients α and β can be determined by solving Eqn. (6.7):

$$\beta = - \frac{\frac{Z'_{str} j \omega}{\tilde{Y}_p^E b_p h_p} \sin k_p a_p - k_p (1 - \cos k_p a_p)}{\frac{Z'_{str} j \omega}{\tilde{Y}_p^E b_p h_p} \left(\frac{Z'_{str}}{Z_{str}} + \cos k_p a_p \right) - k_p \sin k_p a_p} \cdot \alpha = -M \cdot \alpha \quad (6.8a)$$

$$\alpha = \frac{1}{k_p + \frac{Z'_{str} j \omega}{\tilde{Y}_p^E b_p h_p} M} d_{31} \bar{E} = \alpha' \bar{E} \quad (6.8b)$$

Substituting Eqn. (6.8) into Eqns. (6.5) and (6.3a), the stress distribution in the PZT patch is obtained as:

$$T_1 = \tilde{Y}_p^E \left[k_p (\cos k_p x_p + M \sin k_p x_p) \alpha' - d_{31} \right] \bar{E} e^{j \omega t} \quad (6.9)$$

The electrical displacement is calculated by substituting Eqn. (6.9) into the constitutive equation, Eqn. (6.3b)

$$D_3 = \left\{ d_{31} \tilde{Y}_p^E \left[k_p (\cos k_p x_p + M \sin k_p x_p) \alpha' - d_{31} \right] + \tilde{\epsilon}_{33}^T \right\} \bar{E} e^{j \omega t} \quad (6.10)$$

The electric current can be determined by integrating D_3 with respect to the surface area of the PZT patch, given by:

$$I = \int_{A_p} \dot{D}_3 dA_p = b_p \int_0^{a_p} \dot{D}_3 dx \quad (6.11)$$

From the definition of electric admittance $Y = I/V$, we obtain:

$$Y = \frac{j \omega b_p}{h_p} \left\{ d_{31} \tilde{Y}_p^E \left[(\sin k_p a_p + M (1 - \cos k_p a_p)) \alpha' - d_{31} a_p \right] + \tilde{\epsilon}_{33}^T a_p \right\} \quad (6.12)$$

The electrical impedance Z is the inverse of the electrical admittance Y

Chapter 6: Damage Detection in Beam

$$Z = \frac{1}{Y} \quad (6.13)$$

This model represents a general case for the PZT patch. Liang et al. (1993) assumed that the PZT patch is fixed at one end and bonded to the host structure at the other end, which corresponds to the case of $Z'_{str} = \infty$ in the generic model. The PZT active sensor with free, clamped and elastically constrained boundary conditions, analyzed by Giurgiutiu and Zagari (2001), corresponds to the cases of $Z'_{str} = Z_{str} = 0, \infty$, and a value varied with frequency, respectively. Other boundary conditions can be simulated by selecting various values for Z_{str} and Z'_{str} .

In the generic model, the host structure has different force impedances at the two ends of the PZT patch. However, for most cases, the scale of the host structure is much larger than the PZT patch, thus the influence of the dimensions of the PZT patch on the force impedance can be neglected. That is, the two ends of the PZT patch can be viewed as to be close enough such that $Z_{str} = Z'_{str}$.

With this assumption, Eqn. (6.8) can be simplified as:

$$\beta = -\alpha \tan(k_p a_p / 2) \quad (6.14a)$$

$$\alpha = \frac{1}{k_p + \frac{Z_{str} j \omega \tan(k_p a_p / 2)}{\tilde{Y}_p^E b_p h_p}} d_{31} \bar{E} = \alpha' \bar{E} \quad (6.14b)$$

Substituting Eqn. (6.14) into Eqn (6.5), the displacement distribution in the PZT patch is obtained:

$$u = \frac{\alpha}{\cos(k_p a_p / 2)} \sin \frac{k_p (x_p - a_p / 2)}{2} e^{j \omega t} \quad (6.15)$$

The E/M admittance can then be modified as:

$$Y = \frac{j \omega b_p}{h_p} \left\{ d_{31} Y_p^E \left[2 \alpha' \tan \frac{k_p a_p}{2} - d_{31} a_p \right] + \epsilon_{33}^r a_p \right\} \quad (6.16)$$

Chapter 6: Damage Detection in Beam

By now, the E/M admittance of the PZT transducer is obtained as a function of the PZT material properties and the host structure's force impedance. As long as the force impedance of the host structure is determined, the E/M admittance can be predicted by Eqns. (6.12) or (6.16). It is noticed that no restriction has been imposed on the host structure in the derivation of Eqn. (6.12), thus it gives a general solution for any 1-D PZT transducer. Consequently, Eqn. (6.16) simplifies the formula with the assumption of $Z_{str} = Z'_{str}$, which holds if the size of the host structure is much larger than that of the transducer. Since this requirement can be satisfied for most structures, Eqn. (6.16) will be adopted to predict the E/M admittance in the damage detection for 1-D structures.

The forces applied on the host structure are calculated from the force impedance and velocity response at the corresponding points:

$$F|_{x=x_1} = Z'_{str} \dot{u}|_{x=x_1} = Z'_{str} \dot{u}|_{x_p=0} = -Z'_{str} \frac{\alpha}{\cos(k_p a_p/2)} \sin \frac{k_p (a_p/2)}{2} e^{j\omega t} \quad (6.17a)$$

$$F|_{x=x_2} = Z_{str} \dot{u}|_{x=x_2} = Z_{str} \dot{u}|_{x_p=a_p} = Z_{str} \frac{\alpha}{\cos(k_p a_p/2)} \sin \frac{k_p (a_p/2)}{2} e^{j\omega t} \quad (6.17b)$$

where x is the longitudinal axis established for the host structure; and x_1 and x_2 are the coordinates of the left and right PZT ends, respectively, as shown in Fig. 6.3.

Fig. 6.3. Beam actuated by a pair of PZT patches

Considering the assumption of $Z_{str} = Z'_{str}$, Eqn. (6.17) implies that the forces applied on the host structure are a pair of forces with the same amplitude but in the opposite directions. With $\bar{F}$ denoting the magnitude, the output forces can be expressed as:

$$F = \bar{F}[\delta(x - x_1) - \delta(x - x_2)] \quad (6.18)$$

where $\delta(x)$ is the dirac function which is defined as:

$$\delta(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \infty & \text{if } x = 0 \end{cases} \text{ and } \int_{-\infty}^{\infty} \delta(x) dx = 1.$$

Moreover, subtracting Eqn. (6.17a) by Eqn. (6.17b) and considering Eqn. (6.18), the following equation is obtained:

$$Z_{str} = 2 \frac{\bar{F}}{\dot{u}|_{x=x_1} - \dot{u}|_{x=x_2}} \quad (6.19)$$

which provides a method to calculate the force impedance of the host structure. The relative velocity of the host structure under a pair of forces with the same amplitude but opposite directions can be obtained by analytical, numerical or experimental method for different frequencies. With Eqn. (6.19), the force impedance of the host structure can be calculated, and the E/M admittance of the interacting system can then be predicted by Eqn. (6.16).

6.2.2 Modeling of host structure

In this study, a beam is selected to represent a 1-D structure for damage detection. The length, width and thickness of the beam are denoted as a , b and h , respectively. Two PZT transducers are bonded to the beam at the same location but opposite sides, as shown in Fig. 6.3. Through the multiplex terminal in the measurement system, opposite electrical fields with the same amplitude are applied to the two patches so that only moment

$$M = \bar{F} \cdot \frac{h + h_p}{2} [\delta(x - x_1) - \delta(x - x_2)] = \bar{M} [\delta(x - x_1) - \delta(x - x_2)] \quad (6.20)$$

is applied to the beam and pure bending vibration is actuated, where $\bar{M}$ represents the magnitude of the moment.

Chapter 6: Damage Detection in Beam

For simplicity, Berlouli-Euler beam model is considered. Thus, only the transverse displacement needs to be considered, which is denoted as w . The displacement field is expressed as:

$$\begin{aligned} w &= w(x, t) \\ u &= -z \frac{\partial w}{\partial x} \end{aligned} \quad (6.21)$$

Various types of damages may occur in the structure. However, this study only concentrates on the damage's effect on the structure's mechanical properties. Generally, the emergence of damage mainly causes a difference in the structure's stiffness, thus, the damage is simulated by the change of Young's modulus in the damaged area. The effect of damage on the mass density and damping is neglected.

Fig. 6.4. Damaged beam model

Fig. 6.4 shows a damaged beam. The shaded area represents the damage element. The damage is simulated by a reduced Young's modulus E_d , which is less than the Young's modulus of the undamaged beam, E . The length of the damage element is denoted as L_d . The location of the damage is defined by the coordinate of the damage center, x_c .

Ritz method is employed to analyze the vibration of the beam with a damage element. The potential and kinetic energy for the damaged beam can be expressed as:

$$U = \frac{1}{2} \int \sigma^T \epsilon dV = \frac{1}{24} E b h^3 \int_0^a \left(\frac{\partial^2 w}{\partial x^2} \right)^2 dx - \frac{1}{24} (E - E_d) b h^3 \int_{L_d} \left(\frac{\partial^2 w}{\partial x^2} \right)^2 dx \quad (6.22)$$

$$T = \frac{1}{2} \rho b h \int_0^a \dot{w}^2 dx \quad (6.23)$$

Chapter 6: Damage Detection in Beam

where ρ is the mass density of the beam. For a minor damage, the potential energy can be approximated as:

$$U = \frac{1}{24} E b h^3 \int_0^a \left(\frac{\partial^2 w}{\partial x^2} \right)^2 dx - \frac{1}{24} (E - E_d) b h^3 \left(\frac{\partial^2 w}{\partial x^2} \right)^2 \bigg|_{x=x_d} \cdot L_d \quad (6.24)$$

Without loss of generality, the beam in x - z coordinate system is mapped to a standard beam in ξ - z coordinate system by the following transformation:

$$\xi = \frac{2x}{a} - 1 \quad (6.25)$$

Thus, the energy functional can be written as:

$$U = \frac{1}{3} \frac{E b h^3}{a^3} \int_{-1}^1 \left(\frac{\partial^2 w}{\partial \xi^2} \right)^2 d\xi - \frac{2}{3} \frac{(E - E_d) b h^3}{a^4} \cdot \left(\frac{\partial^2 w}{\partial \xi^2} \right)^2 \bigg|_{\xi=\xi_d} \cdot L_d \quad (6.26)$$

$$T = \frac{1}{4} \rho a b h \int_{-1}^1 \dot{w}^2 d\xi \quad (6.27)$$

In Ritz method, the displacement is approximated as a combination of a set of shape functions:

$$w = \sum_{i=0}^N c_i \phi_i e^{j\omega t} = \mathbf{C} \boldsymbol{\Phi} e^{j\omega t}$$

where N is the degree set of polynomial space; $\mathbf{C}$ is the coefficient vector; $\boldsymbol{\Phi}$ is the shape function vector; and $e^{j\omega t}$ stands for harmonic vibration.

Since polynomial has some advantages in calculation, differentiation and integration, it is selected as the type of shape functions. Similar to the 2-D polynomial shape function for plate structures, which has been described in Chapter 4, the shape function for the beam is designed as:

$$\phi_i = \xi^i \phi_0, i = 1, 2, \dots, N \quad (6.29)$$

$$\phi_0 = (\xi + 1)^{\mathcal{G}_1} (\xi - 1)^{\mathcal{G}_2} \quad (6.30)$$

where the values of $\mathcal{G}_1$ and $\mathcal{G}_2$ can be selected from Table 4.1 for various boundary conditions at the corresponding ends of the beam.

Chapter 6: Damage Detection in Beam

Thus, the stiffness and mass matrices can be calculated as:

$$K_{mn} = \frac{\partial^2 U}{\partial c_m \partial c_n} = \frac{2}{3} \frac{Ebh^3}{a^3} \int_{-1}^1 \phi_{mn} d\xi - \frac{4}{3} \frac{(E - E_d)bh^3}{a^4} \phi_{mn} \Big|_{\xi=\xi_c} \cdot L_d \quad (6.31)$$

$$M_{mn} = \frac{1}{\omega^2} \frac{\partial^2 T}{\partial c_m \partial c_n} = \frac{1}{2} \rho abh \int_{-1}^1 \phi_m \phi_n d\xi \quad (6.32)$$

where $\phi_{mn} = \frac{\partial^2 \phi_m}{\partial \xi^2} \frac{\partial^2 \phi_n}{\partial \xi^2}$ and $m, n = 0, 1, \dots, N$.

By solving the eigenvalue equation:

$$(\mathbf{K} - \mathbf{M}\lambda)\mathbf{C} = 0 \quad (6.33)$$

the natural frequencies and modal shapes can be calculated:

$$\omega_k = \sqrt{\lambda_k} \quad (6.34)$$

$$\text{and } \Phi_k = \sum_i c_{ik} \phi_i = \mathbf{C}_k^T \Phi \quad (6.35)$$

where $\mathbf{C}_k$ is the eigenvector corresponding to the k -th eigenvalue; and the superscript T denotes transpose.

Assuming that the actuated vibration is a superposition of different vibration modes,

$$w = \sum_k \gamma_k \Phi_k e^{j\omega t} \quad (6.36)$$

and considering the orthogonality of the modal shapes, the modal participation factor can be calculated as:

$$\gamma_k = \frac{G_k}{\omega_k^2 - \omega^2} \quad (6.37)$$

$$\text{where } G_k = \frac{\int_0^a \mathbf{C}_k^T \Phi \frac{\partial M}{\partial x} dx}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k}$$

$$= \frac{\frac{2}{a} \sum_i \left[c_{ik} \left(\frac{\partial \phi_i}{\partial \xi} \Big|_{\xi=\xi_2} - \frac{\partial \phi_i}{\partial \xi} \Big|_{\xi=\xi_1} \right) \right]}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k} \cdot \bar{M} \quad (6.38)$$

Chapter 6: Damage Detection in Beam

The relative velocity in Eqn. (6.19) can be calculated:

u|_{x=x_1} - u|_{x=x_2} = -j\omega \frac{h+h_p}{2} \left[\left(\frac{\partial w}{\partial x} \right) \Big|_{x=x_1} - \left(\frac{\partial w}{\partial x} \right) \Big|_{x=x_2} \right] \tag{6.39}

and the force impedance of the damaged beam can then be determined.

6.2.3 Model verification

To verify the above model, the electrical admittance of two PZT patches bonded on a lab-sized beam is recorded with the measurement system developed in the previous chapter. Tables 6.1 and 6.2 listed the properties of the PZT patch and the beam specimen, respectively. Two PZT patches are bonded to the beam at the same location but on the opposite sides. RS159-3957high strength epoxy adhesive is used for bonding to ensure fullattachment between the PZT and the specimen. The beam specimen rests on soft papers to simulate free-free boundary condition. Opposite electrical fields are applied to the two patches so that pure bending vibration is actuated in the specimen. The exciting frequency ranges from 1KHz to 10KHz. The interval of the exciting frequencyis selected as small as 1Hz, so that very small variations caused by minor damage can be detected and accurate data can be obtained for the system identification algorithm to identify the damage.

Table 6.1. Properties of PZT patch

Dimension (mm)	Location (mm)	Young's Modulus (GPa)	Loss Factor	Mass Density (kg/m ³)	Strain Constant d (m/volt)	Permittivity (farad/m)	Dielectric Loss Factor
a _p =20 b _p =2 h _p =0.2	x ₁ =80 x ₂ =100	66.7	0.03	7800	-2.10E-10	2.14E-08	1.85E-02

The vibration of the beam structure is first analyzed using Ritz method to calculate the force impedance at the corresponding points. The accuracy of

Chapter 6: Damage Detection in Beam

calculated force impedance depends on the degree set of the polynomial. With more terms in the shape function, the result will be more accurate but more computing time is needed. Considering both the computing time and accuracy, the degree set of the polynomial shape functions is selected as $N = 24$.

Table 6.2. Properties of beam specimen

Length (mm)	Width (mm)	Height (mm)	Young's Modulus (GPa)	Mass Density (kg/m ³)	Damping Ratio
300	20	2	66.5	2600	0.01

The electrical admittance of the PZT transducer is then predicted by Eqns. (5.16). The comparison between the experimentally measured admittance (real part) and numerical prediction is shown in Fig. 6.5. Only the real portion of the E/M admittance is analyzed because it is more sensitive to damage than the imaginary part or magnitude, which are dominated by the capacitive response of the sensor and less sensitive to structural damage effects (Sun et al 1995).

Fig. 6.5. Comparison between experimental measurement and numerical prediction of undamaged beam

Chapter 6: Damage Detection in Beam

In the admittance spectra, there are some peaks, which are caused by resonance. Although small differences exist between values of the numerical prediction and experimental measurement, it is observed that the positions of the peaks agree well with each other. The frequencies corresponding to the measured and predicted admittance peaks are listed in Table 6.3, as well as the natural frequencies of the beam, for comparison.

Table 6.3. E/M admittance peaks and natural frequencies (KHz)

Natural Frequencies	1.542	2.153	2.867	3.682	4.599	5.619	6.738	7.985	9.203
Predicted Peaks	1.447	2.079	/	3.557	4.443	5.599	6.656	7.766	9.131
Measured Peaks	1.508	2.102	/	3.614	4.523	5.507	6.615	7.867	9.116

It is observed that each peak in the admittance corresponds to a natural frequency of the structure. One of the natural frequencies, i.e., 2.867KHz is not reflected in the admittance spectra. This is because the PZT patch is located on the vibration nodal line of the missed mode. It is calculated that the vibration mode corresponding to the natural frequency of 2.867KHz has a nodal line at $x=90\text{mm}$, which happens to be the center line of the PZT patch. Thus, this vibration mode has not been detected in the experiment. The admittance peaks in the predicted data agree well with the measured results, which demonstrates that the impedance model can predict the admittance peaks. In addition, the PZT patches locate at an unsymmetrical position in the structure, which lead to different force impedances at the two bonding points. However, the good agreement between the predicted results and the measured data indicates that the assumption of $Z_{str} = Z'_{str}$ is acceptable.

In spite of the good agreement between the measurements and predictions, there are still some discrepancies between them. These discrepancies are probably due to the modeling errors, e.g., the boundary condition is not totally free, variations in the specimen's properties, neglecting the adhesive layer, etc. For damage

detection using system identification technique, these modeling errors must be carefully handled to obtain a reliable result.

6.3 DAMAGE DETECTION

After the verification of the impedance model, the beam is then damaged by drilling a hole at the location of $x_c = 227.3\text{mm}$ with a radius of $r_d = 2.8\text{mm}$. The E/M admittance for the damaged beam is measured and compared with the baseline, as shown in Fig. 6.6a. Although the E/M admittance spectra are very close to the baseline in low frequency, changes are observed even for the minor damage (Fig. 6.6b).

In numerical simulation, the hole in the beam is simulated by a damage element. The element has the same width and thickness as the original beam but different Young's modulus. The length of the damaged element is defined as $L_d = 6r_d$ based on Saint Vienen's principle, and the Young's modulus in this area is approximately calculated by:

$$E_d \frac{bh^3}{12} \cdot L_d = \int_{L_d} E \frac{bh^3}{12} dx = E \frac{bh^3}{12} (L_d - 2r_d) + E \frac{h^3}{12} (2br_d - \pi r_d^2) \quad (6.40)$$

The measured and predicted impedance spectra of the damaged beam are compared in Fig. 6.7. Good match between them confirms that the developed impedance model is able to predict the impedance for undamaged and damaged beam. Thus, it is a generic model that can be applied to other 1-D structures.

It is observed that the occurrence of damage in the beam has caused variations in the E/M impedance, both in experiment and prediction. A typical admittance peak shift is illustrated in Fig. 6.8. These peak shifts contains the information of the damage, and will be employed to identify the damage.

Chapter 6: Damage Detection in Beam

The Young's modulus, location and length of the damaged element are supposed to be identified with the integrated evolutionary programming and impedance-based (inEPIB) NDE method. Thus, in the hybrid EP algorithm, each chromosome contains three variables: x_c to define the location, and E_d and L_d to define the quantity, of the damage. For convenience, the three variables are non-dimensionalized as $\frac{x_c}{a}$, $\frac{E_d}{E}$ and $\frac{L_d}{a}$, so that the search space is constrained in

$$0 < \frac{x_c}{a} < 1, 0 < \frac{E_d}{E} < 1 \text{ and } 0 < \frac{L_d}{a} < 0.1 \text{ for minor damage.}$$

Fig. 6.6. Measured E/M admittance for undamaged and damaged

Fig. 6.7. Comparison between experimental measurement and numerical prediction of damaged beam

Fig. 6.8. A typical admittance peak shift in beam caused by damage

Chapter 6: Damage Detection in Beam

In the traditional NDE methods, the predicted data for a damaged model are directly compared with the measured results for a damaged structure to assess the assumed damage:

$$f(\mathbf{D}) = \sum_i |P_i^p(\mathbf{D}) - P_i^e| \quad (6.41)$$

where $\mathbf{D}$ represents the damage; P is the data to be compared, which is the corresponding frequency of the admittance peak in this study; the superscripts p and e indicate that the data are numerically predicted and experimentally measured, respectively. In this formula, the discrepancy between $P_i^p(\mathbf{D})$ and P_i^e contains both the effect of damage and the modeling errors. As pointed out by Doebling et al (1996), an important feature of NDE methods “is their ability to discriminate between the model/data discrepancies caused by modeling errors and the discrepancies that are a result of structural damage”.

This study employs the baseline for the undamaged structure to discriminate the modeling errors. Eqn. (6.41) is modified as:

$$f(\mathbf{D}) = \sum_i | [P_i^p(\mathbf{D}) - P_i^e] - [P_i^p(\mathbf{D} = 0) - P0_i^e] | \quad (6.42a)$$

where $P0$ is the measured data for the undamaged structure. The second term in Eqn. (6.42a), which is the discrepancy between the prediction and the measurement for the undamaged structure, is considered as the discrepancy caused by modeling errors. On the other hand, Eqn. (6.42a) can also be written as:

$$f(\mathbf{D}) = \sum_i | [P_i^p(\mathbf{D}) - P_i^p(\mathbf{D} = 0)] - [P_i^e - P0_i^e] | \quad (6.42b)$$

The two brackets represent the discrepancies caused by the damage in the numerical model and in the real structure, respectively. Thus, only the variations caused by damages are considered and compared, and the effect of modeling errors can be minimized.

Chapter 6: Damage Detection in Beam

The effectiveness of using Eqn. (6.42) to discriminate the effect of modeling errors can also be illustrated by Fig. 6.8. A certain damage has caused similar shifts of the admittance peak in prediction and in experiment. More importantly, this similarity holds not only for this typical peak, but also for all of the other peaks, as illustrated in Fig. 6.9. Thus, by minimizing the fitness function, Eqn. (6.42), it is possible to identify the damage.

Fig. 6.9. Shifts of admittance peaks for beam specimen

With the measured data, the EP-MMLX algorithm presented in Chapter 3 is employed to identify the damage. Eqn. (6.42) is used for fitness evaluation. The values of the control parameters in EP algorithm, such as population size, tournament size, etc., are the same as those used in Chapter 3. The evolutionary processes of fitness and damage variables are illustrated in Fig. 6.10. Within 15 generations, the center of the damage is found at $\frac{x_c}{a} = 0.75612$ ($x_c = 226.84mm$).

The distance from the predicted damage center to the exact position ($x_c = 227.3mm$) is less than $0.5mm$. The Young's modulus and length of the

damaged element are obtained as $\frac{E_d}{E}=0.91344$ and $\frac{L_d}{a}=0,048505$. For direct comparison with the experiment, the identified radius of the hole in the beam is approximately calculated using Eqn. (6.40) and obtained as $r_d=2.83mm$, which is only $0.03mm$ larger than the exact radius of $r_d=2.8mm$. This result is only of 1% relative error. The evolutionary algorithm has conducted 3261 function evaluations before the convergence to the solution.

Fig. 6.10. Evolutionary process for damage detection in beam

Chapter 6: Damage Detection in Beam

The evolution continues to find an even better solution. After 100 generations, the damage is found at the location of $x_c = 226.85mm$ with the radius of $r_d = 2.78mm$. Both the location and quantity of the damage are very close to their exact values, with relative errors of 0.2% and 0.7%, respectively. Thus, it can be concluded that the damage in the beam can be quickly and precisely located and quantified using the integrated NDE technique.

6.4 SUMMARY

The inEPIB NDE technique is applied to a lab-sized beam to detect damage. An impedance model for the 1-D PZT-structure interacting system is developed. The developed model can precisely predict the admittance peaks. The small discrepancies between the measurements and predictions are viewed as modeling errors. The integrated technique measures the admittance spectra for both the undamaged and damaged structure to reduce the effect of modeling errors with a specially designed fitness function. The fitness function, which has been proven to be effective to exclude the effect of modeling errors, can not only be implemented in the present technique, but can also be adopted in the other vibration-based NDE methods.

It is noticed that the position of the PZT patch will affect the force impedance of the host structure, thus the E/M admittance of the patch will be different at different locations. If the PZT transducer happens to be located at the vibration nodal line of certain modes, the E/M admittance peaks corresponding to these modes may not be detected. However, in this case, the other peaks can be used to identify the location and quantity of a single damage.

The identification result shows that both the location and quantity of the damage in the beam can be successfully identified. The relative errors are less than 0.2% and 0.7%, respectively. Moreover, the convergence speed is fast, which is attributed to the high search efficiency of the hybrid EP algorithm.

Chapter 6: Damage Detection in Beam

In the next chapter, the possibility of damage detection in 2-D structures is investigated.

CHAPTER 7 DAMAGE DETECTION IN PLATE

7.1 INTRODUCTION

In this chapter, application of the integrated evolutionary programming and impedance-based (inEPIB) NDE technique is extended to two-dimensional (2-D) structures. Firstly, a generic 2-D impedance model is developed to predict the electrical admittance spectra of a PZT transducer attached to a 2-D structure. A thin plate is selected to represent the 2-D structure to verify the impedance model. Damage in the plate is then identified using the inEPIB technique.

7.2 IMPEDANCE MODEL

7.2.1 Modeling of PZT transducer

Unlike the 1-D problems, 2-D PZT-structure interacting system has seldom been studied. Zhou et al (1994, 1996) analyzed the vibration of plates excited by a pair of PZT actuators. The PZT-structure interaction was modeled as shown in Fig. 7.1, where the PZT actuator was fixed at two neighboring edges and bonded to the structure at the other two edges. By analyzing the vibration of the actuator and the structure separately, the velocity responses of the plate at certain points were calculated. The results were compared with the experimental measurements. Good agreement between them has been obtained. However, in Zhou's analysis and

Chapter 7: Damage Detection in Plate

experiment, the PZT patches worked only as actuator. The sensing function of the transducer was neglected. In addition, no closed form electrical impedance formulation for the PZT patch has been derived.

Fig. 7.1. Zhou's model

Zhou's approach is adopted in this study to develop a generic impedance model for 2-D PZT-structure interacting system. To closely simulate the real situation, the PZT patch is assumed to be bonded to the host structure at the four edges, as shown in Fig. 7.2. The interaction between the PZT and the host structure is represented by the force impedances of the host structure at the corresponding points. For a 2-D PZT-structure interacting system, the force impedance at a certain point contains the direct force impedances, Z_{11} and Z_{22} , as well as the cross impedances, Z_{12} and Z_{21} , in which the first and the second subscripts represent the direction of the applied force and the direction of the velocity response, respectively. Thus, the applied forces and the velocity responses at a certain point have the following relationship:

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{Bmatrix} \dot{u} \\ \dot{v} \end{Bmatrix} = \mathbf{Z} \begin{Bmatrix} \dot{u} \\ \dot{v} \end{Bmatrix} \quad (7.1)$$

Although the force impedance is point dependent, it has been demonstrated in the previous chapter that if the dimension of the PZT patch is very small compared with the scale of the host structure, the force impedances at the two bonding points of the PZT patch can be assumed to be the same. For the 2-D PZT-

structure interacting system, the assumption of $Z_{pq} = Z'_{pq}$, $p, q = 1, 2$ is also adopted so as to avoid complicated calculation.

Fig. 7.2. Generic two-dimensional PZT-structure interacting system

The constitutive equation for a 2-D PZT patch is written as:

$$\begin{Bmatrix} S_1 \\ S_2 \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u}{\partial x_p} \\ \frac{\partial v}{\partial y_p} \end{Bmatrix} = \frac{1}{\tilde{Y}_p^E} \begin{bmatrix} 1 & -\nu_p \\ -\nu_p & 1 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \end{Bmatrix} + \begin{Bmatrix} d_{31} \\ d_{32} \end{Bmatrix} E_3 \quad (7.2a)$$

$$D_3 = \{d_{31} \quad d_{32}\} \cdot \begin{Bmatrix} T_1 \\ T_2 \end{Bmatrix} + \tilde{\epsilon}_{33}^T E_3 \quad (7.2b)$$

where ν_p is the Poisson's ratio of the PZT material; and the other symbols are as defined in the previous chapter.

The in-plane motion equation of the PZT patch can be obtained from the constitutive equation and Newton's law:

$$\frac{\tilde{Y}_p^E}{1-\nu_p^2} \frac{\partial^2 u}{\partial x_p^2} + \frac{\tilde{Y}_p^E}{2(1-\nu_p)} \frac{\partial^2 v}{\partial x_p \partial y_p} + \frac{\tilde{Y}_p^E}{2(1+\nu_p)} \frac{\partial^2 u}{\partial y_p^2} = \rho_p \ddot{u} \quad (7.3a)$$

$$\frac{\tilde{Y}_p^E}{1-\nu_p^2} \frac{\partial^2 v}{\partial y_p^2} + \frac{\tilde{Y}_p^E}{2(1-\nu_p)} \frac{\partial^2 u}{\partial x_p \partial y_p} + \frac{\tilde{Y}_p^E}{2(1+\nu_p)} \frac{\partial^2 v}{\partial x_p^2} = \rho_p \ddot{v} \quad (7.3b)$$

Chapter 7: Damage Detection in Plate

To solve these complicated equations, Zhou et al (1996) neglected the terms of changing rate of the shear strain and obtained a satisfactory velocity response. Thus, this simplification is also adopted in this study so as to obtain a simple closed solution. Eqn. (7.3) is modified as:

$$\frac{\tilde{Y}_p^E}{1-\nu_p^2} \frac{\partial^2 u}{\partial x_p^2} = \rho_p \frac{\partial^2 u}{\partial t^2} \quad (7.4a)$$

$$\frac{\tilde{Y}_p^E}{1-\nu_p^2} \frac{\partial^2 v}{\partial y_p^2} = \rho_p \frac{\partial^2 v}{\partial t^2} \quad (7.4b)$$

Using the approach of separation of variables, the solution of Eqn. (7.4) can be written as:

$$u = (A \sin K_p x_p + B \cos K_p x_p) e^{j\omega t} \quad (7.5a)$$

$$v = (C \sin K_p y_p + D \cos K_p y_p) e^{j\omega t} \quad (7.5b)$$

where $K_p = \omega \sqrt{\frac{\rho_p (1-\nu_p^2)}{\tilde{Y}_p^E}}$ is the spatial frequency of oscillation; and A, B, C and D are the unknown coefficients which can be determined from the boundary conditions.

Eqn. (7.1) defines the relationship between the applied forces to the host structure and the velocity responses. Thus, the forces applied on the PZT patch can be expressed as:

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} = -\mathbf{Z} \begin{Bmatrix} \dot{u} \\ \dot{v} \end{Bmatrix} \quad (7.6)$$

where the minus sign indicates that the forces applied to the structure and to the patch are equal in magnitude but opposite in direction.

Consequently, the stresses at the boundary of the patch may be approximately expressed by the applied forces as:

$$\begin{Bmatrix} T_1|_{x_p=0} \\ T_2|_{y_p=0} \end{Bmatrix} = - \begin{Bmatrix} \frac{F_1|_{x_p=0}}{h_p b_p} \\ \frac{F_2|_{y_p=0}}{h_p a_p} \end{Bmatrix} = \frac{j\omega}{h_p} \begin{bmatrix} \frac{1}{b_p} & 0 \\ 0 & \frac{1}{a_p} \end{bmatrix} \cdot \mathbf{Z} \cdot \begin{Bmatrix} B \\ D \end{Bmatrix} e^{j\omega t} \quad (7.7a)$$

$$\begin{Bmatrix} T_1|_{x=a_p} \\ T_2|_{y=b_p} \end{Bmatrix} = \begin{Bmatrix} \frac{F_1|_{x=a_p}}{h_p b_p} \\ \frac{F_2|_{y=b_p}}{h_p a_p} \end{Bmatrix} = - \frac{j\omega}{h_p} \begin{bmatrix} \frac{1}{b_p} & 0 \\ 0 & \frac{1}{a_p} \end{bmatrix} \cdot \mathbf{Z} \cdot \begin{Bmatrix} A \sin(K_p a_p) + B \cos(K_p a_p) \\ C \sin(K_p b_p) + D \cos(K_p b_p) \end{Bmatrix} e^{j\omega t} \quad (7.7b)$$

In the above equations, the stresses have different signs because the applied forces cause tension and compression at the opposite edges, respectively.

Substituting Eqns. (7.5) and (7.7) into Eqn. (7.2a), the unknown coefficients A, B, C and D can be determined:

$$\begin{Bmatrix} B \\ D \end{Bmatrix} = \begin{bmatrix} -\tan\left(\frac{K_p a_p}{2}\right) & 0 \\ 0 & -\tan\left(\frac{K_p b_p}{2}\right) \end{bmatrix} \begin{Bmatrix} A \\ C \end{Bmatrix} = \mathbf{M} \cdot \begin{Bmatrix} A \\ C \end{Bmatrix} \quad (7.8)$$

$$\begin{Bmatrix} A \\ C \end{Bmatrix} = \mathbf{N}^{-1} \begin{Bmatrix} d_{31} \\ d_{32} \end{Bmatrix} \bar{E} = \begin{Bmatrix} A' \\ C' \end{Bmatrix} \bar{E} \quad (7.9)$$

$$\text{where } \mathbf{N} = \mathbf{I} - \frac{j\omega}{\tilde{Y}_p^E K_p h_p} \begin{bmatrix} \frac{Z_{11}}{b_p} - \nu_p \frac{Z_{21}}{a_p} & \frac{Z_{12}}{b_p} - \nu_p \frac{Z_{22}}{a_p} \\ -\nu_p \frac{Z_{11}}{b_p} + \frac{Z_{21}}{a_p} & -\nu_p \frac{Z_{12}}{b_p} + \frac{Z_{22}}{a_p} \end{bmatrix} \cdot \mathbf{M} \quad (7.10)$$

From the constitutive equation, Eqn. (7.2b), the electrical displacement is calculated as:

Chapter 7: Damage Detection in Plate

$$D_3 = \frac{\tilde{Y}_p^E e^{j\omega t}}{1 - \nu_p^2} \left[\left(d_{31} + \nu_p d_{32} \right) \left(\frac{A' K_p}{\cos \frac{K_p a_p}{2}} \cos K_p \left(x_p - \frac{a_p}{2} \right) - d_{31} \right) \right. \\ \left. + \left(\nu_p d_{31} + d_{32} \right) \left(\frac{C' K_p}{\cos \frac{K_p b_p}{2}} \cos K_p \left(y_p - \frac{b_p}{2} \right) - d_{32} \right) \right] \bar{E} + \tilde{\varepsilon}_{33}^T E_3 \quad (7.11)$$

The admittance is obtained as:

$$Y = \frac{I}{V} = \frac{\int \dot{D}_3 dA}{V} \\ = \frac{j\omega}{h_p} \left\{ \frac{\tilde{Y}_p^E}{1 - \nu_p^2} \left[\left(d_{31} + \nu_p d_{32} \right) \left(2A' b_p \tan \frac{K_p a_p}{2} - d_{31} a_p b_p \right) \right. \right. \\ \left. \left. + \left(\nu_p d_{31} + d_{32} \right) \left(2C' a_p \tan \frac{K_p b_p}{2} - d_{32} a_p b_p \right) \right] + \tilde{\varepsilon}_{33}^T a_p b_p \right\} \quad (7.12)$$

The impedance is the inverse of the admittance:

$$Z = \frac{1}{Y} \quad (7.13)$$

Substituting the solved coefficients, Eqn. (7.8), into the displacement distribution, Eqn. (7.5), it is observed that the displacements at the opposite edges of the PZT patch have the same magnitude but opposite directions:

$$\left\{ \begin{matrix} u|_{x_p=0} \\ v|_{y_p=0} \end{matrix} \right\} = \left\{ \begin{matrix} -A \tan \frac{K_p a_p}{2} e^{j\omega t} \\ -C \tan \frac{K_p b_p}{2} e^{j\omega t} \end{matrix} \right\} = - \left\{ \begin{matrix} u|_{x_p=a_p} \\ v|_{y_p=b_p} \end{matrix} \right\} \quad (7.14)$$

The forces applied on the host structure are multiplications of the force impedances and the velocities at the corresponding points:

Chapter 7: Damage Detection in Plate

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}_{x=x_1, y=y_1} = \mathbf{Z} \begin{Bmatrix} \dot{u} \\ \dot{v} \end{Bmatrix}_{x=x_1, y=y_1} \quad (7.15a)$$

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}_{x=x_2, y=y_2} = \mathbf{Z} \begin{Bmatrix} \dot{u} \\ \dot{v} \end{Bmatrix}_{x=x_2, y=y_2} \quad (7.15b)$$

Since the force impedances at the opposite edges are assumed to be the same, the output forces at the opposite edges of the PZT patch also have the same magnitude but opposite directions:

$$\begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}_{x_p=0, y_p=0} = - \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}_{x_p=a_p, y_p=b_p} = \begin{Bmatrix} \bar{F}_1 \\ \bar{F}_2 \end{Bmatrix} \quad (7.16)$$

Thus, the forces distribution on the host structure can be expressed as:

$$N_1 = \frac{\bar{F}_1}{b_p} [\delta(x-x_1) - \delta(x-x_2)] [H(y-y_1) - H(y-y_2)] \quad (7.17a)$$

$$N_2 = \frac{\bar{F}_2}{a_p} [\delta(y-y_1) - \delta(y-y_2)] [H(x-x_1) - H(x-x_2)] \quad (7.17b)$$

where x, y are the coordinate axes established for the host structure; x_1, x_2, y_1, y_2 are the coordinates of the PZT patch, as shown in Fig. 7.3; $\delta(x)$ is the dirac function; and $H(x)$ is the step function defined as:

$$H(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases} \quad (7.18)$$

Fig. 7.3. Plate bonded with a pair of PZT patches

Chapter 7: Damage Detection in Plate

Moreover, the subtraction of Eqn. (7.15a) by Eqn. (7.15b) obtains:

$$\begin{Bmatrix} 2\bar{F}_1 \\ 2\bar{F}_2 \end{Bmatrix} = \mathbf{Z} \begin{Bmatrix} \dot{u}|_{x=x_1} - \dot{u}|_{x=x_2} \\ \dot{v}|_{y=y_1} - \dot{v}|_{y=y_2} \end{Bmatrix} \quad (7.19)$$

The vibration of the structure can be determined by analytical, numerical or experimental methods. Thus, the velocity responses of the structure under the output forces of the PZT patches can be obtained, which can be expressed as:

$$\begin{Bmatrix} \dot{u}|_{x=x_1} - \dot{u}|_{x=x_2} \\ \dot{v}|_{y=y_1} - \dot{v}|_{y=y_2} \end{Bmatrix} = \mathbf{Q} \cdot \begin{Bmatrix} \bar{F}_1 \\ \bar{F}_2 \end{Bmatrix} \quad (7.20)$$

The force impedance of the structure can then be calculated by:

$$\mathbf{Z} = 2\mathbf{Q}^{-1} \quad (7.21)$$

Similar to the 1-D problems, no restriction has been implemented on the host structure. Eqns. (7.12) and (7.13) are generic formulation to predict the electrical impedance of the PZT transducer. Eqn. (7.21) can be adopted to calculate the force impedance of general 2-D structures.

7.2.2 Modeling of host structure

In this study, a thin plate is selected to represent the 2-D structure for damage detection. The length, width and thickness of the plate are denoted as a , b and h , respectively. Two PZT transducers are bonded to the plate at the same location but on the opposite sides, as shown in Fig. 7.3. The two patches are subjected to opposite electrical fields so that only pure bending vibration is excited in the plate. The displacements in the plate are assumed to be:

$$\bar{w} = w(x, y, t) \quad (7.22a)$$

$$\bar{u} = -z \frac{\partial w}{\partial x} \quad (7.22b)$$

$$\bar{v} = -z \frac{\partial w}{\partial y} \quad (7.22c)$$

The vibration of the plate is analyzed by Ritz method with polynomial shape functions. Various damages may occur in the structure. However, only the effect of the damage on structural vibration is considered. The damage in the plate is simulated by a damaged element with a reduced Young's modulus but with the same thickness as the plate. The potential and kinetic energy of the plate with a damaged element can be expressed as:

$$U = \frac{Eh^3}{24(1-\nu^2)} \int_A \left[\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right)^2 - 2(1-\nu) \left[\frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} - \left(\frac{\partial^2 w}{\partial x \partial y} \right)^2 \right] \right] dA$$

$$- \frac{(E - E_d)h^3}{24(1-\nu^2)} \int_{A_d} \left[\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right)^2 - 2(1-\nu) \left[\frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} - \left(\frac{\partial^2 w}{\partial x \partial y} \right)^2 \right] \right] dA \quad (7.23)$$

$$T = \frac{1}{2} \rho h \int_A \dot{w}^2 dA \quad (7.24)$$

where E is the Young's modulus; the subscript d represents the damage; ν is the Poisson's ratio; and A_d is the damaged area in the plate.

For a small damage, the potential energy is approximated as:

$$U = \frac{Eh^3}{24(1-\nu^2)} \int_A \left[\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right)^2 - 2(1-\nu) \left[\frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} - \left(\frac{\partial^2 w}{\partial x \partial y} \right)^2 \right] \right] dA$$

$$- \frac{(E - E_d)h^3}{24(1-\nu^2)} \left[\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right)^2 - 2(1-\nu) \left[\frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} - \left(\frac{\partial^2 w}{\partial x \partial y} \right)^2 \right] \right] \bigg|_{x=x_c, y=y_c} \cdot A_d \quad (7.25)$$

where x_c and y_c are the coordinates of the center of the damage.

Without loss of generality, the plate with various shapes in the x - y coordinate system can be mapped to a plate of size 2×2 in the ξ - η coordinate system. For the rectangular plate investigated in this study, the transformation mechanism is straightforward and can be defined as:

$$\xi = \frac{2x}{a} - 1 \quad (7.26a)$$

Chapter 7: Damage Detection in Plate

$$\eta = \frac{2y}{b} - 1 \quad (7.26b)$$

Thus, Eqns. (7.25) and (7.24) may be rewritten as:

$$U = \frac{Eh^3}{24(1-\nu^2)} \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \left\{ \left(\frac{4}{a^2} \frac{\partial^2 w}{\partial \xi^2} + \frac{4}{b^2} \frac{\partial^2 w}{\partial \eta^2} \right)^2 - 2(1-\nu) \frac{16}{a^2 b^2} \left[\frac{\partial^2 w}{\partial \xi^2} \frac{\partial^2 w}{\partial \eta^2} - \left(\frac{\partial^2 w}{\partial \xi \partial \eta} \right)^2 \right] \right\} d\xi d\eta - \frac{(E - E_d)h^3}{24(1-\nu^2)} \left\{ \left(\frac{4}{a^2} \frac{\partial^2 w}{\partial \xi^2} + \frac{4}{b^2} \frac{\partial^2 w}{\partial \eta^2} \right)^2 - 2(1-\nu) \frac{16}{a^2 b^2} \left[\frac{\partial^2 w}{\partial \xi^2} \frac{\partial^2 w}{\partial \eta^2} - \left(\frac{\partial^2 w}{\partial \xi \partial \eta} \right)^2 \right] \right\} \bigg|_{\substack{\xi=\xi_c \\ \eta=\eta_c}} \cdot A_d \quad (7.27)$$

$$T = \frac{1}{2} \rho h \frac{ab}{4} \int_{-1}^1 \int_{-1}^1 \dot{w}^2 d\xi d\eta \quad (7.28)$$

The transverse displacement is approximated as a combination of a set of shape functions:

$$w = \sum_{q=0}^N \sum_{i=0}^q c_m \phi_m e^{j\omega t} = \mathbf{C} \boldsymbol{\phi} e^{j\omega t} \quad (7.29)$$

where $m = \frac{(q+1)(q+2)}{2} - i$; N is the degree set of polynomial space; $\mathbf{C}$ is the coefficient vector; $\boldsymbol{\phi}$ is the shape function vector; and $e^{j\omega t}$ stands for harmonic vibration.

As in Chapter 4, the 2-D polynomial shape functions for plate are defined as:

$$\phi_m = \xi^i \eta^{q-i} \phi_0, \quad m = 1, 2, \dots, \frac{(N+1)(N+2)}{2} \quad (7.30)$$

$$\phi_0 = \prod_p [\Gamma_p(\xi, \eta)]^{q_p} \quad (7.31)$$

where Γ_p is the boundary equation of the p th supporting edge in the $\xi - \eta$ plane and the value of $\mathcal{G}_p$ is selected from Table 4.1 to satisfy the boundary condition at the corresponding edge.

The stiffness and mass matrix can then be determined:

$$K_{mn} = \frac{\partial^2 U}{\partial c_m \partial c_n} = \frac{Eabh^3}{3(1-\nu^2)} \int_{-1}^1 \int_{-1}^1 \varphi_{mn} d\xi d\eta - \frac{4(E-E_d)h^3}{3(1-\nu^2)} \varphi_{mn} \Big|_{\substack{\xi=\xi_c \\ \eta=\eta_c}} A_d \quad (7.32)$$

$$M_{mn} = \frac{1}{\omega^2} \frac{\partial^2 T}{\partial c_m \partial c_n} = \frac{ab}{4} \rho h \int_{-1}^1 \int_{-1}^1 \phi_m \phi_n d\xi d\eta \quad (7.33)$$

where

$$\begin{aligned} \varphi_{mn} = & \left(\frac{1}{a^2} \frac{\partial^2 \phi_m}{\partial \xi^2} + \frac{1}{b^2} \frac{\partial^2 \phi_m}{\partial \eta^2} \right) \left(\frac{1}{a^2} \frac{\partial^2 \phi_n}{\partial \xi^2} + \frac{1}{b^2} \frac{\partial^2 \phi_n}{\partial \eta^2} \right) \\ & - (1-\nu) \frac{1}{a^2 b^2} \left[\frac{\partial^2 \phi_m}{\partial \xi^2} \frac{\partial^2 \phi_n}{\partial \eta^2} + \frac{\partial^2 \phi_m}{\partial \eta^2} \frac{\partial^2 \phi_n}{\partial \xi^2} - 2 \frac{\partial^2 \phi_m}{\partial \xi \partial \eta} \frac{\partial^2 \phi_n}{\partial \xi \partial \eta} \right] \end{aligned} \quad (7.34)$$

and $m, n = 0, 1, \dots, \frac{(N+1)(N+2)}{2}$

The natural frequencies and modal shapes can be calculated by solving the eigenvalue equation

$$(\mathbf{K} - \mathbf{M}\lambda)\mathbf{C} = 0 \quad (7.35)$$

and obtained as:

$$\omega_k = \sqrt{\lambda_k} \quad (7.36)$$

$$\Phi_k = \sum_m c_{mk} \phi_m = \mathbf{C}_k^T \Phi \quad (7.37)$$

where λ_k is the k th eigenvalue; $\mathbf{C}_k$ is the eigenvector corresponding to λ_k ; and the superscript T denotes transpose.

The exciting moment distribution provided by a pair of PZT patch is

$$M_1 = \frac{\bar{F}_1(h+h_p)}{b_p} [\delta(x-x_1) - \delta(x-x_2)] \cdot [H(y-y_1) - H(y-y_2)] \quad (7.38a)$$

Chapter 7: Damage Detection in Plate

$$M_2 = \frac{\bar{F}_2(h+h_p)}{a_p} [\delta(y-y_1) - \delta(y-y_2)] \cdot [h(y-y_1) - h(y-y_2)] \quad (7.38b)$$

Assuming that the forced vibration of the plate is a superposition of the vibration modes,

$$w = \sum_k \gamma_k \Phi_k e^{j\omega t} \quad (7.39)$$

Based on the orthogonality of the modal shapes, the modal participation factor can be calculated as:

$$\gamma_k = \frac{G_k}{\omega_k^2 - \omega^2} \quad (7.40)$$

where $G_k = G_k^x + G_k^y$ (7.41a)

$$\begin{aligned} G_k^x &= \frac{\int_0^a \int_0^b \mathbf{C}_k^T \boldsymbol{\Phi} \left(\frac{\partial M_1}{\partial x} \right) dx dy}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k} \\ &= \frac{\frac{h+h_p}{b_p} \frac{b}{a} \sum_m \left[c_{mk} \int_{\eta_1}^{\eta_2} \left(\frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_2} - \frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_1} \right) d\eta \right]}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k} \cdot \bar{F}_1 \end{aligned} \quad (7.41b)$$

$$\begin{aligned} G_k^y &= \frac{\int_0^a \int_0^b \mathbf{C}_k^T \boldsymbol{\Phi} \left(\frac{\partial M_2}{\partial y} \right) dx dy}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k} \\ &= \frac{\frac{h+h_p}{a_p} \frac{a}{b} \sum_m \left[c_{mk} \int_{\xi_1}^{\xi_2} \left(\frac{\partial \phi_m}{\partial \eta} \Big|_{\eta=\eta_2} - \frac{\partial \phi_m}{\partial \eta} \Big|_{\eta=\eta_1} \right) d\xi \right]}{\mathbf{C}_k^T \mathbf{M} \mathbf{C}_k} \cdot \bar{F}_2 \end{aligned} \quad (7.41c)$$

The elements of $\mathbf{Q}$ in Eqn. (7.20) can be calculated:

$$Q_{11} = j\omega \frac{(h+h_p)^2}{ab_p} \sum_k \left\{ \frac{G_k^x}{\omega_k^2 - \omega^2} \sum_m \left[c_{mk} \left(\frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_2, \eta=\eta_c} - \frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_1, \eta=\eta_c} \right) \right] \right\} \quad (7.42a)$$

$$Q_{12} = j\omega \frac{(h+h_p)^2}{aa_p} \sum_k \left\{ \frac{G_k^y}{\omega_k^2 - \omega^2} \sum_m \left[c_{mk} \left(\frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_2, \eta=\eta_c} - \frac{\partial \phi_m}{\partial \xi} \Big|_{\xi=\xi_1, \eta=\eta_c} \right) \right] \right\} \quad (7.42b)$$

Chapter 7: Damage Detection in Plate

$$Q_{21} = j\omega \frac{(h + h_p)^2}{bb_p} \sum_k \left\{ \frac{G_k^x}{\omega_k^2 - \omega^2} \sum_m \left[c_{mk} \left(\frac{\partial \phi_m}{\partial \eta} \Big|_{\substack{\xi=\xi_c \\ \eta=\eta_2}} - \frac{\partial \phi_m}{\partial \xi} \Big|_{\substack{\xi=\xi_c \\ \eta=\eta_1}} \right) \right] \right\} \quad (7.42c)$$

$$Q_{22} = j\omega \frac{(h + h_p)^2}{ba_p} \sum_k \left\{ \frac{G_k^y}{\omega_k^2 - \omega^2} \sum_m \left[c_{mk} \left(\frac{\partial \phi_m}{\partial \eta} \Big|_{\substack{\xi=\xi_c \\ \eta=\eta_2}} - \frac{\partial \phi_m}{\partial \eta} \Big|_{\substack{\xi=\xi_c \\ \eta=\eta_1}} \right) \right] \right\} \quad (7.42d)$$

The force impedance of the plate can then be determined from Eqn. (7.21) and the admittance of the PZT patch can be calculated by substituting Eqns. (7.10) and (7.9) into Eqn. (7.12).

7.2.3 Model verification

A lab-sized aluminum plate is tested to verify the 2-D impedance model. The configuration of the specimen is shown in Fig. 7.3. The same bonding process as for the beam specimen is adopted. The properties of the PZT patches and the plate specimen are listed in Tables 7.1 and 7.2, respectively.

Table 7.1. Properties of PZT patch

Dimension (mm)	Location (mm)	Young's Modulus (GPa)	Loss Factor	Mass Density (kg/m ³)	Strain Constant <i>d</i> (m/volt)	Permittivity (farad/m)	Dielectric Loss Factor
<i>a_p</i> =10 <i>b_p</i> =10 <i>h_p</i> =0.2	<i>x</i> ₁ =30 <i>x</i> ₂ =40 <i>y</i> ₁ =70.5 <i>y</i> ₂ =80.5	66.7	0.005	7800	-2.10E-10	1.93E-08	1.50E-02

Table 7.2. Properties of plate specimen

Length (mm)	Width (mm)	Thickness (mm)	Young's Modulus (GPa)	Poisson's Ratio	Mass Density (kg/m ³)	Damping's Ratio
101	101	2	66.5	0.33	2800	0.005

Chapter 7: Damage Detection in Plate

Fig. 7.4 pictured the tested undamaged plate specimen. The specimen rested on soft papers is represented by a free plate model in numerical simulation. For each PZT patch, two electrical wires are soldered on the two electrodes. The wires are subsequently connected to a channel in the multiplex module. For convenience of control, the label attached to the electrical wires indicates which channel is connected. Through the switch control system, opposite electrical fields are applied to the two PZT patches so that the output forces of the two patches have the same magnitude but opposite directions, thus, pure bending vibration is actuated in the plate specimen.

Fig. 7.4. Undamaged plate specimen

The electrical admittance spectra of the PZT patches are measured by the impedance analyzer with a frequency range from 5KHz to 30KHz. To increase the sensitivity of the admittance spectra to the damage, the interval of the scanning frequency is selected as 1Hz.

With the degree set of the polynomial shape function as $N = 20$, Eqns. (7.42) and (7.21) are employed to calculate the force impedance of the plate specimen. The electrical admittance is then predicted by Eqn. (7.12). The real part

Chapter 7: Damage Detection in Plate

of the admittance is compared with the experimentally measured results, as shown in Fig. 7.5. The two curves are close to each other. Good agreement between them proves that the developed generic 2-D impedance model can predict the E/M admittance with satisfactory precision. The small differences between them could be attributed to the following modeling errors:

- the material of the structure is assumed to be homogeneous and isotropic;
- the specimen is represented by a Kirchhoff plate model;
- the vibration of the plate is approximated by the Ritz method;
- the boundaries are assumed to be totally free;
- discrepancies exist between the assumed parameters and the exact properties of the PZT and the host structure; and
- the effect of the adhesive layer (epoxy) has not been considered in this model.

Fig. 7.5. Experimentally measured and numerically predicted admittance spectra for undamaged plate

As demonstrated in the previous chapter, the inEPIB NDE technique is based on the peaks of the admittance spectra. Table 7.3 thus lists the frequencies of

Chapter 7: Damage Detection in Plate

the admittance peaks obtained from the experimental measurements and the numerical prediction. A very good agreement between them is observed.

Table 7.3. EM admittance peaks and natural frequencies (KHz) for plate

Natural frequencies	Measured peaks	Predicted peaks	Natural frequencies	Measured peaks	Predicted peaks
5.417	5.465	5.401	16.819	16.672	16.741
5.696	5.762	5.672	18.188	17.988	18.082
6.083	6.126	6.068	19.300	19.189	19.214
7.039	7.029	6.996	19.390	/	/
7.442	7.46	7.427	19.915	19.777	19.889
7.796	7.862	7.784	20.822	20.564	20.806
9.199	9.257	9.165	21.218	20.982	21.123
9.426	9.459	9.403	21.318	/	/
9.937	9.902	9.844	21.635	21.356	21.577
9.942	/	/	23.486	23.110	23.389
11.238	11.263	11.230	23.881	23.530	23.808
12.973	12.937	12.970	25.483	25.087	25.364
13.476	13.404	13.388	25.548	/	/
13.614	/	/	25.837	/	/
13.811	13.969	13.890	26.055	25.667	25.983
13.946	/	/	27.091	26.536	27.045
14.223	14.239	14.178	27.389	26.896	27.329
15.569	/	/	27.822	27.340	27.792
15.991	/	/	30.099	29.305	29.883

It is observed that each E/M admittance peak corresponds to a natural frequency of the structure. This indicates that the E/M admittance spectra recorded by the PZT transducers reflect the mechanical property of the host structure. Since the size and weight of the PZT transducers are very small compared with the host structure, the differences between the E/M admittance peaks and the structural natural frequencies are very small, which indicates that the PZT transducers bonded

to the host structure would not notably modify the structural dynamics, and the inEPIB technique is a non-invasive damage detection technique. Moreover, the small and light PZT patch can be permanently bonded on the host structure for on-line health monitoring.

Similar to the beam specimen, since the PZT patch happens to be located on the nodal line of some vibration modes, these modes can not be detected by the PZT transducer.

7.3 DAMAGE DETECTION

A hole is drilled in the plate specimen to simulate the damage. The center of the hole locates at $x_c = 80.4mm$ and $y_c = 21.0mm$. The radius of the hole is measured as $r_d = 2.8mm$.

For the damaged plate, Fig. 7.6 shows the experimentally measured and numerically predicted E/M admittance spectra. The corresponding frequencies of the admittance peaks in experiment and prediction are listed in Table 7.4 for the damaged specimen. By comparing them with Table 7.3, it is apparent that the damage has caused shifts of the admittance peaks both in experimental measurements and numerical prediction, as expected. Those shifts contain the information of the damage, and will be used for damage detection in inEPIB NDE technique.

The inEPIB technique uses a fitness function, which is reiterated here:

$$f(\mathbf{D}) = \sum_i \left| [P_i^p(\mathbf{D}) - P_i^p(\mathbf{D} = 0)] - [P_i^e - P_0^e] \right| \quad (7.43)$$

to sum up the differences between the peak shifts in experiment and prediction, and to assess the damage.

Chapter 7: Damage Detection in Plate

Fig. 7.6. Experimentally measured and numerically predicted admittance spectra for damaged plate

Table 7.4. E/M admittance peaks (KHz) for damaged plate

Measured		Predicted		Measured		Predicted	
Peaks	Shifts	Peaks	Shifts	Peaks	Shifts	Peaks	Shifts
5.452	0.013	5.39	0.011	17.958	0.03	18.059	0.023
5.735	0.027	5.649	0.023	19.182	0.007	19.207	0.007
6.102	0.024	6.043	0.025	19.749	0.028	19.886	0.003
7.018	0.011	6.987	0.009	20.548	0.016	20.733	0.073
7.458	0.002	7.426	1E-3	20.969	0.013	21.115	0.008
7.864	-0.002	7.78	0.004	21.326	0.03	21.544	0.033
9.254	0.003	9.165	0	23.105	0.005	23.387	0.002
9.443	0.016	9.389	0.014	23.49	0.04	23.776	0.032
9.891	0.011	9.821	0.023	25.01	0.077	25.345	0.019
11.263	0	11.227	0.003	25.656	0.011	25.948	0.035
12.915	0.022	12.928	0.042	26.488	0.048	27.039	0.006
13.397	0.007	13.385	0.003	26.897	-1E-3	27.329	0
13.962	0.007	13.889	1E-3	27.332	0.008	27.774	0.018
14.208	0.031	14.159	0.019	29.296	0.009	29.878	0.005
16.655	0.017	16.705	0.036	29.618	0.03	30.453	0.038

Fig. 7.7 compares the shifts in experiment and prediction by the same damage. The similarity between the two curves indicates that Eqn. (7.43) has minimized the effect of modeling errors in damage identification and is suitable for fitness assessment, as previously elaborated in Section 6.3.

Fig. 7.7. Experimentally measured and numerically predicted admittance peak shift in plate

Evolutionary programming (EP) is employed by the inEPIB technique to search in the possible solutions and to minimize the value of Eqn. (7.43). Three variables are contained in EP chromosome. $\frac{x_c}{a}$ and $\frac{y_c}{a}$ are used to locate the damage, and $\frac{r_d}{a}$ is used to quantify the damage. All variables are non-dimensionalized for convenience. The constraints for the variables are $0 < \frac{x_c}{a} < 1$, $0 < \frac{y_c}{a} < 1$ and $0 < \frac{r_d}{a} < 0.1$ for a minor damage.

Chapter 7: Damage Detection in Plate

The evolutionary process is illustrated in Fig. 7.8. The decreasing fitness indicates that the algorithm is finding a solution which is getting closer and closer to the exact solution.

Fig. 7.8. Evolutionary process for damage detection in plate

After 20 generations, at the cost of 2000 function evaluations, the inEPIB technique located the damage at $\frac{x_c}{a} = 0.83574$ and $\frac{y_c}{a} = 0.21079$, i.e., $x_c = 84.41\text{mm}$ and $y_c = 21.29\text{mm}$, while the exact location is at $x_c = 80.4\text{mm}$, $y_c = 21.0\text{mm}$. The distance between the predicted location and the actual damage center is 4.02mm , which is less than 4% of the dimension of the structure. The radius of the hole is also found as $\frac{r_d}{a} = 0.02558$, $r_d = 2.58\text{mm}$, which is only 7.7% smaller than the exact value.

7.4 RELIABILITY ANALYSIS

The reliability of the inEPIB technique is also investigated. The measured E/M admittance spectra ranged from 5KHz to 7, 10, 15 and 20KHz which contains

Chapter 7: Damage Detection in Plate

4, 6, 9, 14 and 18 peaks, respectively, are used as input data to detect the damage. The optimized results after 100 generations are listed in Table 7.5 for comparison.

Table 7.5. Identified damage for various ranges of actuating frequency

	$\frac{x_c}{a}$	$\frac{y_c}{a}$	$\frac{r_d}{a}$	Error in location (mm)	Error in radius (mm)
Exact solution	0.79600	0.20790	0.02772	-	-
5~7KHz (including 4 peaks)	0.71077	0.27084	0.04147	11.26 (11.1%)*	1.39 (49.6%)#
5~10KHz (including 9 peaks)	0.82296	0.21711	0.02699	2.87 (2.8%)*	0.07 (2.5%)#
5~15KHz (including 14 peaks)	0.81553	0.21587	0.02675	2.13 (2.1%)*	0.10 (3.6%)#
5~20KHz (including 18 peaks)	0.79150	0.16588	0.02825	4.27 (4.2%)*	0.05 (1.8%)#
5~30KHz (including 29 peaks)	0.85008	0.20642	0.02544	5.46 (5.4%)*	0.23 (8.2%)#

* is the relative error of the optimized damage location, which is calculated by:

$$\frac{\text{Distance to the exact location}}{\text{Length of the plate}} \times 100\%$$

is the relative error of the optimized radius, which is calculated by:

$$\frac{\text{Discrepancy to the exact radius}}{\text{Exact radius}} \times 100\%$$

The following findings are obtained from Table 7.5:

1. When the input data ranged from 5KHz to 7KHz, the results are not satisfactory. This is because only 4 peaks are involved in this frequency range. Since the optimized results are sensitive to the measurement, any accidental errors in the experiment will significantly affect the results. Limited input data are not enough to shade the effect of these unavoidable accidental errors, and thus lead to unsatisfactory results.
2. Acceptable results are obtained from the E/M impedance spectra from 5KHz

to 10, 15, 20 and 30KHz, which proves that the proposed method is able to effectively and reliably find the location and quantity of the damage.

3. The accuracy of the backcalculated results reduces when the peaks in the range higher than 20KHz are used. This can be explained by the inaccuracy in structural analysis. In this research, Ritz method is employed to calculate the force impedance of the structure. The accuracy of structural analysis depends on the number of terms in the shape functions. With limited number of terms, the method can not precisely calculate the E M admittance peaks in the higher frequency range. This inaccuracy in structural analysis can be alleviated by introducing more terms in the shape functions, which will consequently require higher computing cost.

7.5 SUMMARY

A generic 2-D impedance model is developed. The PZT patch is assumed to be bonded at its edges on the host structure. The interaction between the transducer and the structure is represented by the force impedance of the host structure. By investigating the vibration of the PZT transducer and the host structure, a closed form impedance formulation is obtained.

The generic 2-D impedance model is applied to a plate structure. The predicted admittance spectra agree well with the experimentally measured results. More importantly, the impedance model can precisely predict the variations in admittance spectra caused by damage.

Damage in the plate is identified by the inEPIB technique. The relative error of the identified damage is less than 4% for location and 8% for radius, which proves that the inEPIB technique can effectively locate and quantify the damage simultaneously. Owing to the high search efficiency of the hybrid EP algorithm, the damage in the plate structure can be detected in less than 20 generations.

Chapter 7: Damage Detection in Plate

The reliability of the inEPIB technique has also been investigated. With the measured E/M admittance spectra in various frequency ranges, the damage can be effectively and reliably identified.

CHAPTER 8 CONCLUSIONS AND FUTURE WORK

8.1 SUMMARY

Structural parameter identification and damage detection techniques generally measure the structural responses and employ an optimization method to backcalculate the mechanical properties and damages. Due to the large search space and numerous local optima, the traditional local optimization strategies may not be able to find the correct global optimum. The main objective of this study is to develop a general approach based on EP as a backcalculation algorithm for structural nondestructive evaluation. The EP-based backcalculation algorithm is integrated with the impedance-based NDE technique to develop an inEPIB technique. The inEPIB technique uses piezoelectric wafers as transducers to collect the information of structure and the EP-based algorithm to backcalculate the location and quantity of damages in structures.

The present research work emphasizes on the following tasks:

- (1) an EP algorithm hybridized with local search operators to improve search efficiency;
- (2) the application of EP-based backcalculation algorithm to parameter identification of composite plates; and
- (3) the systematic development and implementation of the inEPIB technique for damage detection in beams and plates.

Chapter 8: Conclusions and Future Work

The hybrid EP algorithm has been presented in Chapter 3. The algorithm improves the search efficiency of the traditional EP method without loss of effectiveness in finding the global optimum. The high search efficiency and effectiveness enhance the feasibility and practicality of applying EP-based backcalculation algorithm for structural nondestructive evaluation.

The EP-based backcalculation algorithm is employed in Chapter 4 to identify the elastic constants of laminated composite plates. The effect of human factors has been minimized in the identification process. The good results show that the EP-based backcalculation algorithm is promising for other inverse problem solving in civil engineering.

The inEPIB NDE technique presented in Chapters 5, 6 and 7 includes state-of-concept analysis, impedance model development and experimental verification. The inEPIB technique has been implemented for 1-D beams and 2-D plates. A special fitness function is proposed to exclude the effect of modeling errors. The inEPIB program code is written in C++ language, which contains an EP kernel module for flow control and a structural analysis module for fitness calculation. For different types of structures, only the structural analysis module needs to be modified. The inEPIB technique inherits the advantages of the traditional impedance-based method but overcomes the limitations, i.e., both the location and quantity of damage can be identified simultaneously.

8.2 CONCLUSIONS

This study has demonstrated that EP, as an effective global optimization method, is a feasible tool for inverse problem solving in civil engineering. The inEPIB technique developed in this study has shown some advantages over the traditional methods. It could be a promising technique for continuous health monitoring of structures in service.

The present work led to the following conclusions:

- (1) The EP-MMLSX algorithm improves the search efficiency of the traditional EP technique without loss of effectiveness in finding the global optimum. The algorithm employs macro mutation and local linear bisection search operators to explore and exploit the search space, respectively. The crossover operator retains the diversity of the generation and prevents the evolution from stagnating in later generations. The high search efficiency enhances the practicality of using EP for inverse problem solving in civil engineering. Since the algorithm has not incorporated any domain-specific knowledge, it provides a general algorithm for parameter optimization problems.
- (2) The EP-based backcalculation algorithm is a feasible technique for inverse problem solving in civil engineering. With the EP-based algorithm, the elastic parameters of composite plates have been backcalculated from the measured natural frequencies. The agreement between these results and those published in the literature demonstrates that the algorithm can be practically applied to parameter identification problems. Moreover, the effect of human factors has been minimized during the identification process. Since the EP-based algorithm is independent of the problem domains, it can find great applications in a wide variety of engineering problems, such as damage detection, structural design and other parameter optimization problems.
- (3) The inEPIB technique is able to simultaneously locate and quantify the damage in structures. The generic impedance models developed for 1-D and 2-D PZT-structure interacting systems can precisely predict the positions of the admittance peaks for both undamaged and damaged structures. The development of the generic impedance models provides the possibility of detecting damage in various structures using the inEPIB technique. Furthermore, the inEPIB technique compares the

Chapter 8: Conclusions and Future Work

predicted admittance peaks with the measured data for the undamaged structure to minimize the effect of the modeling errors. This is essential for the technique to be practically applied to real structures. In addition, the traditional impedance-based NDE technique utilizes signals at very high frequency (typically higher than 30KHz). The excited elastic waves diminish in the structures because of damping. The area of sensing region depends on the propagation distance of the excited vibration wave. Since the damping is more significant at high frequencies (Sun et al 1995), the lower the excited frequency, the larger the sensing region. In this thesis, damages are successfully detected from signals collected at frequencies lower than 30KHz. Thus, compared with the traditional impedance method, the sensing region of the transducer can be enlarged. Furthermore, the low frequency vibration has also eased the task of structural analysis.

- (4) Although the inEPIB technique has so far been applied to beam and plate only, the range of applications goes beyond this class of structures. The impedance models are developed with no restriction being imposed on the structure. It can be applied to other 1-D or 2-D structures. On the other hand, the EP-based algorithm is adaptive to the optimization problem. Thus, the inEPIB technique can be conveniently employed to detect damage in other structures with suitable structural analysis module.

8.3 FUTURE WORK

The principal areas of future work can be divided into the following main parts:

- (1) To improve the backcalculation algorithm. The characteristic that make EP robust also makes it computation-intensive. This study has done

Chapter 8: Conclusions and Future Work

some work to improve the search efficiency. However, it could be further improved so that the inEPIB technique can detect the damage at the shortest possible time to provide real-time monitoring of the structures. The efficiency of the algorithm can be improved by incorporating domain-specific knowledge, employing new operators or hybridizing with other strategies.

- (2) To increase the sensitivity of the inEPIB technique to detect minor damage. The inEPIB technique uses the admittance spectra at lower frequency to increase the sensing area. However, this will also decrease the sensitivity to the damage. One possible solution is to employ not only the admittance peaks but also the amplitude of the backcalculate the damage, e.g, the fitness function can be changed as:

$$f(\mathbf{D}) = \sum_i | [Y_i^p(\mathbf{D}) - Y_i^e] - [Y_i^p(\mathbf{D} = 0) - Y0_i^e] | \quad (8.1)$$

where Y is the admittance; $Y0$ is the measured data for the undamaged structure; $\mathbf{D}$ is the damage, and the superscripts p and e represent numerical prediction and experimental measurements, respectively. Thus, more information could be interpreted from the measurement. For further analysis of the sensitivity, more experiments and numerical simulation must be conducted.

- (3) To consider various types and multiple damages. In this study, only one damage, which was simulated as a hole, was induced in the specimens. In future, multiple damages or damages with different types should be considered. The type of the damage can be represented by an additional variable contained in the EP chromosome. The increase of damage number also introduces new variables to the optimization problem. However, EP has been proven to be a robust optimization technique for problems with numerous variables. Thus, with suitable structural analysis module, the inEPIB technique has the potential for various types and multiple damages detection.

Chapter 8: Conclusions and Future Work

- (4) To investigate the effect of changing boundary conditions and ambient temperature. For real structures, the boundary conditions and ambient temperature could change during the monitoring process. It is important for the inEPIB technique to distinguish the changes in admittance spectra caused by damage from those caused by other reasons. The measured signature must be "repaired" according to the changing conditions (Park et al. 1999) before it is used for backcalculation.
- (5) To apply the inEPIB technique on real structures with complex geometry. This thesis has described the basic concept of the technique. The application includes two essential modules, the optimization algorithm and the calculation of the E/M impedance. When it is applied to a real structure, the difficulty lies in that the theoretical solution for the force impedance, thus the E/M impedance, is generally not available. Numerical methods, such as FEM, have to be employed. Fortunately, most of the commercial FEM packages, e.g., ANSYS, have convenient input/output interfaces so that certain FEM package can work as a function or subroutine in the inEPIB program. An APDL (ANSYS parametric design language) file can be included in the inEPIB program to create models corresponding to each chromosome and to calculate the force impedances automatically. Thus, by combining with such a general-purpose FEM package, the inEPIB technique can be applied to real structures.

PUBLICATIONS

1. Xu, J. F., Yang, Y. W., and Soh, C. K. (2002). "A hybrid EP with macro and local mutation and crossover." Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, 18-22 Nov., Singapore, 225-229.
2. Xu, J. F., Yang, Y. W., and Soh, C. K. (2004), "E/M Impedance-Based Structural Health Monitoring with Evolutionary Programming", Journal of Aerospace Engineering, ASCE, will be appeared in the October 2004 issue.
3. Yang, Y. W., Xu, J. F., and Soh, C. K., (2004), "A Generic Impedance-Based Model for Structure - PZT interacting System", Journal of Aerospace Engineering, ASCE, Accepted.
4. Yang, Y. W., Xu, J. F., and Soh, C. K., (2004), "An evolutionary programming algorithm for continuous global optimization", European Journal of Operational Research, Accepted.

REFERENCES

- Abbass, H. A. (2002), "An evolutionary artificial neural networks approach for breast cancer diagnosis", Artificial Intelligence in Medicine, Vol. 25, pp. 265-281.
- Abd, E.-G. K. M., and Farag, M. M. (2002), "AutoNDT: new software for remote ultrasonic scanning via the internet", NDT&E International, Vol. 35, Issue 1, pp. 1-8.
- Angeline, P. J., Reynolds, R. G., McDonnell, J. R., and Eberhart, R. C. (Editors) (1997), Proceedings of the 6th Annual Conference on Evolutionary Programming, Indianapolis, Indiana, USA.
- Ashton, J. E., and Anderson, J. D. (1969), "The natural modes of vibration of boron-epoxy plates", Shock and Vibration Bulletin, Vol. 39, pp. 81-91.
- Ayres, J., Lalande, F., Chaudhry, Z., and Rogers, C. A. (1996), "Qualitative health monitoring of a steel bridge structure via piezoelectric actuator/sensor patches", Proceedings of SPIE Nondestructive Evaluation Techniques for Aging Infrastructure and Manufacturing, Scottsdale, AZ, Vol. 2946, pp. 211-218.
- Back, T. (1996), Evolutionary Algorithms in Theory and Practice: Evolution strategies, Evolutionary programming, Genetic algorithms. Oxford University Press, New York.
- Back, T., and Schwefel, H. -P. (1993), "An overview of evolutionary algorithms for parameter optimisation", Evolutionary Computation, Vol. 1, No. 1, pp. 1-23.
- Back, T., Fogel, D. B., and Michalewicz, Z. (1997), Handbook of Evolutionary Computation, Oxford University Press, New York.

References

-
-
- Back, T., Rudolph, G., and Schwefel, H. –P. (1993), “Evolutionary programming and evolution strategies: similarities and differences”, Proceedings of the 2nd Annual Conference on Evolutionary Programming, Fogel, D. B. and Atmar, W. (Editors), pp. 11-22, Evolutionary Programming Society.
- Bailey, T. B., and Hubbard, J. E. (1985), “Distributed piezoelectric-polymer active vibration control of a cantilever beam”, Journal of Guidance Control and Dynamics, Vol. 8, No. 5, pp. 605-611.
- Bar-Cohen, Y. (1986), “NDE of fiber reinforced composites—A review”, Material Evaluation, Vol. 44, No. 4, pp. 446-454.
- Bhalla, S., Soh, C. K., Tseng, K. K. –H., and Naidu, A. S. K. (2001), “Diagnosis of incipient damage in steel structures by means of piezoceramic patches”, Proceedings of the 8th East Asia-Pacific Conference on Structural Engineering and Construction on Challenges in the 21st Century, 5-7 December, Singapore, in CD, paper No. 1598.
- Birru, H. K., Chellapilla, K., and Rao, S. S. (1999), “Local search operators in fast evolutionary programming”, Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), Vol. 2, pp. 1506-1513.
- Boscheti, F., Dentith, D. C., and List, R. D. (1996), “Inversion of seismic refraction data using genetic algorithms”, Geophysics, Vol. 61, No. 6, pp. 1715-1727.
- Bray, D., and McBride, D. (1992), Nondestructive testing techniques, Wiley, New York.
- Burczynski, T., Beluch, W., Dlugosz, A., Orantek, P., and Nowakowski, M. (2000), “Evolutionary methods in inverse problems of engineering mechanics”, Inverse Problems in Engineering Mechanics, Tanaka, M., and Dulikravich, G. S. (Editor), Elsevier Science Ltd., pp.553-562.
- Cao, H., Kang, L., and Chen, Y. (1999), “Evolutonary modelling of ordinary differential equations for dynamic system”, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), pp. 959-965.
- Cao, H., Kang, L., Michalewicz, Z, and Chen, Y. (1998), “A two-level evolutionary algorithm for modelling system of ordinary differential equations”, Proceedings of the 3rd Annual Conference on Genetic Programming, pp. 17-22.

References

- Cao, Y. J., Liang, L., and Wu, Q. H. (2000), “An evolutionary programming approach to mixed-variable optimisation problems”, *Applied Mathematical Modelling*, Vol. 24, pp. 931-942.
- Chaudhry, Z., Joseph, T., Sun, F. and Rogers, C. (1995), “Local-area health monitoring of aircraft via piezoelectric actuator/sensor patches”, *Proceedings of SPIE North American Conference on Smart Structures and Materials*, San Diego, CA, 26 Feb–3 March, Vol. 2443, pp. 268–276.
- Chaudhry, Z., Sun, F. P., and Rogers, C. A. (1994), “Health monitoring of space structures using impedance measurements”, *Proceedings of the 5th International Conference on Adaptive Structures*, Sendai, Japan, 5–7 December, pp. 584 –591.
- Chellapilla, K. (1998), “Combining mutation operators in evolutionary programming”, *IEEE Transactions on Evolutionary Computation*, Vol. 2, Issue 3, pp. 91-96.
- Chellapilla, K., and Fogel, D. B. (1997), “Two new mutation operators for enhanced search and optimization in evolutionary programming,” *Proceedings of SPIE’s International Symposium on Optical Science, Engineering, and Instrumentation*, Conference 3165: Applications of Soft Computing, Bosacchi, B., Bezdek, J. C., and Fogel, D. B. (Editors), Billingham, WA: SPIE, pp. 260-269.
- Chellapilla, K., and Fogel, D. B. (1999), “Evolution, neural networks, games, and intelligence”, *Proceedings of the IEEE*, Vol. 87, No. 9, pp. 1471-1496.
- Chellapilla, K., Birru, H. K., and Rao, S. S. (1998), “Effectiveness of local search operators in EP”, *Genetic Programming: Proceedings of the 3rd Annual Conference*, pp. 753-761.
- Choi, D. -H. (2002), “Cooperative mutation based evolutionary programming for continuous function optimization”, *Operations Research Letters*, Vol. 30, pp. 195-201.
- Choi, D. -H., and Oh, S. -Y. (2000), “A new mutation rule for evolutionary programming motivated from backpropagation learning,” *IEEE Transactions on Evolutionary Computation*, Vol.4, Issue 2, pp. 188-190.

References

- Chow, S. T., Liew, K. M., and Lam, M. Y. (1992), "Transverse vibration of symmetrically laminated rectangular composite plates", Composite Structures, Vol. 20, No. 4, pp. 213-226.
- Cook, D. F., Ragsdale, C. T., and Major, R. L. (2000), "Combining a neural network with a genetic algorithm for process parameter optimization", Engineering Applications of Artificial Intelligence, Vol. 13, pp. 391-396.
- Crawley, E. F., and de Luis, J. (1987), "Use of piezoelectric actuators as elements of intelligent structures", AIAA Journal, Vol. 25, No. 10, pp. 1373-1385.
- Cudney, H. H., Inman, D. J., and Oshman, Y. (1990), "Distributed structural control using multilayered piezoelectric actuators", Proceedings of the AIAA/ASME/ASCE/AHS/ASC 31st Structures. Structural Dynamics and Materials Conference, Long Beach, CA, April 2-4, 1990, AIAA-90-1088-CP, pp. 2257-2264.
- Davidon, W. C. (1959), "Variable metric method for minimization", Atomic Energy Commission Research and Development Report, ANL-5990.
- Dimitriadis, E. K., Fuller, C. R., and Rogers, C. A. (1989), "Piezoelectric actuators for distributed noise and vibration excitation of thin plates", ASME Failure Prevention and Reliability, DE-Vol. 16, pp. 223-233.
- Doebling, S. W., Fmar, C. R., Prime, M. B., and Shevitz, D. W. (1996), "Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review", Report LA-13070-MS, Los Alamos, NM.
- El-Sharkh, M. Y., and El-Keib, A. A. (2003), "An evolutionary programming-based solution methodology for power generation and transmission maintenance scheduling", Electric Power Systems Research, Vol. 65, pp.35-40.
- Esteban, J. (1996), "Modeling of the sensing region of a piezoelectric actuator/hensor", Ph. D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Va.
- Fainveather, J. A. (1998), "Designing with active materials: An impedance based approach", Ph. D. Thesis, Rensselaer Polytechnic Institute.
- Fanson, J. L., and Caughey, T. K. (1987), "Positive position feedback control for large space structures", Proceedings of AIAA/ASME/ASCE/AHS/ASC 28th

References

- Structures, Structural Dynamics and Materials Conference, Monterey, CA. AIAA publishing, Washington D.C. pp 588-598.
- Farrar, C. R., Doebling, S. W., and Nix, D. A. (2001), "Vibration-based structural damage identification", Philosophical Transactions of the Royal Society of London Series A, Mathematical Physical and Engineering Sciences, Vol. 359, No. 1778, pp.131-49.
- Favro, L. D., and Thomas, R. L. (1995), "Thermal wave hardware evolution using the NDI validation center", Materials Evaluation, Vol. 53, pp. 840-843.
- Fletcher, R., and Powell, M. J. D. (1963), "A rapidly convergent descent method for minimization," Computer Journal, Vol. 6, pp. 163-168.
- Fogel, D. B. (1988), "An evolutionary approach to the traveling salesman problem", Biological Cybernetics, Vol, 60, pp.139-44.
- Fogel, D. B. (1989). "Evolutionary programming for voice feature analysis", Proceedings of the 23rd Annual Conference on Signals, Systems and Compactors, Pacific Grove, CA, pp. 381-383.
- Fogel, D. B. (1990), "A parallel processing approach to a multiple traveling salesman problem using evolutionary programming", Proceedings on the 4th Annual Paralled Processing Symposium, Canler, L. (Editor), Fullerton, CA, pp.318-326.
- Fogel, D. B. (1991), System Identification through Simulated Evolution: A Machine Learning Approach to Modeling, Needham Heights, MA: Ginn.
- Fogel, D. B. (1992), "Evolving artificial intelligence", Ph. D. Thesis, University of California, San Diego, CA.
- Fogel, D. B. (1993), "Applying evolutionary programming to selected traveling salesman problems". Cybernetics and Systems, Vol. 24, pp.27-36.
- Fogel, D. B. (1994), "An introduction to simulated evolutionary optimization", IEEE Transactions on Neural Networks, Vol. 5, Issue 1, pp. 3-14.
- Fogel, D. B. (1999), "An introduction to evolutionary computation and some applications", Evolutionary Algorithms in Engineering and Computer Science, Miettinen, K., Neittaanmaki, P., Makela, M. M., and Periaux, J. (Editors), John Wiley & Sons, Ltd, Singapore, pp.23-41.

References

- Fogel, D. B. (2000a), Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 2nd Ed., IEEE Press, New York.
- Fogel, D. B. (2000b). "Applying Fogel and Burgin's 'competitive goal-seeking through evolutionary programming' to coordination, trust, and bargaining games", Proceedings of the 2000 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, pp.1210-1216.
- Fogel, D. B., and Atmar, J. W. (1990), "Comparing genetic operators with gaussian mutations in simulated evolutionary processing using linear systems", Biological Cybernet, Vol. 63, pp.111-114.
- Fogel, D. B., and Fogel, L. J. (1988). "Route optimization through evolutionary Programming", Proceedings of 25th Annual Conference on Signals, Systems and Compactors, Pacific Grove, CA, pp.679-680.
- Fogel, D. B., Fogel, L. J., and Atmar, J. W. (1991), "Meta-evolutionary programming", Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers, pp. 540-545.
- Fogel, L. J. (1962), "Autonomous automata", Industrial Research, Vol. 4, pp. 14-19.
- Fogel, L. J., (1968), "Extending communication and control through simulated evolution, bioengineering -- an engineering view", Proceedings of the Symposium on Engineering Significance of the Biological Sciences, Bugliarello, G. (Editor) San Francisco, CA: San Francisco Press, pp.286-304.
- Fogel, L. J., and Burgin, G. H. (1969), "Competitive goal-seeking through evolutionary programming", Final report under Contract No. AF 19(628)-5927, Air Force Cambridge Research Laboratories.
- Fogel, L. J., and Fogel, D. B. (1986). "Artificial intelligence through evolutionary programming". US Army Research Institute Final Report, Contract No. p09-x56-1102c-01.
- Fogel, L. J., Angeline, P. J., and Back, T. (Editors) (1996), Proceedings of the 5th Annual Conference On Evolutionary Programming, MIT Press, Cambridge, Mass.

References

- Fogel, L. J., Owens, A. J., and Walsh, M. J. (1964), "On the evolution of artificial intelligence", Proceedings of the 5th National Symposium on Human Factors in Electronics, San Diego, CA: IEEE, pp. 63-76.
- Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966), Artificial Intelligence through Simulated Evolution, Wiley, New York.
- Frederiksen, P. S. (1992), "Identification of material parameters in anisotropic plates—a combined numerical-experimental method", Ph. D. Thesis, Department of Solid Mechanics, The Technical University of Denmark.
- Frederiksen, P. S. (1997), "Experimental procedure and results for the identification of elastic constants of thick orthotropic plates", Journal of Composite Materials, Vol. 31, No. 4, pp. 360-382.
- Gen, M. and Cheng, R. (1997), Genetic Algorithms and Engineering Design, John Wiley & Sons, Inc., New York.
- Gen, M., Liu, B. D., and Ida, K. (1996), "Evolution program for deterministic and stochastic optimizations", European Journal of Operational Research, Vol. 94, pp. 618-625.
- Ghoshal, A., Harrison, J., Sundaresan, M. J., Hughes, D., and Schulz, M. J. (2001), "Damage detection testing on a helicopter flexbeam." Journal of Intelligent Material Systems and Structures, Vol. 12, No. 5, pp. 315-329.
- Giurgiutiu, V. (2000), "Active sensors for health monitoring of ageing aerospace structures", Proceedings of SPIE Conference on Smart Structures and Integrated Systems, Vol. 3985, pp. 294-305.
- Giurgiutiu, V., and Rogers, C. A. (1997), "Electro-Mechanical (EM) impedance method for structural health monitoring and non-destructive evaluation", International Workshop on Structural Health Monitoring, Stanford University, CA., Sep. 18-20, pp. 433-444.
- Giurgiutiu, V., and Rogers, C. A. (1999), "Modeling of the electro-mechanical (E/M) impedance response of a damaged composite beam", Adaptive Structures and Materials Systems, ASME, AD-Vol. 59/MD-Vol.87, pp 39-46.

References

- Giurgiutiu, V., and Zagrai, A. N. (2001), "Characterization of piezoelectric wafer active sensors", Journal of Intelligent Material Systems and Structures, Vol. 11, pp. 959-976.
- Giurgiutiu, V., Zagrai, A. N. and Bao, J. (2001), "Piezoelectric wafer embedded active sensors for aging aircraft structural health monitoring", Structural Health Monitoring, Vol. 1, No. 1, pp. 41-61.
- Goch, G., Schmitz, B., Karpuschewski, B., Geerkens, J., Reigl, M., Sprongl, P., and Ritter, R. (1999), "Review of non-destructive measuring methods for the assessment of surface integrity: a survey of new measuring methods for coatings, layered structures and processed surfaces", Precision Engineering, Vol. 23, Issue 1, pp. 9-33.
- Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, New York.
- Gormme, P., and Harrald, P. G. (1998), "Applying evolutionary programming to selected set partitioning problems", Fuzzy Sets and Systems, Vol. 95, Issue 1, pp. 67-76.
- Green, R. E. (1997), "Emerging technologies for NDE of aging aircraft structures", Proceedings of the Workshop on Intelligent NDE Sciences for Aging and Futuristic Aircraft, C. Ferregut, R. Osegueda and A. Nunez (Editors.), UTEP, El Paso, TX, ISBN 87404-279-8, pp.267-278.
- Hajela, P., and Soeiro, F. J. (1989), "Structural damage detection based on static and modal analysis", AIAA Journal, Vol. 28, No. 6, pp. 1110-1115.
- Han, X., Favro, L. D., Kuo, P. K., and Thomas, R. L. (1996), "Early-time pulse-echo thermal wave imaging", Review of Progress in Quantitative NDE, Vol. 15, Thompson, D. O., and Chimenti, D. (Editors), Plenum Press, NY, pp. 519-524.
- Hart, W. (1994), "Adaptive global optimization with local search", Ph.D thesis, Dept. of Computer Science and Engineering, University of California at San Diego, CA.
- Hart, W., and Belew, K. R. (1996), "Optimization with genetic algorithms hybrids that use local search", Adaptive Individuals in Evolving Populations, Belew, R. K., and Mitchell, M. (Editors), Addison Wesley.

References

-
-
- Hoffmeister, F. and Back, T. (1991), “Genetic algorithms and evolution strategies: similarities and differences”, Parallel Problem Solving from Nature, PPSN I. Lectures Notes in Computer Science, Schwefel, H. –P. and Manner, R. (Editors), Springer, Vol. 496, pp. 455-470.
- Holland, J. H. (1975), Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI.
- Howard, L. M., and D’Angelo, J. (1995), “The GA-P: a genetic algorithm and genetic programmign hybrid”, IEEE, Expert, Issue 3, pp. 11-15.
- Huang, Y., Gedeon, T. D., and Wong, P. M. (2001), “An integrated neural-fuzzy-genetic-algorithm using hyper-surface membership functions to predict permeability in petroleum reservoirs”, Engineering Applications of Artificial Intelligence, Vol. 14, pp. 15-21.
- IEEE Std. 176, (1987). *IEEE Standard on Piezoelectricity*, The Institute of Electrical and Electronics Engineering, Inc.
- Ip, K. H., Tse, P. C., and lai, T. C. (1998), “Material Characterization for orthotropic shells using modal analysis and Rayleigh-Ritz models”, Composites, Part B, Vol. 29, No. 4, pp. 397-409.
- Iwamatsu, M. (2001), “Applying evolutionary programming to structural optimization of atomic clusters”, Computer Phvsics Communications, Vol. 142, Issues 1-3, pp. 214-218.
- Iwamatsu, M. (2002), “Generalized evolutionary programming with Levy-type mutation”, Computer Physics Communications, Vol. 147, pp. 729-732.
- Jones, R. M. (1975), Mechanics of Composite Materials, Hemisphere Publishing Co., New York.
- Juneja, V., Haftka, R. T., and Cudney, H. H. (1994), “Location of damage in a space structure using contrast maximization”, Proceedings of the 35th AIAA/ASME/AHS/ASC Structures, Structural Dynamics and Materials Conference, Washinton, D. C., April, pp. 296-306.
- Juneja, V., Haftka, R. T., and Cudney, H. H. (1997), “Damage detection and damage detectability—analysis and experiments”, Journal of Aerospace Engineering, ASCE, Vol. 10, No. 4, pp. 135-142.

References

- Kam, T. Y., and Liu, C. K. (1998), "Stiffness identification of laminated composite shafts", International Journal of Mechanical Science, Vol. 40, No. 9, pp. 927-936.
- Karr, C. L., Yakushin, I., and Nicolosi, K. (2000), "Solving inverse initial-value, boundary-value problems via genetic algorithm", Engineering Applications of Artificial Intelligence, Vol. 13, pp. 625-633.
- Kim, H. M., and Bartkowicz, T. J. (1994), "A two-step structural damage detection approach with limited instrumentation", Proceedings of the 35th AIAA/ASME/AHS/ASC Structures, Structural Dynamics and Materials Conference, Washinton, D. C., April, pp. 318-324.
- Kim, H. -S., and Cho, S. -B. (2000), "Application of interactive genetic algorithm to fashion design", Engineering Applications of Artificial Intelligence, Vol. 13, pp. 635-644.
- Kim, H., Querin, O. M., and Steven, G. P. (2002), "On the development of structural optimization and its relevance in engineering design", Design Studies Vol. 23, Issue 1, pp. 85-102.
- Kim, J. H., Chae, H. K., Jeon, J. Y., and Lee, S. W. (1996), "Identification and control of systems with friction using accelerated evolutionary programming", IEEE Control Systems Magazine, Vol. 16, pp. 38-47.
- Kim, S. J., and Jones, J. D. (1991), "Optimal design of piezoactuators for active noise and vibration control, AIAA Journal, Vol. 29, No. 12, pp. 2047-2053.
- Koza, J. R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA.
- Lalande, F. (1995), "Modelling of the induced strain actuation of shell structures", Ph.D. Thesis, Virginia Polytechnic Institute and State University.
- Li, B., Lin, J., and Yao, X. (2002), "A novel evolutionary algorithm for determining unified creep damage constitutive equations", International Journal of Mechanical Sciences, Vol. 44, Issue 5, pp. 987-1002.
- Li, J., and Kwan, R. S. K. (2003), "A fuzzy genetic algorithm for driver scheduling", European Journal of Operational Research, Vol. 147, Issue 2, pp. 334-344.

References

- Li, M., and Wilkinson, D. (2001), "Particle size distribution determination from spectral extinction using evolutionary programming", Chemical Engineering Science, Vol. 56, Issue 10, pp. 3045-3052.
- Liang, C., Sun, F. P., and Rogers, C. A. (1993), "An impedance method for dynamic analysis of active material systems." Proceedings at ATAA/ASME/ASCE Material Systems, La Jolla, California, pp.3587-3599.
- Liang, C., Sun, F. P., and Rogers, C. A. (1996), "Electro-mechanical impedance modeling of active material systems", Smart Materials and Structures, Vol. 5, pp. 171-186.
- Liang, K. -H., Yao, X., Newton, C., and Hoffman, D. (1998a), "An experimental investigation of self-adaptation in evolutionary programming", Proceedings of the 7th International Conference on Evolutionary Programming, Springer, Berlin, pp. 291-300.
- Liang, K. -H., Yao, X., and Newton, C. (1998b), "Dynamic control of adaptive parameters in evolutionary programming", Proceedings of Simulated Evolution and Learning (SEAL98): Second Asia Pacific Conference, Springer, Berlin, pp. 42-49.
- Liang, R. Y., Hu, J., and Choy, F. (1992), "Quantitative NDE technique for assessing damages in beam structures", Journal of Engineering Mechanics, ASCE, Vol. 118, No. 7, pp. 1468-1487.
- Liew, K. M. (1990), "The development of 2-D orthogonal polynomials for vibration of plates", Ph. D. Thesis, National University of Singapore, Singapore.
- Liew, K. M., Karunasena, W., Kitipornchai, S., and Chen, C. C. (1999), "Vibration of unsymmetrically laminated thick quadrilateral plates", Journal of Acoustical Society of America, Vol. 105, No. 3, pp. 1672-1681.
- Liew, K. M., Wang, C. M., Xiang, Y., and Kitipornchai, S. (1998). Vibration of Mindlin Plates, Elsevier Science Ltd.
- Lozano, J. A. (2002), "An introduction to evolutionary algorithms", Estimation of Distribution Algorithms, Larranaga, P. and Lozano, J. A. (Editors.), Kluwer Academic Publisher, Boston/Dordrecht/London, pp. 3-25.
- Lubin, G. (1982), Handbook of Composites, Van Nostrand Reinhold, New York.

References

-
-
- McCann, D. M., and Forde, M. C. (2001), “Review of NDT methods in the assessment of concrete and masonry structures”, NDT&E International, Vol. 34, Issue 2, pp. 71-84.
- Mcdonnell, J. R., Reynolds, R. G., and Fogel, D. B. (Editors) (1995), Proceedings of the 4th Annual Conference On Evolutionary Programming, MIT Press, Cambridge, Mass.
- Michalewicz, Z. (1996), Genetic Algorithm + Data Structures = Evolution Programs, 3rd edition, Springer, Berlin.
- Michalewicz, Z., Hinterding, R. and Michalewicz, M. (1997), “Evolutionary algorithms”, Fuzzy Evolutionary Computation, Pedrycz, W. (Editor), Kluwer Academic Publishers, Bonston/London/Dordrecht, pp. 3-31.
- Mindlin, R. D. (1951), “Influence of rotatory inertia and shear on flexural motions of isotropic elastic plates”, Journal of Applied Mechanics, Vol. 18, pp. 31-38.
- Minster, J. -B. H., Williams, N. P., Masters, T. G., Gilbert, J. F., and Haase, J. S. (1995), “Application of evolutionary programming to earthquake hypocenter determination,” Proceedings of the 4th Annual Conferenc on Evolutionary Promamming, Mcdonnel, J. R., et al (Editors), The MIT Press. pp. 3-17.
- Mohan, D., and Kingsbury, H. D. (1971), “Free vibration of general orthotropic plates”, Journal of Acoustical Society of America, Vol. 50, pp. 266-269.
- Ochoa, O. O., and Reddy, J. N. (1992), Finite Element Analysis of Composite Laminates, Kluwer Academic Publishers, London.
- Pandey, A. K., Biswas, M., and Samman, M. M. (1991), “Damage detection from changes in mode shapes”, Journal of Sound and Vibration, Vol. 145, No. 2, pp. 321-332.
- Park, G., Cudney, H. H., and Inman, D. J. (2000), “Impedance-based health monitoring of civil structural components”, Journal of Infrastructure Systems, ASCE, Vol. 6, No. 4, pp. 153-160.
- Park, G., Cudney, H. H., and Inman, D. J. (2001), “Feasibility of using impedance-based damage assessment for pipeline systems”, Earthquake Engineering and Structural Dynamics Journal, Vol. 30, No. 10, pp. 1463–1474.

References

- Park, G., Kabeya, K., Cudney, H., and Inman, D. J. (1999), “Impedance-based structural health monitoring for temperature varying applications”, JSME International Journal Series A, Vol. 42, No. 2, pp. 249–258.
- Patton, T. C., and Hsu, D. K. (1998), “Recent developments of the driplless bubbler ultrasonic scanner,” Review of Progress in Quatitative NDE, Vol. 15B, D. O. Thompson and D.E. Chimenti (Eds.), Plenum Press, New York, pp. 2045-2051.
- Pawar, P. M., and Ganguli, R. (2003), “Genetic fuzzy system for damage detection in beams and helicopter rotor blades”, Computer Methods in Applied Mechanics and Engineering, Vol. 192, Issues 16-18, pp. 2031-2057.
- Pedersen, P. (1989), “Optimization method applied to identification of materail parameters”, Discretization Methods and Structural Optimization—Procedures and applications, Eschenauer, H. A. and Ehierauf, G. (Editors), Springer-Verlag, Berlin, pp. 277-283.
- Porto, V. W. (1997), “Evolutionary programming”, Handbook of Evolutionary Computation, Back, T., Fogel, D. B., and Michalewics, Z. (Editors), Oxford University Press, New York, Chapter B1.4.
- Porto, V. W., Saravanan, N., Waagen, D. E., and Eiben, A. E. (Editors) (1998), Proceedings of the 7th Annual Conference on Evolutionary Programming, San Diego, CA, USA.
- Rechenberg, I. (1973), Evolutionstrategies: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Fromman-Holzboog, Stuttgart.
- Reissner, E. (1945), “The effect of transverse shear deformation on the bending of elastic plates”, Journal of Applied Mechanics, Vol. 12, pp. A69-A77.
- Rhim, J., and Lee, S. W. (1995), “A neural network approach for damage detection and identification of structures”, Computational Mechanics, Vol. 16, pp. 437-443.
- Rickles, J. M., and Kosmatka, J. H. (1992), “Damage detection in elastic vibratory structures using residual forces and weighted sensitivity”, AIAA Journal, Vol. 30, No. 9, pp. 2310-2316.

References

- Rikards, R., Chate, A., Steinchen, W., Kessler, A., and Bledzki, A. K. (1999), "Method for identification of elastic properties of laminates based on experiment design", Composites, Part B, Vol. 30, pp. 279-289.
- Roddis, W. M. K., and Bocox, J. (1997), "Case-based approach for steel bridge fabrication errors", Journal of Computing in Civil Engineering, ASCE, Vol. 11, No. 2, pp. 84-91.
- Roddis, W. M. K., Hess, M. R., Melhem, H. G., and Nagaraja, S. (1995), "BFX: an operational expert system for bridge fabrication", Transportation Research Record, Vol. 1491, pp. 62-68.
- Rossi, A., Liang, C., and Rogers, C. A. (1993). "Coupled electric-mechanical analysis of a piezoceramic actuator driven system—an application to a circular ring." Proceedings of the AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics and Materials Conference, LaJolla, CA, April 19-22, pp. 3618-3624.
- Rudolph, G. (1997), "Local convergence rates of simple evolutionary algorithms with Cauchy mutations", IEEE Transactions on Evolutionary Computation, Vol. 1, No. 4, pp. 249-258.
- Rytter, A. (1993), "Vibration based inspection of civil engineering structures", Ph. D. Thesis, Department of Building Technology and Structural Engineering, Aalborg University, Denmark.
- Sanayei, M., and Onipede, O. (1991), "Assessment of structures using static test data", AIAA Journal, Vol. 29, No. 7, pp. 1156-1179.
- Sanayei, M., and Scampoli, S. F. (1991), "Structural element stiffness identification from static test data", Journal of Engineering Mechanics, Vol. 117, No. 5, pp. 1021-1036.
- Schuetze, M., Doll, H., and Hildebrandt, P. (1994), "Review on an expert system under development: measuring techniques in civil engineering including structural assessment", Proceeding of the 13th International Modal Analysis Conference, pp. 1349.
- Schwartz, M. M. (1983), Composite Materials Handbook, Maraw-Hill, New York.

References

- Schwefel, H. -P. (1981), Numerical Optimization of Computer Models, John Willey & Sons, Inc.
- Sensmeier, M. D., Sensharma, P. K., Haftka, R. T., Griffin, O. H., and Watson, L. T. (1995), "Experimental validation of anti-optimization approach for detecting delamination damage", Proceedings of the 36th AIAA/ASME/AHS/ ASC Structures, Structural Dynamics and Materials Conference, Washinton, D. C., April, pp. 3171-3184.
- Shen, M. -H. H., and Taylor, J. E. (1991), "An identification problem for vibrating cracked beams", Journal of Sound and Vibration, Vol. 150, No. 3, pp. 457-484.
- Shi, L., and Xu, G. (2001), "Self-adaptive evolutionary programming and its application to multi-objective optimal operation of power systems", Electric Power Systems Research, Vol. 57, pp. 181-187.
- Sivakumaran, K. S. (1987), "Natural frequencies of symmetrically laminated rectangular plates with free edges", Composite Structures, Vol. 7, pp. 191-204.
- Soares, C. M. M., Moreira de Freitas, M., and Araujo, A. L. (1993), "Identification of material properties of composite plate specimens", Composite Structures, Vol. 25, pp. 277-285.
- Soh, C. K., and Dong, Y. X. (2001), "Evolutionary programming for inverse problems in civil engineering", Journal of Computing in Civil Engineering, ASCE, Vol. 15, No. 2, pp. 144-150.
- Soh, C. K., and Yang, Y. W. (2000), "Genetic programming-based approach for structural optimization", Journal of Computing in Civil Engineering, ASCE, Vol. 14, No. 1, pp. 31-37.
- Soh, C. K., Tseng, K. K. -H., Bhalla, S. and Gupta, A. (2000), "Performance of smart piezoceramic patches in health monitoring of a RC bridge", Smart Materials and Structures, Vol. 9, No. 4, pp. 533-542.
- Sol, H. (1986), "Identification of anisotropic plate rigidities using free vibration data", Ph. D. Thesis, Free University of Brussels.
- Solis, F. J., and Wets, R. J.-B. (1981), "Minimization by random search techniques," Mathematical Operations Research, Vol. 6, pp. 19-30.

References

- Stanley, C. C., and Balendran, R. V. (1995), “Developments in assessing the structural integrity of applied surfaces to concrete buildings and structures using infra-red thermography”, Proceedings of the International Conference on Structural Faults & Repair-95, Engineering Technics Press, Vol. 3, pp. 39-44.
- Steven, G. P., Li, Q., and Xie, Y. M. (2000), “Evolutionary topology and shape design for general physical field problems”, Computational Mechanics, Vol. 26, No. 2, pp.129-139.
- Sun, F., Chaudhry, Z., Liang, C., and Rogers, C. A. (1995), “Truss structure integrity identification using PZT sensor-actuator”, Journal of Intelligent Material Systems and Structures, Vol. 6, No. 2, pp. 134–139.
- Swain, A. K., and Morris, A. S. (2002), “Performance improvement of self-adaptive evolutionary methods with dynamic lower bound”, Information Processing Letters, Vol. 82, pp. 55-63.
- Swain, A. K., and Morris, A. S. (2003), “**An** evolutionary approach to the automatic generation of mathematical models”, Applied Soft Computing, Vol. 51, pp. 1-21.
- Tsay, M. –T., Lin, W. –M., and Lee, J. –L. (2001), “Application of evolutionary programming for economic dispatch of cogeneration systems under emission constraints”, International Journal of Electrical Power & Energy Systems, Vol. 23, Issue 8, pp. 805-812.
- van Kemenade, C. H. M. (1995), “A two-level evolution strategy (balancing global and local search)”, CWI report CS-R9559, ISSN 0169-118X.
- Vestroni, F., and Capecchi, D. (1996), “Damage evaluation in cracked vibrating beams using experimental frequencies and finite element models”, Journal of Vibration and Control, Vol. 2, No. 1, pp. 69-86.
- Vestroni, F., and Capecchi, D. (2000), “Damage detection in beam structures based on frequency measurements”, Journal of Engineering Mechanics, Vol. 126, No. 7, pp. 761-768.
- Wang, B. T., and Rogers, C. A. (1991), “Modelling of finite-length spatially distributed induced strain actuators for laminate beams and plates”, Journal of Intelligent Material Systems and Structures, Vol. 2, No. 1, pp. 38-58.

References

- Wang, W. T., and Kam, T. Y. (2001), "Elastic constants identification of shear deformable laminated composite plates", Journal of Engineering Mechanics, Vol. 127, No. 11, pp. 1117-1123.
- Wang, X., Hu, N., Fukunaga, H., and Yao, Z. H. (2001), "Structural damage identification using static test data and changes in frequencies", Journal of Engineering Structures, Vol. 23, Issue 6, pp. 610-621.
- Weber, L. (1998), "Evolutionary combinatorial chemistry: application of genetic algorithms", Drug Discovery Today, Vol. 3, Issue 8, pp. 379-385.
- Wei, C., Yao, S., and He, Z. (1996), "A modified evolutionary programming", Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 135-138.
- Wilde, W. P., and Sol, H. (1987), "Anisotropic material identification using measured resonant frequencies of rectangular composite plates", Composite Structures, Vol. 4, No. 2, pp. 2317-2324.
- Wittrick, W. H. (1987), "Analytical, three-dimensional elasticity solutions to some plate problems, and some observations on Mindlin's plate theory", International Journal of Solids and Structures, Vol. 23, pp. 441-464.
- Wong, K. P., and Algie, C. (2002), "Evolutionary programming approach for combined heat and power dispatch", Electric Power Systems Research, Vol. 61, pp. 227-232.
- Xie, Y. M., and Steven, G. P. (1993), "A simple evolutionary procedure for structural optimisation", Computers and Structures, Vol. 49, pp. 885-896.
- Xie, Y. M., and Steven, G. P. (1997), Evolutionary structural optimisation, Springer-Verlag, Berlin.
- Yang, Y. W., and Soh, C. K. (2000), "Fuzzy logic integrated genetic programming for optimization and design", Journal of Computing in Civil Engineering, Vol. 14, No. 4, pp. 249-254.
- Yao, X., and Liu, Y. (1996), "Fast evolutionary programming", Evolutionary Programming V: Proceedings of the 5th Annual Conference on Evolutionary Programming, Fogel, L. J., Angeline, P. J., and Back, T. (Editors), Cambridge, MA: MIT Press, pp. 451-460.

References

- Yao, X., and Liu, Y. (1998), “Scaling up evolutionary programming algorithms”, Evolutionary Programming VII: Proceedings of the 7th Annual Conference on Evolutionary Programming (EP98), Lecture Notes in Computer Science, Springer-Verlag, Berlin.
- Yao, X., Liu, Y., and Lin, G. (1999), “Evolutionary programming made faster”, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2, pp. 82-102.
- Yoseph, B.-C. (1999) “In-service NDE of aerospace structures--Emerging technologies and challenges at the end of the 2nd Millennium”, NDT.net, Vol. 4, No. 9, <http://www.ndt.net/article/v04n09/bcohen/bcohen.htm>.
- Zhang, J. -H., and Xu, X. -H. (1998), “An efficient evolutionary programming algorithm”, Computers & Operations Research, Vol. 26, pp. 645-663.
- Zhou, S., Liang, C., and Rogers, C. A. (1994). “A dynamic model of piezoelectric actuator-driven thin plates.” Proceeding of the Smart Structures and Materials Conference, Orlando, SPIE, Vol. 2190, pp 550-562.
- Zhou, S., Liang, C., and Rogers, C. A. (1996). “An impedance-based system modeling approach for induced strain actuator-driven structures.” Journal of Vibration and Acoustics, Vol. 118, pp. 323-331.
- Zimmerman, D. C., Smith, S. W., Kim, H. M., and Bartkowicz, T. J. (1994), “An experimental study of structural damage detection using incomplete measurements”, Proceedings of the 35th AIAA/ASME/AHS/ASC Structures, Structural Dynamics and Materials Conference, Washinton, D. C., April, pp. 307-317.
- Zou, Y., Tong, L., and Steven, G. P. (2000), “Vibration-based model-dependent damage (delamination) identification and health monitoring for composite structures—A review”, Journal of Sound and Vibration, Vol. 230, No. 2, pp. 357-378.

APPENDIX

A list of C++ code for the computer program of the integrated evolutionary programming and the impedance-based method (inEPIB) is presented in this section. The inEPIB code is segmented into 22 files.

```
//=====BuildinHeader.h=====
#ifndef __BUILDINHEADER_H
#define __BUILDINHEADER_H

#include <math.h>
#include <time.h>
#include <iomanip.h>
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>
#include <e:\mathd.h>
#endif

//=====Random.h =====
#ifndef __RANDOM_H
#define __RANDOM_H

#include "BuildinHeader.h"

class CRandom
{
public:
    CRandom();
    void  warmup_random(double);//get random off and mning.
    void  advance_random();//create next batch of 55 random numbers.
    double pickarandom();
    int  random_int(int,int);
    double random_dbl(double,double);
    double UnitGaussian();
private:
    int  irand;
    int  mdcalfag;
    double oldrandom[55];
    double ranx2;
    double randomseed;
};
#endif

//=====Chromosome.h =====
#ifndef __CHROMOSOME_H
#define __CHROMOSOME_H

class CChromosome
{
public:
    CChromosome();
    CChromosome(int);
    CChromosome(CChromosome*);
    ~CChromosome();
    void  putNVariable(int);
    int  getNVariable();
    double putvariable(int,double);
    double getvariable(int);
    double getdeviation(int);
    double putdeviation(int,double);
private:
    int  NVariable;
    double *AVariable;//=X[1],X[2],...X[NVariable];
    double *ADeviation;//deviation;
};
#endif // !defined(_CHROMOSOME_H INCLUDED-)
```

```
//=====Complex.h =====
#ifndef __COMPLEX_H
#define __COMPLEX_H

class CComplex
{
private:
    double  x;//real part
    double  y;//imaginary part
public:
    CComplex();
    CComplex(const double,const double);
    virtual ~CComplex();
    double real();//get real
    void  real(const double);//set real
    double imag();//get imaginary
    void  imag(const double);
//Operators for Complex -----//
    void  operator=(const CComplex& C);
    void  operator=(const double a);

    friend CComplex operator+(const CComplex& CC1, const
        CComplex& CC2);
    friend CComplex operator+(const CComplex& CC1, const
        CComplex& CC2);
    friend CComplex operator+(const double a,const CComplex& CC1);
    friend CComplex operator+(const CComplex& CC1,const double a);
    friend CComplex operator-(const CComplex& CC1);
    friend CComplex operator-(const CComplex& CC1,const CComplex&
        CC2);
    friend CComplex operator-(const CComplex& CC1,const double a);
    friend CComplex operator-(const double a,const CComplex& CC1);
    friend CComplex operator*(const CComplex& CC1,const CComplex&
        CC2);
    friend CComplex operator*(const CComplex& CC1,const double a);
    friend CComplex operator*(const double a,const CComplex& CC1);
    friend CComplex operator/(const CComplex& CC1,const CComplex&
        CC2);
    friend CComplex operator/(const CComplex& CC1,const double a);
    friend CComplex operator/(const double a,const CComplex& CC1);
    friend CComplex cexp(const CComplex& CC1);
    friend CComplex sin(const CComplex& CC1);
    friend CComplex cos(const CComplex& CC1);
    friend CComplex tan(const CComplex& CC1);
    friend CComplex csqrt(const CComplex& CC1);
//End of Operatorsfor Complex -----//
};
#endif

//=====Poly.h =====
#ifndef __POLY_H
#define __POLY_H

class CPoly
{
public:
    CPoly();
    CPoly(CPoly&);
    CPoly(int);
    ~CPoly();
```

```

void init_CPoly(int);
void putNterm(int);
void putXpower(int,int);
void putYpower(int,int);
void putCoef(int,double);
int getNterm();
int getXpower(int);
int getYpower(int);
double getCoef(int);
double getValue(double,double);
double intexy(double,double,double,double);
double intex_y(int,double,double,double);
void simplify();
CPoly& operator=(const CPoly&);
private:
    int Nterm;
    int *Xpower,*Ypower;
    double *Coef;
//Operators for Poly-----//
friend CPoly operator+(CPoly& CP1,CPoly& CP2);
friend CPoly operator-(CPoly& CP1,CPoly& CP2);
friend CPoly operator*(CPoly& CP1,CPoly& CP2);
friend CPoly operator*(double a,CPoly& CP1);
friend CPoly operator*(CPoly& CP1,double a);
friend CPoly dx(CPoly& CP1);
friend CPoly dy(CPoly& CP1);
//End of Operators for Poly-----//

//=====Polyx.h =====
#ifndef __POLYX_H
#define __POLYX_H

class CPolyx
{
public:
    CPolyx();
    CPolyx(CPolyx&);
    CPolyx(int);
    virtual ~CPolyx();
    void putXpower(int,int);
    void putCoef(int,double);
    int getNterm();
    int getXpower(int);
    double getCoef(int);
    virtual void init(int);
    virtual void putNterm(int);
    virtual double getValue(double);
    virtual double inte(double,double);
    virtual void simplify();
    CPolyx& operator=(const CPolyx&);
private:
    int Nterm;
    int Xpower;
    double *Coef;
//Operatorsfor Polyx-----//
friend CPolyx operator+(CPolyx& CP1,CPolyx& CP2);
friend CPolyx operator-(CPolyx& CP1,CPolyx& CP2);
friend CPolyx operator*(CPolyx& CP1,CPolyx& CP2);
friend CPolyx operator*(double a,CPolyx& CP1);
friend CPolyx operator*(CPolyx& CP1,double a);
friend CPolyx dx(CPolyx& CP1);
//End of Operatorsfor Polyx-----//
};
#endif

//=====Parameter.h =====
#ifndef __PARAMETER_H
#define __PARAMETER_H

#include "BuildinHeader.h"

class CParameter
{
public:
    CParameter();
    CParameter(int,int);
    ~CParameter();
    int getMaxMinID();
    int getMax_gen();
    int getPop_size();
    int getNTourna_size();
    int getNVariable();
    double getpara_prob(int);
    double getpara_scal(int,char);
    void setNPara_prob(int);
    void setNVariable(int);
    int getStrTyp();
private:
    int MaxMinID,Max_gen,NPop_size,NPara_prob,
        NVariable,NTourna_size;
    char SMaxMinID[20],SMax_gen[20],SPop_size[20],SNpara_prob[20],
        SNvariable[20],SNTourna_size[20];
    char **SPara_prob,**SPara_scal;
    double *APara_prob,*APara_scal;
    int StrTyp; //1Plate, 2 Beam

    char SStrTyp[10];

    friend istream& operator>>(istream&, CParameter&);
    friend ostream& operator<<(ostream&, const CParameter&);

    endif

//=====Struct.h =====
#ifndef __STRUCT_H
#define __STRUCT_H

#include "BuildinHeader.h"
#include "Chromosome.h"
#include "Complex.h"

class CStruct
{
public:
    int NumEval;
    CStruct();
    virtual ~CStruct();
    int Allocate(int,int,int);
    virtual void IOutput(char*,ofstream&);
    virtual void setBoundary();
    virtual void normalize();
    virtual void stiffmass();//undamaged;
    virtual void stiffmass(CChromosome*);//damaged
    void Eigen();
    virtual void Relang_each_mode();
    virtual void factor();
    virtual double Conduct(double);
    void GetPeak();
    double fitness(CChromosome*);
protected:
    double PI;
    CComplex CC;
    double E, Rou, Niu, H, A, B, Damping; int BondCond;
    char SE[10],SRou[10],SNiu[10],SH[10],SA[10],SB[10],
        SDamping[10], SBondCond[10];
    double E0,Rou0,L0,F0,Omega0,Z0;//the unit of variable properbes,
    double Ep, Yita, RouP, Niup, Hp, Strcoe, Permit, DieLoss;
    char SEp[10],SYita[10],SRoup[10],SNIup[10],SHp[10],SStrcoe[10],
        SPermit[10],SDieLoss[10];
    CComplex CplxY,CplxPermit;//poperties of the PZT patch,
    int Np, mnl, "tem,
    char Snp[10];
    double *K0,*M0;//Undamaged
    double *K,*M;//Damaged
    double *Eval,*Evec;//Eigenvalues and eigenvectors

    double LF,UF;
    char SLF[10],SUF[10];
    int inPeakNum;
    double MUPeak[100],MDPeak[100],PUPeak[100],
        Refer_Peak[100];//RP=PUP+MDP-MUP
    char SMUPeak[20],SMDPeak[20],SPUPeak[20];
    //measured undam, measured dam, and predicted undam
    double CalPeak[100];
    int PeakNum
};
#endif

//===== Beam.h =====
#ifndef __BEAM_H
#define __BEAM_H

#include "Polyx.h"
#include "Struct.h"

class CBeam .public CStruct
private:
    CPolyx faiw;//Bounday condition
    double Uxp,Lxp,Ap;//Location of the PZT patch;
    char SUxp[10],SLxp[10];
    double *Betax;//relative angle on the opposite edges for each mode
    double *Vyita;//Vyita is for storage of factors
    CComplex Z;//force impedance,
    CPolyx *CPK,*CPM; //Poly matrix for calculating K and M matrix.
public:
    CBeam();
    virtual ~CBeam();
    void IOutput(char*,ofstream&);
    void Initialize();
    void setBoundary();
    void normalize();
    void stiffmass();
    void stiffmass(CChromosome*);
    void Relang_each_mode();
    void factor();
    double Conduct(double);
};
#endif

//===== Plate.h =====
#ifndef __PLATE_H
#define __PLATE_H

```



```

#include "Poly.h"
#include "Struct.h"

class CPlate : public CStruct
{
private:
    CPoly faiw; //Boundary condition
    double Uxp, Lxp, Uyp, Lyp, Ap, Bp; //Location of the PZT patch;
    char SUxp[10], SLxp[10], SUyp[10], SLyp[10];
    double *Betax, *Betay; //rela-angle on opposite edges for each mode.
    double *Vytax, *Vyta; //Vyita is for storage of factors.
    CComplex Z[4]; //force impedance;
    CPoly *CPK1, *CPK2, *CPM;
public:
    CPlate();
    virtual ~CPlate();
    void IOutput(char*, ofstream&);
    void Initialize();
    void setBoundary();
    void normalize();
    void stiffmass();
    void stiffmass(CChromosome*);
    void Relang_each_mode();
    void factor();
    double Conduct(double);
};
#endif

//===== Evaluation.h =====
#ifndef EVALUATION_H
#define EVALUATION_H

#include "Parameter.h"
#include "Chromosome.h"
#include "Struct.h"

class CEvaluation
{
public:
    CEvaluation();
    CEvaluation(int);
    CEvaluation(CParameter&);
    ~CEvaluation();
    void cal_fitness(int, CChromosome**, CStruct*);
    void cal_weight(int, CChromosome**, CParameter&);
    void rearrange(int);
    double get_fitness(int);
    void put_fitness(int, double);
    double get_weight(int);
    double get_chromosome(int, int, int);
    void dsort(int, char, CChromosome**, CParameter&);
private:
    int NPop_size;
    double *fitness, *weight;
};
#endif

//===== Random.cpp =====

#include "Random.h"

CRandom::CRandom ()
{
    int j;
    for (j=0; j<=54; j++) oldrandom[j]=0.0;
    jrand=0;
    randomseed=(double)rand();
    randomseed=randomseed/(randomseed+(double)rand());
    warmup=1-random(randomseed);
}

void CRandom::warmup_random(double randomseed)
{
    int j, i;
    double newrandom, prevrandom;
    oldrandom[54]=randomseed;
    newrandom=0.0000000001;
    prevrandom=randomseed;
    for (j=1; j<=54; j++)
    {
        i=(21*j)%54;
        oldrandom[i]=newrandom;
        newrandom=prevrandom-newrandom;
        if (newrandom<0.0) newrandom=newrandom+1.0;
        prevrandom=oldrandom[i];
    }

    advance=1-random();
    advance=1-random();
    advance=1-random();

    jrand=0;
}

void CRandom::advance_random()
{
    double newrandom;

    for (j=0; j<24; j++)
    {
        newrandom=oldrandom[j]-oldrandom[j+31];
        if (newrandom<0.0)
            newrandom=newrandom+1.0;
        oldrandom[j]=newrandom;
    }
    for (j=24; j<55; j++)
    {
        newrandom=oldrandom[j]-oldrandom[j-24];
        if (newrandom<0.0)
            newrandom=newrandom+1.0;
        oldrandom[j]=newrandom;
    }
}

double CRandom::pickarandom()
{
    jrand++;
    if (jrand>=55) {jrand=1; advance=1-random();}
    return (oldrandom[jrand]);
}

int CRandom::random_int(int low, int high)
{
    int i;
    if (low>=high) i=low;
    else
    {
        i=low+(int)( pickarandom()*(high-low+1));
        if (i>high) i=high;
    }
    return i;
}

double CRandom::random_db(double low, double high)
{
    return ( (low+pickarandom()*(high-low)) );
}

// Return a number from a unit Gaussian distribution The means 0
// and the standard deviation is 1.0.
// First we generate two uniformly random variables inside the
// complex unit circle. Then we transform these into Gaussians using
// the Box-Muller transformation. This method is described in
// Numerical Recipes in C
// ISBN0-521-43108-5 at http://world.std.com/~nr
// When we find a number, we also find its twin, so we cache that here
// so that every other call is a lookup rather than a calculation
double CRandom::UnitGaussian()
{
    static bool cached=false;
    static double cachevalue;
    if (cached == true)
    {
        cached=false;
        return cachevalue;
    }

    double rsquare, factor, var1, var2;
    do
    {
        var1 = 2.0 * pickarandom() - 1.0;
        var2 = 2.0 * pickarandom() - 1.0;
        rsquare = var1*var1 + var2*var2;
    } while(rsquare >= 1.0 || rsquare == 0.0);

    double val = -2.0 * log(rsquare) / rsquare;
    if (val > 0.0) factor = sqrt(val);
    else factor = 0.0; //should not happen, but might due to
    roundoff

    cachevalue = var1 * factor;
    cached=true;

    return (var2 * factor);
}

//-----
//End of Class CRandom
//===== Chromosome.cpp =====
#include "BuildinHeader.h"
#include "Chromosome.h"

CChromosome::CChromosome()
{
    //Allocate memory for the chromosome;
    NVariable=1;
    AVariable = new double [NVariable];
    if ( AVariable == NULL)
    {
        cout << "Fail to allocate memory for AVariable[]!" << endl;
    }
}

```



```

    exit(-1);
}
AVariable = new double[NVariable];
if (AVariable == NULL)
{
    cout << "Fail to allocate memory for AVariable[]" << endl;
    exit(-1);
}
}

CChromosome::CChromosome(int nvariable)
{
    NVariable=nvariable;
    AVariable= new double[NVariable];
    if (AVariable == NULL)
    {
        cout << "Fail to allocate memory for AVariable[]" << endl;
        exit(-1);
    }
    ADeviation = new double[NVariable];
    if (ADeviation == NULL)
    {
        cout << "Fail to allocate memory for ADeviation[]" << endl;
        exit(-1);
    }
}

CChromosome::CChromosome(CChromosome* cc)
{
    if (NVariable!=cc->NVariable)
    {
        delete [] AVariable;
        delete [] ADeviation;
        NVariable=cc->NVariable;
        AVariable = new double[NVariable];
        if (AVariable == NULL)
        {
            cout << "Fail to allocate memory for AVariable[]" << endl;
            exit(-1);
        }
        ADeviation = new double[NVariable];
        if (ADeviation == NULL)
        {
            cout << "Fail to allocate memory for ADeviation[]" << endl;
            exit(-1);
        }
    }
    for (inti=0;i<NVariable;i++)
    {
        AVariable[i]=cc->getvariable(i);
        ADeviation[i]=cc->getdeviation(i);
    }
}

CChromosome::~CChromosome()
{
    if (AVariable) delete [] AVariable;
    if (ADeviation) delete [] ADeviation;
}

void CChromosome::putNVariable(int nvariable)
{
    if (nvariable=0)
    {
        cout << "Error for the parameters in 'putNVariable'" << endl;
        cout << "nvariable=0" << endl;
        exit(-1);
    }
    else if (NVariable!=nvariable)
    {
        delete [] AVariable;
        delete [] ADeviation;
        NVariable=nvariable;
        AVariable = new double[NVariable];
        if (AVariable == NULL)
        {
            cout << "Fail to allocate memory for AVariable[]" << endl;
            exit(-1);
        }
        ADeviation = new double[NVariable];
        if (ADeviation == NULL)
        {
            cout << "Fail to allocate memory for ADeviation[]" << endl;
            exit(-1);
        }
    }
}

int CChromosome::getNVariable()
{
    return NVariable;
}

double CChromosome::putvariable (int i,double x)
{
    try
    {
        if (i<NVariable)
        {
            AVariable[i]=xi;
            return 1;
        }
        else throw "Error in CChromosome::putvariable:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= NVariable!" << endl;
        cout << " Program returns -1 " << endl;
        return -1.0;
    }
}

double CChromosome::putdeviation (int i,double di)
{
    try
    {
        if (i<NVariable)
        {
            ADeviation[i]=di;
            return 1;
        }
        else throw "Error in CChromosome::putdeviation:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= NVariable!" << endl;
        cout << " Program returns -1 " << endl;
        return -1.0;
    }
}

double CChromosome::getvariable (int i)
{
    try
    {
        if (i<NVariable) return AVariable[i];
        else throw "Error in CChromosome::getvariable:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= NVariable!" << endl;
        cout << " Program returns -1 " << endl;
        return -1.0;
    }
}

double CChromosome::getdeviation (int i)
{
    try
    {
        if (i<NVariable) return ADeviation[i];
        else throw "Error in CChromosome::getdeviation:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= NVariable!" << endl;
        cout << " Program returns -1 " << endl;
        return -1.0;
    }
}

//-----
//End of Class CChromosome
//=====
1.1----- Complex.cpp =====
#include "BuildinHeader.h"
#include "Complex.h"

CComplex::CComplex()
{
    x=0;
    y=0;
}

CComplex::CComplex(const double a, const double b)
{
    x=a;
    y=b;
}

CComplex::~CComplex()
{
}

void CComplex::real(const double a)
{
    x=a;
}

void CComplex::imag(const double b)

```

```

{
    y=b;
}

double CComplex::real()
{
    return x;
}

double CComplex::imag()
{
    return y;
}

//Operators for Complex-----//
void CComplex::operator=(const CComplex& C)
{
    x=C.x;
    y=C.y;
}

void CComplex::operator=(const double a)
{
    x=a;y=0.0;
}

CComplex operator+(const CComplex& CC1, const CComplex& CC2)
{
    CComplex CC;
    CC.x=CC1.x+CC2.x;
    CC.y=CC1.y+CC2.y;
    return CC;
}

CComplex operator+(const double a,const CComplex& CC1)
{
    CComplex CC;
    CC.x=a+CC1.x;
    CC.y=CC1.y;
    return CC;
}

CComplex operator+(const CComplex& CC1,const double a)
{
    CComplex CC;
    CC.x=CC1.x+a;
    CC.y=CC1.y;
    return CC;
}

CComplex operator-(const CComplex& CC1)
{
    CComplex CC;
    CC.x=-CC1.x;
    CC.y=-CC1.y;
    return CC;
}

CComplex operator-(const CComplex& CC1,const CComplex& CC2)
{
    CComplex CC;
    CC.x=CC1.x-CC2.x;
    CC.y=CC1.y-CC2.y;
    return CC;
}

CComplex operator-(const CComplex& CC1,const double a)
{
    CComplex CC;
    CC.x=CC1.x-a;
    CC.y=CC1.y;
    return CC;
}

CComplex operator-(const double a,const CComplex& CC1)
{
    CComplex CC;
    CC.x=a-CC1.x;
    CC.y=-CC1.y;
    return CC;
}

CComplex operator*(const CComplex& CC1,const CComplex& CC2)
{
    CComplex CC;
    CC.x=CC1.x*CC2.x-CC1.y*CC2.y;
    CC.y=CC1.x*CC2.y+CC1.y*CC2.x;
    return CC;
}

CComplex operator*(const CComplex& CC1,const double a)
{
    CComplex CC;
    CC.x=CC1.x*a;
    CC.y=CC1.y*a;
    return CC;
}

CComplex operator*(const double a,const CComplex& CC1)
{
    CComplex CC;
    CC.x=a*CC1.x;
    CC.y=a*CC1.y;
    return CC;
}

CComplex operator/(const CComplex& CC1,const CComplex& CC2)
{
    CComplex CC;
    double a=CC2.x*CC2.x+CC2.y*CC2.y;
    CC.x=(CC1.x*CC2.x+CC1.y*CC2.y)/a;
    CC.y=(CC1.y*CC2.x-CC1.x*CC2.y)/a;
    return CC;
}

CComplex operator/(const CComplex& CC1,const double a)
{
    CComplex CC;
    CC.x=CC1.x/a;
    CC.y=CC1.y/a;
    return CC;
}

CComplex operator/(const double a,const CComplex& CC1)
{
    CComplex CC;
    double b=CC1.x*CC1.x+CC1.y*CC1.y;
    CC.x=a*CC1.x/b;
    CC.y=-a*CC1.y/b;
    return CC;
}

CComplex cexp(const CComplex& CC1)
{
    CComplex CC;
    CC.x=exp(CC1.x)*cos(CC1.y);
    CC.y=exp(CC1.x)*sin(CC1.y);
    return CC;
}

CComplex sin(const CComplex& CC1)
{
    CComplex CC,CCj,CC2,CC3;
    CCj.x=0;CCj.y=1;
    CC2=cexp(CCj*CC1);CC3=cexp(-CCj*CC1);
    CC=(CC2-CC3)/(2*CCj);sin(z)=(ejz-e-jz)/(2j);
    return CC;
}

CComplex cos(const CComplex& CC1)
{
    CComplex CC,CCj,CC2,CC3;
    CCj.x=0;CCj.y=1;
    CC2=cexp(CCj*CC1);CC3=cexp(-CCj*CC1);
    CC=(CC2+CC3)/2;cos(z)=(ejz+e-jz)/2;
    return CC;
}

CComplex tan(const CComplex& CC1)
{
    CComplex CC,CCj,CC2,CC3;
    CCj.x=0;CCj.y=1;
    CC2=cexp(CCj*CC1);CC3=cexp(-CCj*CC1);
    CC=(CC2-CC3)/(CCj*(CC2+CC3));tg(z)=(ejz-e-jz)/(j(ejz+e-jz));
    return CC;
}

CComplex sqrt(const CComplex& CC1)
{
    CComplex CC;
    double x=CC1.x; double y=CC1.y;
    if (y==0)
    {
        if (x>=0)
        {
            CC.x=sqrt(x);CC.y=0;
        }
        else
        {
            CC.x=0;CC.y=sqrt(x);
        }
    }
    else
    {
        CC.x=sqrt((sqrt(x*x+y*y)+x)/2.0);
        CC.y=y/(2*CC.x);
    }
    return CC;
}

//===== Poly.cpp =====
#include "BuildinHeader.h"
#include "Poly.h"
CPoly::CPoly()
{
    Nterm=1;
    //allocate memory for the dynamic arrays.
    Xpower = new int [Nterm];
    if( Xpower == NULL)
    {
        cout << "Fail to allocate memory for Xpower[]" << endl;
        exit(-1);
    }
    Ypower = new int [Nterm];
    if( Ypower == NULL)
    {
        cout << "Fail to allocate memory for Ypower[]" << endl;
        exit(-1);
    }
    Coef = new double [Nterm];
    if( Coef == NULL)
    {
        cout << "Fail to allocate memory for Coef[]" << endl;
        exit(-1);
    }
}

```

```

}

CPoly::CPoly(CPoly& CP)
{
    Nterm=CP.Nterm;
    //allocate memory for the dynamic arraies.
    Xpower = new int [Nterm];
    if( Xpower == NULL)
    {
        cout << "Fail to allocate memory for Xpower[]" <<endl;
        exit(-1);
    }
    Ypower = new int [Nterm];
    if( Ypower == NULL)
    {
        cout << "Fail to allocate memory for Ypower[]" <<endl;
        exit(-1);
    }
    Coef = new double [Nterm];
    if( Coef == NULL)
    {
        cout << "Fail to allocate memory for Coef[]" <<endl;
        exit(-1);
    }

    //Copy from CP.
    for (int i=0;i<Nterm;i++)
    {
        Xpower[i] = CP.Xpower[i];
        Ypower[i] = CP.Ypower[i];
        Coef[i] = CP.Ccoef[i];
    }
}

CPoly::CPoly(int N)
{
    if (N<=0)
    {
        cout<<"Nterm of a polynomial can't be negative"<<endl;
        exit(-1);
    }
    else
    {
        Nterm=N;
        //allocate memory for the dynamic arraies.
        Xpower = new int [Nterm];
        if( Xpower == NULL)
        {
            cout << "Fail to allocate memory for Xpower[]" <<endl;
            exit(-1);
        }
        Ypower = new int [Nterm];
        if( Ypower == NULL)
        {
            cout << "Fail to allocate memory for Ypower[]" <<endl;
            exit(-1);
        }
        Coef = new double [Nterm];
        if( Coef == NULL)
        {
            cout << "Fail to allocate memory for Coef[]" <<endl;
            exit(-1);
        }
    }
}

CPoly::~CPoly()
{
    //de-allocate dynamic memory.
    if (Xpower) delete [] Xpower;
    if (Ypower) delete [] Ypower;
    if (Coef) delete [] Coef;
}

void CPoly init_Cpoly(int Nt)
{
    if (Nterm!=Nt)
    {
        if (Xpower) delete [] Xpower;
        if (Ypower) delete [] Ypower;
        if (Coef) delete [] Coef;
        Nterm=Nt;
        //allocate memory for the dynamic arraies
        Xpower = new int [Nterm];
        if( Xpower == NULL)
        {
            cout << "Fail to allocate memory for Xpower[]" <<endl;
            exit(-1);
        }
        Ypower = new int [Nterm];
        if( Ypower == NULL)
        {
            cout << "Fail to allocate memory for Ypower[]" <<endl;
            exit(-1);
        }
        Coef = new double [Nterm];
        if( Coef == NULL)
        {
            cout << "Fail to allocate memory for Coef[]" <<endl;
            exit(-1);
        }
    }
}

if( Coef == NULL)
{
    cout << "Fail to allocate memory for Coef[]" <<endl;
    exit(-1);
}

//Integration on the domain [Lx,Ux][Ly,Uy]
double CPoly::intexy(double Ux,double Lx,double Uy,double Ly)
{
    double t=0.0;
    for (i=0;i<Nterm;i++)
    {
        t+=Coef[i]*1.0/(Xpower[i]+1.0)*(pow(Ux,Xpower[i]+1)-
            pow(Lx,Xpower[i]+1))
            *1.0/(Ypower[i]+1.0)*(pow(Uy,Ypower[i]+1)-
            pow(Ly,Ypower[i]+1));
    }
    return t;
}

//Integrate x or y while y or x is set as val.
double CPoly::intex_y(int index,double val,double Ub,double Lb)
{
    double t=0.0;
    switch (index)
    {
        case 1://x is set as val and integrate y;
            for (i=0;i<Nterm;i++)
            {
                t+=Coef[i]*pow(val,Xpower[i])*1.0/(Ypower[i]+1)
                    *(pow(Ub,Ypower[i]+1)-pow(Lb,Ypower[i]+1));
            }
            break;
        case 2://y is set as val and integrate x;
            for (i=0;i<Nterm;i++)
            {
                t+=Coef[i]*pow(val,Ypower[i])*1.0/(Xpower[i]+1)
                    *(pow(Ub,Xpower[i]+1)-pow(Lb,Xpower[i]+1));
            }
            break;
        default:
            cout<<"Error in index. index== "<<index<<endl;
            cout<<"program exit -1!"<<endl;
            exit(-1);
    }
    return t;
}

//Get the value on the point (x,y)
double CPoly::getValue(double x,double y)
{
    double t=0.0;
    for (int i=0;i<Nterm;i++)
    {
        t+=Coef[i]*pow(x,Xpower[i])*pow(y,Ypower[i]);
    }
    return t;
}

int CPoly::getNterm()
{
    return Nterm;
}

void CPoly::putNterm(int N)
{
    if (Nterm!=N)
    {
        delete [] Xpower;
        delete [] Ypower;
        delete [] Coef;
        Nterm=N;
        //allocate memory for the dynamic arraies
        Xpower = new int [Nterm];
        if( Xpower == NULL)
        {
            cout << "Fail to allocate memory for Xpower[]" <<endl;
            exit(-1);
        }
        Ypower = new int [Nterm];
        if( Ypower == NULL)
        {
            cout << "Fail to allocate memory for Ypower[]" <<endl;
            exit(-1);
        }
        Coef = new double [Nterm];
        if( Coef == NULL)
        {
            cout << "Fail to allocate memory for Coef[]" <<endl;
            exit(-1);
        }
    }
}

void CPoly::putCoef(int i,double xi)
{
    try
    {
        if (i<Nterm)
        {
            Coef[i]=xi;
            return;
        }
    }
}

```

```

    else throw "Error in CPoly::putCoef";
}
catch (char *e)
{
    cout << e << endl;
    cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
    cout << " Programexit -1." << endl;
    exit(-1);
}
}

void CPoly::putXpower(int i, int xi)
{
    try
    {
        if (i < Nterm)
        {
            Xpower[i] = xi;
            return;
        }
        else throw "Error in CPoly::putXpower";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Programexit -1." << endl;
        exit(-1);
    }
}

void CPoly::putYpower(int i, int yi)
{
    try
    {
        if (i < Nterm)
        {
            Ypower[i] = yi;
            return;
        }
        else throw "Error in CPoly::putYpower";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Programexit -1." << endl;
        exit(-1);
    }
}

double CPoly::getCoef (int i)
{
    try
    {
        if (i < Nterm) return Coef[i];
        else throw "Error in CPoly::getCoef:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Programexit -1." << endl;
        exit(-1);
    }
}

int CPoly::getXpower (int i)
{
    try
    {
        if (i < Nterm) return Xpower[i];
        else throw "Error in CPoly::getXpower:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Programexit -1." << endl;
        exit(-1);
    }
}

int CPoly::getYpower (int i)
{
    try
    {
        if (i < Nterm) return Ypower[i];
        else throw "Error in CPoly::getYpower:";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Programexit -1." << endl;
        exit(-1);
    }
}

    cout << " Programexit -1." << endl;
    exit(-1);
}

}

void CPoly simplify()
{
    int i, Nt;
    Nt = Nterm;
    int *x, *y;
    double *a;
    a = new double[Nt];
    if (a == NULL)
    {
        cout << "Fail to allocate memory for a[] !" << endl;
        exit(-1);
    }
    x = new int[Nt];
    if (x == NULL)
    {
        cout << "Fail to allocate memory for x[] !" << endl;
        exit(-1);
    }
    y = new int[Nt];
    if (y == NULL)
    {
        cout << "Fail to allocate memory for y[] !" << endl;
        exit(-1);
    }

    for (j = 0; j < Nt; j++) { x[j] = Xpower[j]; y[j] = Ypower[j]; a[j] = Coef[j]; }
    //eliminate the terms with the same power.
    for (j = 0; j < Nt; j++)
    {
        for (int k = j + 1; k < Nt; k++)
        {
            if ((x[j] == x[k]) && (y[j] == y[k]))
            {
                a[j] = a[j] + a[k];
                for (int l = j; l < Nt; l++)
                {
                    a[l] = a[l + 1];
                    x[l] = x[l + 1];
                    y[l] = y[l + 1];
                }
                if (Nt > 1) Nt = Nt - 1;
            }
        }
    }
    //eliminate the terms with coef=0.
    i = 0;
    while (i < Nt)
    {
        while ((fabs(a[i]) < 1.0e-8) && (Nt > i))
        {
            for (int j = i; j < Nt - 1; j++)
            {
                a[j] = a[j + 1];
                x[j] = x[j + 1];
                y[j] = y[j + 1];
            }
            Nt--;
        }
        i++;
    }

    putNterm(Nt);
    for (i = 0; i < Nt; i++)
    {
        Xpower[i] = x[i]; Ypower[i] = y[i]; Coef[i] = a[i];
        if (a) delete [] a;
        if (x) delete [] x;
        if (y) delete [] y;
    }
    return;
}

CPoly & CPoly::operator =(const CPoly & CP)
{
    putNterm(CP.Nterm);
    for (inti = 0; i < Nterm; i++)
    {
        Coef[i] = CP.Coeff[i];
        Xpower[i] = CP.Xpower[i];
        Ypower[i] = CP.Ypower[i];
    }
    return *this;
}

//Operators for poly-----^-----//
CPoly operator+(CPoly & CP1, CPoly & CP2)
{
    int i, j;
    int Nterm1, Nterm2;
    Nterm1 = CP1.getNterm();
    Nterm2 = CP2.getNterm();
    int Nterm = Nterm1 + Nterm2;
    CPoly CP; CP.putNterm(Nterm);
    for (i = 0; i < Nterm1; i++)

```

```

{
    CP.putCoeff(i,CP1.getCoeff(i));
    CP.putXpower(i,CP1.getXpower(i));
    CP.putYpower(i,CP1.getYpower(i));
}
for (i=0;i<Nterm2;i++)
{
    j=i+Nterm1;
    CP.putCoeff(j,CP2.getCoeff(i));
    CP.putXpower(j,CP2.getXpower(i));
    CP.putYpower(j,CP2.getYpower(i));
}
CP.simplify();
return CP;
}

CPoly operator-(CPoly& CP1,CPoly&CP2)
{
    int i,j;
    int Nterm1,Nterm2;
    Nterm1=CP1.getNterm();
    Nterm2=CP2.getNterm();
    int Nterm=Nterm1+Nterm2;
    CPoly C P CP.putNterm(Nterm);
    for (i=0;i<Nterm1;i++)
    {
        CP.putCoeff(i,CP1.getCoeff(i));
        CP.putXpower(i,CP1.getXpower(i));
        CP.putYpower(i,CP1.getYpower(i));
    }
    for (i=0;i<Nterm2;i++)
    {
        j=i+Nterm1;
        CP.putCoeff(j,-CP2.getCoeff(i));
        CP.putXpower(j,CP2.getXpower(i));
        CP.putYpower(j,CP2.getYpower(i));
    }

    CP.simplify();
    return CP;
}

CPoly operator*(CPoly& CP1,CPoly& CP2)
{
    int i,j;
    int Nterm1,Nterm2;
    Nterm1=CP1.getNterm();
    Nterm2=CP2.getNterm();
    int Nterm=Nterm1*Nterm2;
    CPoly C P CP.putNterm(Nterm);
    for (i=0;i<Nterm1;i++)
    {
        for (j=0;j<Nterm2;j++)
        {
            CP.putXpower(i*Nterm2+j,CP1.getXpower(i)+CP2.getXpower(j));
            CP.putYpower(i*Nterm2+j,CP1.getYpower(i)+CP2.getYpower(j));
            CP.putCoeff(i*Nterm2+j,CP1.getCoeff(i)*CP2.getCoeff(j));
        }
    }

    CP.simplify();
    return CP;
}

CPoly operator*(double a,CPoly& CP1)
{
    int i;
    int Nterm1=CP1.getNterm();
    int Nterm=Nterm1;
    CPoly CP; CP.putNterm(Nterm);
    for (i=0;i<Nterm1;i++)
    {
        CP.putXpower(i,CP1.getXpower(i));
        CP.putYpower(i,CP1.getYpower(i));
        CP.putCoeff(i,a*CP1.getCoeff(i));
    }

    CP.simplify();
    return CP;
}

CPoly operator*(CPoly& CP1,double a)
{
    int i;
    int Nterm1=CP1.getNterm();
    int Nterm=Nterm1;
    CPoly C P CP.putNterm(Nterm);
    for (i=0;i<Nterm1;i++)
    {
        CP.putXpower(i,CP1.getXpower(i));
        CP.putYpower(i,CP1.getYpower(i));
        CP.putCoeff(i,a*CP1.getCoeff(i));
    }

    CP.simplify();
    return CP;
}

}

CPoly dx(CPoly& CP1)
{
    int Nterm=CP1.getNterm();
    CPoly CP; CP.putNterm(Nterm);
    for (int i=0;i<Nterm;i++)
    {
        if (CP1.getXpower(i) == 0)
            CP.putCoeff(i,0.0); CP.putXpower(i,0);CP.putYpower(i,0);
        else
        {
            CP.putCoeff(i,CP1.getCoeff(i)*CP1.getXpower(i));
            CP.putXpower(i,CP1.getXpower(i)-1);
            CP.putYpower(i,CP1.getYpower(i));
        }
    }
    CP.simplify();
    return CP;
}

CPoly dy(CPoly& CP1)
{
    int Nterm=CP1.getNterm();
    CPoly CP; CP.putNterm(Nterm);
    for (int i=0;i<Nterm;i++)
    {
        if (CP1.getYpower(i) == 0)
            CP.putCoeff(i,0.0); CP.putXpower(i,0);CP.putYpower(i,0);
        else
        {
            CP.putCoeff(i,CP1.getCoeff(i)*CP1.getYpower(i));
            CP.putYpower(i,CP1.getYpower(i)-1);
            CP.putXpower(i,CP1.getXpower(i));
        }
    }
    CP.simplify();
    return CP;
}

//End of Operators for Poly
-----
----- Polyx.cpp =====
#include "BuildinHeader.h"
#include "Polyx.h"

CPolyx::CPolyx()
{
    Nterm=1;
    //allocate memory for the dynamic arraies.
    Xpower = new int [Nterm];
    if (Xpower == NULL)
    {
        cout << "Fail to allocate memory for Xpower[]" << endl;
        exit(-1);
    }

    Coef = new double [Nterm];
    if (Coef == NULL)
    {
        cout << "Fail to allocate memory for Coef[]" << endl;
        exit(-1);
    }
}

CPolyx::CPolyx(CPolyx& CP)
{
    Nterm=CP.Nterm;
    //allocate memory for the dynamic arraies.
    Xpower = new int [Nterm];
    if (Xpower == NULL)
    {
        cout << "Fail to allocate memory for Xpower[]" << endl;
        exit(-1);
    }

    Coef = new double [Nterm];
    if (Coef == NULL)
    {
        cout << "Fail to allocate memory for Coef[]" << endl;
        exit(-1);
    }

    //Copy from CP.
    for (int i=0;i<Nterm;i++)
    {
        Xpower[i] = CP.Xpower[i];
        Coef[i] = CP.Coeff[i];
    }
}

CPolyx::CPolyx(int N)
{
    if (N<=0)
    {
        cout << "Nterm of a polynomial can't be negative" << endl;
        exit(-1);
    }
    else
    {

```

```

Nterm=Nt;
//allocate memory for the dynamic arraies.
Xpower = new int [Nterm]
if( Xpower == NULL)
{
    cout << "Fail to allocate memory for Xpower[]"!<<endl;
    exit(-1);
}
Coef = new double [Nterm];
if( Coef == NULL)
{
    cout << "Fail to allocate memory for Coef[]"!<<endl;
    exit(-1);
}
}
}

CPolyx::~CPolyx()
{
    //de-allocated dynamic memory.
    if (Xpower)
    {
        delete[] Xpower;
        Xpower=0;
    }
    if (Coef)
    {
        delete[] Coef;
        Coef=0;
    }
}

int CPolyx::getNterm()
{
    return Nterm;
}

void CPolyx::putCoef(int i,double xi)
{
    try
    {
        if (i<Nterm)
        {
            Coef[i]=xi;
            return;
        }
        else throw "Error in CPoly::putCoef";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Program exit -1." << endl;
        exit(-1);
    }
}

void CPolyx::putXpower(int i,int xi)
{
    try
    {
        if (i<Nterm)
        {
            Xpower[i]=xi;
            return;
        }
        else throw "Error in CPoly::putXpower";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Program exit -1." << endl;
        exit(-1);
    }
}

double CPolyx::getCoef (int i)
{
    try
    {
        if (i<Nterm) return Coef[i];
        else throw "Error in CPoly::getCoef";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Program exit -1." << endl;
        exit(-1);
    }
}

int CPolyx::getXpower (int i)
{
    try
    {
        if (i<Nterm) return Xpower[i];
        else throw "Error in CPoly::getXpower";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " >= Nterm!" << endl;
        cout << " Program exit -1." << endl;
        exit(-1);
    }
}

void CPolyx::init(int Nt)
{
    if (Nterm!=Nt)
    {
        delete[] Xpower;
        delete[] Coef;
        Nterm=Nt;
        //allocate memory for the dynamic arraies
        Xpower = new int [Nterm];
        if( Xpower == NULL)
        {
            cout << "Fail to allocate memory for Xpower[]"!<<endl;
            exit(-1);
        }
        Coef = new double [Nterm];
        if( Coef == NULL)
        {
            cout << "Fail to allocate memory for Coef[]"!<<endl;
            exit(-1);
        }
    }
}

void CPolyx::putNterm(int N)
{
    if (Nterm!=N)
    {
        delete[] Xpower;
        delete[] Coef;
        Nterm=N;
        //allocate memory for the dynamic arraies
        Xpower = new int [Nterm];
        if( Xpower == NULL)
        {
            cout << "Fail to allocate memory for Xpower[]"!<<endl;
            exit(-1);
        }
        Coef = new double [Nterm];
        if( Coef == NULL)
        {
            cout << "Fail to allocate memory for Coef[]"!<<endl;
            exit(-1);
        }
    }
}

double CPolyx::getValue(double x)
{
    double t=0;
    for (int i=0; i<Nterm; i++)
        t+=Coef[i]*pow(x,Xpower[i]);
    return t;
}

double CPolyx::inte(double Ux,double Lx)
{
    double t=0.0;
    int i;
    for (i=0;i<Nterm;i++)
        t+=Coef[i]*1.0/(Xpower[i]+1.0)*(pow(Ux,Xpower[i]+1)-
        pow(Lx,Xpower[i]+1));
    return t;
}

void CPolyx::simplify()
{
    int i,Nt;
    Nt=Nterm;
    double *a=new double [Nt];
    if( a == NULL)
    {
        cout << "Fail to allocate memory for a[]"!<<endl;
        exit(-1);
    }
    int *x=new int [Nt];
    if( x == NULL)
    {
        cout << "Fail to allocate memory for x[]"!<<endl;
        exit(-1);
    }
    for (i=0;i<Nt;i++) {x[i]=Xpower[i];a[i]=Coef[i];}
    //eliminate the terms with the same power.
}

```

```

for (i=0; i<Nt; i++)
{
    for (int j=i+1; j<Nt; j++)
    {
        if ((x[i]==x[j]))
        {
            a[i]=a[i]+a[j];
            for (int k=j; k<Nt; k++)
            {
                a[k]=a[k+1];
                x[k]=x[k+1];
            }
            if (Nt>1) Nt=Nt-1;
        }
    }
}
//eliminate the terms with coef=0.
i=0;
while (i<Nt)
{
    while ((fabs(a[i])<1.0e-8)&&(Nt>i))
    {
        for (int j=i; j<Nt-1; j++)
        {
            a[j]=a[j+1];
            x[j]=x[j+1];
        }
        Nt--;
    }
    i++;
}

putNterm(Nt);
for (i=0; i<Nt; i++)
{ Xpower[i]=x[i]; Coef[i]=a[i]; }

if (a) delete[] a;
if (x) delete[] x;

return;
}

CPolyx &CPolyx::operator =(const CPolyx &CP)
{
    putNterm(CP.Nterm);
    for (int i=0; i<Nterm; i++)
    {
        Coef[i]=CP.Ccoef[i];
        Xpower[i]=CP.Xpower[i];
    }
    return "this";
}

CPolyx operator+(CPolyx &CP1, CPolyx &CP2)
{
    int i, j;
    int Nterm1, Nterm2;
    Nterm1=CP1.getNterm();
    Nterm2=CP2.getNterm();
    int Nterm=Nterm1+Nterm2;
    CPolyx CP; CP.putNterm(Nterm);
    for (i=0; i<Nterm1; i++)
    {
        CP.putCoef(i, CP1.getCoef(i));
        CP.putXpower(i, CP1.getXpower(i));
    }
    for (i=0; i<Nterm2; i++)
    {
        j=i+Nterm1;
        CP.putCoef(j, CP2.getCoef(i));
        CP.putXpower(j, CP2.getXpower(i));
    }
    CP.simplify();
    return CP;
}

CPolyx operator-(CPolyx &CP1, CPolyx &CP2)
{
    int i, j;
    int Nterm1, Nterm2;
    Nterm1=CP1.getNterm();
    Nterm2=CP2.getNterm();
    int Nterm=Nterm1+Nterm2;
    CPolyx CP; CP.putNterm(Nterm);
    for (i=0; i<Nterm1; i++)
    {
        CP.putCoef(i, CP1.getCoef(i));
        CP.putXpower(i, CP1.getXpower(i));
    }
    for (i=0; i<Nterm2; i++)
    {
        j=i+Nterm1;
        CP.putCoef(j, -CP2.getCoef(i));
        CP.putXpower(j, CP2.getXpower(i));
    }
    CP.simplify();
    return CP;
}

```

```

}

CPolyx operator*(CPolyx &CP1, CPolyx &CP2)
{
    int i, j;
    int Nterm1, Nterm2;
    Nterm1=CP1.getNterm();
    Nterm2=CP2.getNterm();
    int Nterm=Nterm1*Nterm2;
    CPolyx CP; CP.putNterm(Nterm);
    for (i=0; i<Nterm1; i++)
    {
        for (j=0; j<Nterm2; j++)
        {
            CP.putXpower(i*Nterm2+j, CP1.getXpower(i)+CP2.getXpower(j));
            CP.putCoef(i*Nterm2+j, CP1.getCoef(i)*CP2.getCoef(j));
        }
    }
    CP.simplify();
    return CP;
}

CPolyx operator*(double a, CPolyx &CP1)
{
    int i;
    int Nterm1=CP1.getNterm();
    int Nterm=Nterm1;
    CPolyx CP; CP.putNterm(Nterm);
    for (i=0; i<Nterm1; i++)
    {
        CP.putXpower(i, CP1.getXpower(i));
        CP.putCoef(i, a*CP1.getCoef(i));
    }
    CP.simplify();
    return CP;
}

CPolyx operator*(CPolyx &CP1, double a)
{
    int i;
    int Nterm1=CP1.getNterm();
    int Nterm=Nterm1;
    CPolyx CP; CP.putNterm(Nterm);
    for (i=0; i<Nterm1; i++)
    {
        CP.putXpower(i, CP1.getXpower(i));
        CP.putCoef(i, a*CP1.getCoef(i));
    }
    CP.simplify();
    return CP;
}

CPolyx dx(CPolyx &CP1)
{
    int Nterm=CP1.getNterm();
    CPolyx CP; CP.putNterm(Nterm);
    for (int i=0; i<Nterm; i++)
    {
        if (CP1.getXpower(i) == 0)
        {
            CP.putCoef(i, 0.0);
            CP.putXpower(i, 0);
        }
        else
        {
            CP.putCoef(i, CP1.getCoef(i)*CP1.getXpower(i));
            CP.putXpower(i, CP1.getXpower(i)-1);
        }
    }
    CP.simplify();
    return CP;
}

//===== Parameter.cpp =====
#include "Parameter.h"

CParameter::CParameter()
{
    Max_gen=1;
    NPop_size=1;
    MaxMinID=1;
    NPara_prob=1;
    SPara_prob = new char*[NPara_prob];
    if (SPara_prob == NULL)
    {
        cout << "Fail to allocate memory for SPara_prob[]" << endl;
        exit(-1);
    }
    for (int i=0; i<NPara_prob; i++)
    {
        SPara_prob[i]=new char[20];
        if (SPara_prob[i] == NULL)
        {
            cout << "Fail to allocate memory for SPara_prob[i]" << endl;
            exit(-1);
        }
    }
    APara_prob = new double[NPara_prob];
    if (APara_prob == NULL)
    {

```



```

    cout << "Fail to allocate memory for APara_prob[] !" << endl;
    exit(-1);
}

NVariable=1;
SPara_scal = new char* [NVariable];
if( SPara_scal == NULL)
{
    cout << "Fail to allocate memory for SPara_scal[] !" << endl;
    exit(-1);
}
for (i=0;j<NVariable;i++)
{
    SPara_scal[i]=new char [20];
    if( SPara_scal[i] == NULL)
    {
        cout << "Fail to allocate memory for SPara_scal[i][]" << endl;
        exit(-1);
    }
}
APara_scal = new double [2*NVariable];
if( APara_scal == NULL)
{
    cout << "Fail to allocate memory for APara_scal[] !" << endl;
    exit(-1);
}
}

CParameter::CParameter(int npara_prob, int nvariable)
{
    NPara_prob=npara_prob;
    SPara_prob = new char *[NPara_prob];
    if( SPara_prob == NULL)
    {
        cout << "Fail to allocate memory for SPara_prob[] !" << endl;
        exit(-1);
    }
    for (int i=0;j<NPara_prob;i++)
    {
        SPara_prob[i]=new char [20];
        if( SPara_prob[i] == NULL)
        {
            cout << "Fail to allocate memory for SPara_prob[i][]" << endl;
            exit(-1);
        }
    }
    APara_prob = new double [NPara_prob];
    if( APara_prob == NULL)
    {
        cout << "Fail to allocate memory for APara_prob[] !" << endl;
        exit(-1);
    }
}

NVariable=nvariable;
SPara_scal = new char* [NVariable];
if( SPara_scal == NULL)
{
    cout << "Fail to allocate memory for SPara_scal[] !" << endl;
    exit(-1);
}
for (i=0;j<NVariable;i++)
{
    SPara_scal[i]=new char [20];
    if( SPara_scal[i] == NULL)
    {
        cout << "Fail to allocate memory for SPara_scal[i][]" << endl;
        exit(-1);
    }
}
APara_scal = new double [2*NVariable];
if( APara_scal == NULL)
{
    cout << "Fail to allocate memory for APara_scal[] !" << endl;
    exit(-1);
}
}

CParameter::~CParameter()
{
    if (NPara_prob!=0)
    {
        for (int i=0;j<NPara_prob;i++)
        {
            delete SPara_prob[i];
            delete [] SPara_prob;
            delete [] APara_prob;
        }
    }

    if (NVariable!=0)
    {
        for (int i=0;j<NVariable;i++)
        {
            delete SPara_scal[i];
            delete [] SPara_scal;
            delete [] APara_scal;
        }
    }
}

void CParameter::setNPara_prob(int npara_prob)
{
    if (NPara_prob!=npara_prob)
    {
        if (NPara_prob!=0)
        {
            for (int i=0;j<NPara_prob;i++)
            {
                delete SPara_prob[i];
                delete [] SPara_prob;
                delete [] APara_prob;
            }
        }
        NPara_prob=npara_prob;
        SPara_prob = new char *[NPara_prob];
        if( SPara_prob == NULL)
        {
            cout << "Fail to allocate memory for SPara_prob[] !" << endl;
            exit(-1);
        }
        for (int i=0;j<NPara_prob;i++)
        {
            SPara_prob[i]=new char [20];
            if( SPara_prob[i] == NULL)
            {
                cout << "Fail to allocate memory for SPara_prob[i][]" << endl;
                exit(-1);
            }
        }
        APara_prob = new double [NPara_prob];
        if( APara_prob == NULL)
        {
            cout << "Fail to allocate memory for APara_prob[] !" << endl;
            exit(-1);
        }
    }
}

void CParameter::setNVariable(int nvariable)
{
    if (NVariable!=nvariable)
    {
        if (NVariable!=0)
        {
            for (int i=0;j<NVariable;i++)
            {
                delete SPara_scal[i];
                delete [] SPara_scal;
                delete [] APara_scal;
            }
        }
        NVariable=nvariable;
        SPara_scal = new char* [NVariable];
        if( SPara_scal == NULL)
        {
            cout << "Fail to allocate memory for SPara_scal[] !" << endl;
            exit(-1);
        }
        for (int i=0;j<NVariable;i++)
        {
            SPara_scal[i]=new char [20];
            if( SPara_scal[i] == NULL)
            {
                cout << "Fail to allocate memory for SPara_scal[i][]" << endl;
                exit(-1);
            }
        }
        APara_scal = new double [2*NVariable];
        if( APara_scal == NULL)
        {
            cout << "Fail to allocate memory for APara_scal[] !" << endl;
            exit(-1);
        }
    }
}

CParameter::getMax_gen()
{
    return Max_gen;
}

CParameter::getPop_size()
{
    return NPop_size;
}

CParameter::getNTouma_size()
{
    return NTouma_size;
}

CParameter::getMaxMinID()
{
    return MaxMinID;
}

CParameter::getNVariable()
{
    return NVariable;
}

double CParameter::getpara_prob (int i)

```

```

    if (i<NPara_prob) return APara_prob[i];
    else throw "Error in CParameter::getpara_prob:";
}
catch (char *e)
{
    cout << e << endl;
    cout << " i=" << setw(2) << i << " >= NPara_prob!" << endl;
    cout << " Program returns -1." << endl;
    return -1.0;
}
}

double CParameter::getpara_scal (int i, char c)
{
    try
    {
        if (i<NVariable)
        {
            if ((c=='L')||(c=='I')) return APara_scal[2*i];
            if ((c=='U')||(c=='u')) return APara_scal[2*i+1];
            cout << "Error in parameter 2 in function CP getpara_scal"
                << endl;
            return 0;
        }
        else throw "Error in CParameter::getpara_scal ";
    }
    catch (char *e)
    {
        cout << e << endl;
        cout << " i=" << setw(2) << i << " c=" << c << " >= NVariable!" << endl;
        cout << " Program returns -1." << endl;
        return -1.0;
    }
}

int CParameter::getStrTyp()
{
    return StrTyp;
}

istream& operator>> (istream& is, CParameter& cp)
{
    is>>cp.SMaxMinID>>cp.SPop_size>>cp.SMax_gen
    >>cp.SNtourna_size;
    is>>cp.MaxMinID>>cp.NPop_size>>cp.Max_gen>>cp.NTourn_size;
    int nvariable, npara_prob;
    is>>cp.SNvariable>>cp.SNpara_prob;
    is>>nvariable>>npara_prob;

    cp.setNPara_prob(npara_prob);
    cp.setNVariable(nvariable);

    for (i=0;i<cp.NPara_prob;i++)
        is>>cp.SPara_prob[i];
    for (i=0;i<cp.NPara_prob;i++)
        is>>cp.APara_prob[i];
    for (i=0;i<cp.NVariable;i++)
        is>>cp.SPara_scal[i];
    for (i=0;i<2*cp.NVariable;i++)
        is>>cp.APara_scal[i];
    is>>cp.SStrTyp;
    is>>cp.StrTyp;
    return is;
}

ostream& operator<< (ostream& os, const CParameter& cp)
{
    os<<setw(20)<<cp.SMaxMinID<<setw(20)<<cp.SPop_size
    <<setw(20)<<cp.SMax_gen<<setw(20)<<cp.SNtourna_size
    <<endl;
    os<<setw(20)<<cp.MaxMinID<<setw(20)<<cp.NPop_size
    <<setw(20)<<cp.Max_gen<<setw(20)<<cp.NTourn_size
    <<endl;
    os<<setw(20)<<cp.SNvariable<<setw(20)<<cp.SNpara_prob<<endl
    os<<setw(20)<<cp.NVariable<<setw(20)<<cp.NPara_prob<<endl
    <<endl;

    int npara_prob=cp.NPara_prob;
    for (inti=0;i<npara_prob;i++)
        os<<setw(10)<<cp.SPara_prob[i]; os<<endl;
    for (i=0;i<npara_prob;i++)
        os<<setw(10)<<cp.APara_prob[i]; os<<endl<<endl;
    int nvariable=cp.NVariable;
    for (i=0;i<nvariable;i++)
        os<<setw(20)<<cp.SPara_scal[i]; os<<endl;
    for (i=0;i<2*nvariable;i++)
        os<<setw(10)<<cp.APara_scal[i]; os<<endl<<endl;
    os<<setw(10)<<cp.SStrTyp<<endl;
    os<<setw(10)<<cp.StrTyp<<endl;
    return os;
}

//=====
//End of Class CParameter
//===== Struct.cpp =====

```

```

#include "Struct.h"

CStruct::CStruct()
{
    PI=3.1415926535897932384626433832795;
    CCj.real(0); CCj.imag(1);
    NumEval=0;
    Np=1; mnl=1; NNterm=1;
    KO=new double [mnl*mnl];
    if (KO==NULL)
    {
        cout<<"Fail to allocate memory for KO!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    MO=new double [mnl*mnl];
    if (MO==NULL)
    {
        cout<<"Fail to allocate memory for MO!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    K=new double [mnl*mnl];
    if (K==NULL)
    {
        cout<<"Fail to allocate memory for KO!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    M=new double [mnl*mnl];
    if (M==NULL)
    {
        cout<<"Fail to allocate memory for MJ!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    Eval=new double [mnl];
    if (Eval==NULL)
    {
        cout<<"Fail to allocate memory for Eval!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    Evec=new double [mnl*mnl];
    if (Evec==NULL)
    {
        cout<<"Fail to allocate memory for Evec!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
}

CStruct::~CStruct()
{
    if (KO) delete[] KO;
    if (MO) delete[] MO;
    if (K) delete[] K;
    if (M) delete[] M;
    if (Eval) delete[] Eval;
    if (Evec) delete[] Evec;
}

int CStruct::Allocate(int i, int j, int mnl)
{
    return i*mnl+j;
}

void CStruct::setBoundary(){}
void CStruct::normalize(){}
void CStruct::stiffmass(){}
void CStruct::stiffmass(CChromosome* cc){}

void CStruct::Eigen()
{
    longint dim=mnl;
    DGVCSPI(&dim,K,&dim,M,&dim,Eval,Evec,&dim);
}

void CStruct::Relang_each_model(){}
void CStruct::factor(){}
double CStruct::Conduct(double Omega)
{
    double conduct =0;
    return conduct;
}

void CStruct::GetPeak()//output: CalPeak
{
    //scan with a step 0.1KHz
    int N=int((UF-LF)/0.1)+1;
    double *conduct=new double [N];
    if (conduct==NULL)
    {
        cout<<"Fail to allocate memory for conduct!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
}

```

```

for (inti=0;i<N;i++)
{
    double Omega=(LF+i*0.1)*2000*PI/Omega0;
    conduct[i]=Conduct(Omega);
    PeakNum=0;
    for (i=1;i<N-1;i++)
    {
        if ((conduct[i]>conduct[i-1])&&(conduct[i]>conduct[i+1]))
        {
            CalPeak[PeakNum]=LF+i*0.1;
            PeakNum++;
            if (PeakNum>100)
            {
                cout<<"Warning! the memory for CalPeak is not
                enough!"<<endl;
            }
        }
    }
    if (conduct) delete [] conduct;

    conduct=new double [20];
    if (conduct==NULL)
    {
        cout<<"Fail to allocate memory for conduct[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    //scan with 0.01KHz
    for (i=0;i<PeakNum;i++)
    {
        for (int j=0;j<20;j++)
        {
            double Omega=(CalPeak[i]-0.1+(j+1)*0.01)*2000*PI/Omega0;
            conduct[j]=Conduct(Omega);
        }
        //find me maximum as the peak
        double conduct0=conduct[0];
        int peak=0;
        for (j=1;j<20;j++)
        {
            if (conduct[j]>conduct0)
            {
                conduct0=conduct[j];
                peak=j;
            }
        }
        CalPeak[i]=CalPeak[i]-0.1+(peak+1)*0.01;
    }

    //scan with 0.001KHz
    for (i=0;i<PeakNum;i++)
    {
        for (int j=0;j<20;j++)
        {
            double Omega=(CalPeak[i]-0.01+(j+1)*0.001)*2000*PI/Omega0;
            conduct[j]=Conduct(Omega);
        }
        //find me maximum as the peak
        double conduct0=conduct[0];
        int peak=0;
        for (j=1;j<20;j++)
        {
            if (conduct[j]>conduct0)
            {
                conduct0=conduct[j];
                peak=j;
            }
        }
        CalPeak[i]=CalPeak[i]-0.01+(peak+1)*0.001;
    }
    if (conduct) delete [] conduct;
}

double CStruct::fitness(CChromosome* cc)
{
    double fn=0;
    stiffmass(cc);
    Eigen();
    Relang_each_mode();
    factor();
    GetPeak();
    for (inti=0;i<InPeakNum;i++)
    {
        //find a suitable CalPeak;
        double a=fabs(CalPeak[0]-Refer_Peak[i]);
        for (intj=1;j<PeakNum;j++)
        {
            if (fabs(CalPeak[j]-Refer_Peak[i])<a)
            {
                a=fabs(CalPeak[j]-Refer_Peak[i]);
            }
        }
        if (a<0.2) fn+=a;
    }
    NumEval++;
    cout<<"Number of Evaluation : "<<NumEval<<endl;
    return fn;
}

```

```

void CStruct::IOput(char *Infile,ofstream& StrOutfile)
{
    ifstream StrInfile(Infile);
    if (StrInfile.fail())
    {
        cout<<"Error in Main: Unable to open input.dat!"<<endl<<endl;
        exit (-1);
    }
    StrOutfile<<"Good"<<endl;
}

//=====
//----- Beam.cpp -----
#include "Beam.h"

CBeam::CBeam()
{
    Betax=new double [ "Term];
    if (Betax==NULL)
    {
        cout<<"Fail to allocate memory for Betax[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    Vyitax=new double [NNterm];
    if (Vyitax==NULL)
    {
        cout<<"Fail to allocate memory for Vyitax[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    CPK=new CPolyx [mnl*mnl];
    if (CPK==NULL)
    {
        cout<<"Fail to allocate memory for CPK[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    CPM=new CPolyx [mnl*mnl];
    if (CPM==NULL)
    {
        cout<<"Fail to allocate memory for CPM[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
}

CBeam::~CBeam()
{
    if (Betax) delete [] Betax;
    if (Vyitax) delete [] Vyitax;
    if (CPK) delete [] CPK;
    if (CPM) delete [] CPM;
}

void CBeam::Initialize()//Reallocate memory according to Np;
{
    mnl=Np+1; NNterm=mnl;
    if (K0) delete [] K0;
    K0=new double [mnl*mnl];
    if (K0==NULL)
    {
        cout<<"Fail to allocate memory for K0[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (M0) delete [] M0;
    M0=new double [mnl*mnl];
    if (M0==NULL)
    {
        cout<<"Fail to allocate memory for M0[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (K) delete [] K;
    K=new double [mnl*mnl];
    if (K==NULL)
    {
        cout<<"Fail to allocate memory for K[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (M) delete [] M;
    M=new double [mnl*mnl];
    if (M==NULL)
    {
        cout<<"Fail to allocate memory for M[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Eval) delete [] Eval;
    Eval=new double [mnl];
    if (Eval==NULL)
    {
        cout<<"Fail to allocate memory for Eval[]"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
}

```

```

}
if (Evec) delete[] Evec;
Evec=new double [mnl*mnl];
if (Evec==NULL)
{
    cout<<"Fail to allocate memory for Evec[] !"<<endl;
    cout<<"Program exit -1 !"<<endl;
    exit (-1);
}
if (Betax) delete[] Betax;
Betax=new double [ **term];
if (Betax==NULL)
{
    cout<<"Fail to allocate memory for Betax[] !"<<endl;
    cout<<"Program exit -1 !"<<endl;
    exit (-1);
}

if (Vyitax) delete[] Vyitax;
Vyitax=new double [NNterm];
if (Vyitax==NULL)
{
    cout<<"Fail to allocate memory for Vyitax[] !"<<endl;
    cout<<"Program exit -1 !"<<endl;
    exit (-1);
}
if (CPK) delete[] CPK;
CPK=new CPolyx [mnl*mnl];
if (CPK==NULL)
{
    cout<<"Fail to allocate memory for CPK1[] !"<<endl;
    cout<<"Program exit -1 !"<<endl;
    exit (-1);
}
if (CPM) delete[] CPM;
CPM=new CPolyx [mnl*mnl];
if (CPM==NULL)
{
    cout<<"Fail to allocate memory for CPM[] !"<<endl;
    cout<<"Program exit -1 !"<<endl;
    exit (-1);
}
}

void CBeam::normalize()
{
    E0=E;Rou0=Rou;L0=A;
    E=E/E0;Rou=Rou/Rou0;H=H/L0;A=A/L0;B=B/L0;
    Omega0=sqrt(E0/Rou0)/L0;Z0=sqrt(E0*Rou0)*L0*L0;
    F0=E0*L0*L0;//non-dimensionalize

    Uxp=Uxp/L0;Lxp=Lxp/L0;
    Ap=Uxp*Lxp;

    Ep=Ep/E0;Roup=Roup/Rou0;Hp=Hp/L0;
    Strcoe=Strcoe/L0;Permit=Permit/F0;

    CComplex CCj(0,1);
    CplxY=Ep*(1+CCj*Yfta);
    CplxPermit=Permit*(1-CCj*DieLoss);
}

void CBeam::setBoundary()
{
    switch (BondCond)
    {
        case 1://Free-Free
            faiw.putNterm(1);
            faiw.putCoef(0,1.);faiw.putXpower(0,0);
            break;
        case 2://Simply supported
            faiw.putNterm(2);
            faiw.putCoef(0,1.);faiw.putXpower(0,2);
            faiw.putCoef(1,-1.0);faiw.putXpower(1,0);
            break;
        default:
            cout<<"Error in BondCond !"<<endl<<"BondCond == *"
            <<BondCond<<endl;

            cout<<"program exit -1 !"<<endl;
            exit (-1);
    }

    //Begin to calculate the poly matrix for sbiff and mass matrix
    CPolyx*fai=new CPolyx [mnl];
    if (fai == NULL)
    {
        cout << "Fail to allocate memory for fai[] !"<<endl;
        exit(-1);
    }
    for (int i=0;i<mnl;i++)//fai[i]=x^i*faiw
    {
        fai[i]=faiw;
        for (int j=0;j<faiw.getNterm();j++)
            fai[i].putXpower(j,i+faiw.getXpower(j));
    }
    CPolyx*fai2=new CPolyx [mnl];
    if (fai2 == NULL)
    {
        tout << "Fail to allocate memory for fai2[] !"<<endl;
        exit(-1);
    }
    for (i=0;i<mnl;i++)
    {
        fai2[i]=dx(dx(fai[i]));
    }

    for (int m=0;m<mnl;m++)
        for (int l=m;l<mnl;l++) //Only the upper triangular
        {
            CPK[Allocate(m,l,mnl)]=fai2[m]*fai2[l];
            CPM[Allocate(m,l,mnl)]=fai[m]*fai[l];
        }

    if (fai) delete[] fai;
    if (fai2) delete[] fai2;
}

void CBeam::stiffmass()//to generate KO and MO
{
    //Begin to calculate the stiffness and mass matrix.
    //*****
    for (int m=0;m<mnl;m++)
        for (int l=m;l<mnl;l++)
        {
            K0[Allocate(m,l,mnl)]=2*E*H*H/(3*A*A*A)
                CPK[Allocate(m,l,mnl)].inte(1,0,-1.0);
            M0[Allocate(m,l,mnl)]=Rou*H*A/2
                (CPM[Allocate(m,l,mnl)]).inte(1,0,-1.0);
        }

    for (m=0;m<mnl;m++)
        for (int l=0;l<m;l++)
        {
            K0[Allocate(m,l,mnl)]=K0[Allocate(l,m,mnl)];
            M0[Allocate(m,l,mnl)]=M0[Allocate(l,m,mnl)];
        }
    //Finished calculating the stiffness and mass matrix.
}

void CBeam::stiffmass(CChromosome* cc)//to generate K and M,
{
    double Ed=cc->getvariable(0);
    double Cd=cc->getvariable(1);
    double Ld=cc->getvariable(2);

    double ksi=2*Cd/A-1;
    //Begin to calculate the stiffness and mass matrix.
    //*****
    for (int m=0;m<mnl;m++)
        for (int l=m;l<mnl;l++)
        {
            K[Allocate(m,l,mnl)]=K0[Allocate(m,l,mnl)]
                -2*(E-Ed)*H*H/(3*A*A*A)
                *CPK[Allocate(m,l,mnl)].getValue(ksi)
                -2*Ld/A;
            M[Allocate(m,l,mnl)]=M0[Allocate(m,l,mnl)];
        }
    for (m=0;m<mnl;m++)
        for (int l=0;l<m;l++)
        {
            K[Allocate(m,l,mnl)]=K[Allocate(l,m,mnl)];
            M[Allocate(m,l,mnl)]=M[Allocate(l,m,mnl)];
        }
    //finished calculating the sbffness and mass matrix.
}

void CBeam::Relang_each_mode()
{
    double ksi1,ksi2;
    ksi1=2*Lxp/A-1;ksi2=2*Uxp/A-1;
    for (int k=0;k<NNterm;k++)//for each mode
    {
        Betax[k]=0.0;
        for (inti=0;i<mnl;i++)
        {
            CPolyx fai=faiw;
            for (intij=0;j<fai.getNterm();ij++)
                fai.putXpower(ij,faiw.getXpower(ij)+i); //fai=faiw*x^i=fai[i]
            CPolyx faidx=dx(fai);

            double t1=fai2dx.getValue(ksi1)-fai2dx.getValue(ksi2);
            Betax[k]+=Evec[Allocate(k,l,mnl)]*t1;

            ]*2/A;
        }
    }

void CBeam::factor()
{
    double *t1,*t2;
    t1=new double [NNterm];
    if (t1 == NULL)

```

```

{
    cout << "Fail to allocate memory for t1[]" << endl;
    exit(-1);
}
t2=new double[NNterm];
if (t2 == NULL)
{
    cout << "Fail to allocate memory for t2[]" << endl;
    exit(-1);
}
//t1 and t2 are temporary variables
double ksi1,ksi2;
ksi1=2*Lxp/A-1;ksi2=2*Uxp/A-1;

for (intk=NNterm-1;k>=0;k--)
{
    t1[k]=0.0;t2[k]=0.0;
    for (inti=0;i<mn1;i++)
    {
        for (intj=0;j<mn1;j++)
        {
            t1[k]=Evec[Allocate(k,i,mn1)]*M0[Allocate(i,j,mn1)]
                *Evec[Allocate(k,j,mn1)];
        }
    }
    //t1=XT*M*X

//Mx=M*(delt(x-x1)-delt(x-x2));
//t2=inte(X*(-dx(Mx)))
for (i=0;i<mn1;i++)
{
    CPolyxfai=faiw;
    for (int ij=0;j<fai.getNterm();ij++)
        fai.putXpower(ij,faiw.getXpower(ij)+1);
    //fai=faiw*x^i=fai[i]
    CPolyxfaidx=dx(fai);

    double d=faidx.getValue(ksi2)-faidx.getValue(ksi1);
    t2[k]=Evec[Allocate(k,i,mn1)]*d;
}
t2[k]=t2[k]*2/A;//t2=inte(Faix*Fx)
//+++++
Vyitax[k]=t2[k]/t1[k];
}
delete [] t1;
delete [] t2;
}

double CBeam::Conduct(double Omega)
{
    // CComplex CCj(0,1);
    Z=0;
    for (inti=0;i<NNterm;i++)
        Z=Z+Vyitax[i]/(Eval[i]*(1+CCj*Damping)-Omega*Omega)*Betax[i];

    //Inverse Z
    Z=1/Z;

    //Force impedance from moment impedance
    Z=2*B/((H+Hp)*(H+Hp)*CCj*Omega)*Z;

    Z=Z*2;
    CComplex CplxK=csqrt(Roup/CplxY)*Omega;
    CComplex CplxA;
    CplxA=CplxK*Z*CCj*Omega/(B*Hp*CplxY)*tan(CplxK*Ap/2);
    CplxA=1/CplxA;

    CComplex Admit=CplxY*CCj*Omega*B/Hp
        *(Stroce*Stroce*(2*CplxA*tan(CplxK*Ap/2)
        -Ap)+Permit*Ap/CplxY);

    return Admit.real();
}

void CBeam::IOput(char* Infile,ofstream& StrOutfile)
{
    ifstream StrInfile(Infile);
    if (StrInfile.fail())
    {
        cout<<"Error in CBeam: Unable to open input.dat!"
            <<endl<<endl;
        cin>>a;
        exit (-1);
    }
    StrInfile>>SNp>>SBondCond;
    StrInfile>>Np>>BondCond;
    StrOutfile<<setw(10)<<SNp<<setw(10)<<SBondCond<<endl;
    StrOutfile<<setw(10)<<Np<<setw(10)<<BondCond<<endl;
    Initialize();

    StrInfile>>SE>>SRou>>SHp>>SA>>SB>>SDamping;
    StrInfile>>E>>Rou>>Niu>>H>>A>>B>>Damping;
    StrOutfile<<setw(10)<<SE<<setw(10)<<SRou<<setw(10)<<SHp
        <<setw(10)<<SA<<setw(10)<<SB
        <<setw(10)<<SDamping<<endl;
    StrOutfile<<setw(10)<<E<<setw(10)<<Rou<<setw(10)<<Niu
        <<setw(10)<<H<<setw(10)<<A<<setw(10)<<B
        <<setw(10)<<Damping<<endl;

    StrInfile>>SUxp>>SLxp;
    StrInfile>>Uxp>>Lxp;
    StrOutfile<<setw(10)<<SUxp<<setw(10)<<SLxp<<endl;
    StrOutfile<<setw(10)<<Uxp<<setw(10)<<Lxp<<endl;

    StrInfile>>SEp>>SYita>>SRoup>>SNIup>>SHp
        >>SSStroce>>SPermit>>SDieLoss;
    StrInfile>>Ep>>Yita>>Roup>>NIup>>Hp
        >>Stroce>>Permit>>DieLoss;
    StrOutfile<<setw(10)<<SEp<<setw(10)<<SYita<<setw(10)<<SRoup
        <<setw(10)<<SNIup<<setw(10)<<SHp
        <<setw(10)<<SSStroce<<setw(10)<<SPermit<<setw(10)
        <<SDieLoss<<endl;
    StrOutfile<<setw(10)<<Ep<<setw(10)<<Yita<<setw(10)<<Roup
        <<setw(10)<<NIup<<setw(10)<<Hp<<setw(10)<<Stroce
        <<setw(10)<<Permit<<setw(10)<<DieLoss<<endl;

    normalize();
    setBoundary();

    StrInfile>>SUF>>SLF;
    StrInfile>>UF>>LF;
    StrOutfile<<setw(10)<<SUF<<setw(10)<<SLF<<endl;
    StrOutfile<<setw(10)<<UF<<setw(10)<<LF<<endl;

    StrInfile>>SMUPeak>>SMDPeak>>SPUPeak;
    StrOutfile<<setw(20)<<SMUPeak<<setw(20)<<SMDPeak
        <<setw(20)<<SPUPeak<<setw(20)<<"ReferPeak"<<endl;

    inti=0;
    do
    {
        StrInfile>>MUPeak[i]>>MDPeak[i]>>PUPeak[i];
        Refer_Peak[i]=PUPeak[i]-MUPeak[i]+MDPeak[i];
        i++;
    }
    while (MUPeak[i-1]>0);
    InPeakNum=i-1;
    for (j=0;j<InPeakNum;j++)
        StrOutfile<<setw(20)<<MUPeak[j]<<setw(20)<<MDPeak[j]
            <<setw(20)<<PUPeak[j]<<setw(20)<<Refer_Peak[j]<<endl;
    }

    //===== Plate cpp =====
    #include "Plate.h"

    CPlate::CPlate()
    {
        Betax=new double[ "tem"];
        if (Betax==NULL)
        {
            cout<<"Fail to allocate memory for Betax[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        Betay=new double[ "tem"];
        if (Betay==NULL)
        {
            cout<<"Fail to allocate memory for Betay[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        Vyitax=new double[NNterm];
        if (Vyitax==NULL)
        {
            cout<<"Fail to allocate memory for Vyitax[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        Vyitay=new double[NNterm];
        if (Vyitay==NULL)
        {
            cout<<"Fail to allocate memory for Vyitay[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        CPK1=new CPoly[mn1*mn1];
        if (CPK1==NULL)
        {
            cout<<"Fail to allocate memory for CPK1[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        CPK2=new CPoly[mn1*mn1];
        if (CPK2==NULL)
        {
            cout<<"Fail to allocate memory for CPK2[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
        CPM=new CPoly[mn1*mn1];
        if (CPM==NULL)
        {
            cout<<"Fail to allocate memory for CPM[]" <<endl;
            cout<<"Program exit -1!"<<endl;
            exit (-1);
        }
    }
}

```



```

}

CPlate::~CPlate()
{
    if (Betax) delete[] Betax;
    if (Betay) delete[] Betay;
    if (Vyitax) delete[] Vyitax;
    if (Vyitay) delete[] Vyitay;
    if (CPK1) delete[] CPK1;
    if (CPK2) delete[] CPK2;
    if (CPM) delete[] CPM;
}

void CPlate::Initialize()//Reallocate memory according to Np;
{
    mnl=(Np+1)*(Np+2)/2; NNterm=mnl;
    if (KO) delete[] KO;
    KO=new double[mnl*mnl];
    if (KO==NULL)
    {
        cout<<"Fail to allocate memory for KO!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (M0) delete[] M0;
    M0=new double [mnl*mnl];
    if (M0==NULL)
    {
        cout<<"Fail to allocate memory for M0!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (K) delete[] K;
    K=new double [mnl*mnl];
    if (K==NULL)
    {
        cout<<"Fail to allocate memory for K!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (M) delete[] M;
    M=new double [mnl*mnl];
    if (M==NULL)
    {
        cout<<"Fail to allocate memory for M!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Eval) delete[] Eval;
    Eval=new double [mnl];
    if (Eval==NULL)
    {
        cout<<"Fail to allocate memory for Eval!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Evec) delete[] Evec;
    Evec=new double [mnl*mnl];
    if (Evec==NULL)
    {
        cout<<"Fail to allocate memory for Evec!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Betax) delete[] Betax;
    Betax=new double [ **term];
    if (Betax==NULL)
    {
        cout<<"Fail to allocate memory for Betax!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Betay) delete[] Betay;
    Betay=new double [NNterm];
    if (Betay==NULL)
    {
        cout<<"Fail to allocate memory for Betax!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Vyitax) delete[] Vyitax;
    Vyitax=new double [NNterm];
    if (Vyitax==NULL)
    {
        cout<<"Fail to allocate memory for Vyitax!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (Vyitay) delete[] Vyitay;
    Vyitay=new double [NNterm];
    if (Vyitay==NULL)
    {
        cout<<"Fail to allocate memory for Vyitay!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (CPK1) delete[] CPK1;
    CPK1=new CPoly [mnl*mnl];
    if (CPK1==NULL)
    {
        cout<<"Fail to allocate memory for CPK1!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (CPK2) delete[] CPK2;
    CPK2=new CPoly [mnl*mnl];
    if (CPK2==NULL)
    {
        cout<<"Fail to allocate memory for CPK2!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
    if (CPM) delete[] CPM;
    CPM=new CPoly [mnl*mnl];
    if (CPM==NULL)
    {
        cout<<"Fail to allocate memory for CPM!"<<endl;
        cout<<"Program exit -1!"<<endl;
        exit (-1);
    }
}

void CPlate::normalize()
{
    E0=E;Rou0=Rou;L0=B;
    E=E/E0;Rou=Rou/Rou0;H=H/L0;A=A/L0;B=B/L0;
    Omega0=sqrt(E0/Rou0)/L0;Z0=sqrt(E0*Rou0)*L0*L0;
    F0=E0*L0*L0;//non-dimensionalize

    Uxp=Uxp/L0;Lxp=Lxp/L0;Uyp=Uyp/L0;Lyp=Lyp/L0;
    Ap=Uxp-Lxp;Bp=Uyp-Lyp;

    Ep=Ep/E0;Roup=Roup/Rou0;Hp=Hp/L0;
    Strcoe=Strcoe/L0;Permit=Permit/F0;

    CplxY=Ep*(1+CCj)*Yita);
    CplxPermit=Permit*(1-CCj)*DieLoss);
}

void CPlate::setBoundary()
{
    switch (BondCond)
    {
        case 1://Free-Free
            faiw.putNterm(1);
            faiw.putCoef(0,1);faiw.putXpower(0,0);faiw.putYpower(0,0);
            break;
        case 2://Simply supported
            faiw.putNterm(4);
            faiw.putCoef(0,1);faiw.putXpower(0,2);faiw.putYpower(0,2);
            faiw.putCoef(1,-1,0);faiw.putXpower(1,2);faiw.putYpower(1,0);
            faiw.putCoef(2,-1,0);faiw.putXpower(2,0);faiw.putYpower(2,2);
            faiw.putCoef(3,1,0);faiw.putXpower(3,0);faiw.putYpower(3,0);
            break;
        default:
            cout<<"Error in BondCond!"<<endl<<"BondCond == "
            <<BondCond<<endl;
            cout<<"program exit -1!"<<endl;
            exit (-1);
    }

    //Begin to calculate the poly matrix for stiff and mass matrix
    CPoly *fai=new CPoly [mnl];
    if (fai ==NULL)
    {
        cout<<"Fail to allocate memory for fai!"<<endl;
        exit(-1);
    }
    for (int i=0;i<=Np;i++)
        for (int j=0;j<=Np;j++)
        {
            int m=(i+1)/2+i+1;
            fai[m].putNterm(1); fai[m].putCoef(0,1,0);
            fai[m].putXpower(0,i-1);fai[m].putYpower(0,i);
            fai[m]=fai[m]*faiw;
        }
    CPoly *fai2=new CPoly [mnl];
    if (fai2 == NULL)
    {
        cout<<"Fail to allocate memory for fai2!"<<endl;
        exit(-1);
    }
    CPoly *fai2y=new CPoly [mnl];
    if (fai2y ==NULL)
    {
        cout<<"Fail to allocate memory for fai2y!"<<endl;
        exit(-1);
    }
    CPoly *fai2xdy=new CPoly [mnl];
    if (fai2xdy == NULL)
    {
        cout<<"Fail to allocate memory for fai2xdy!"<<endl;
        exit(-1);
    }
}

```

```

}
for (i=0;i<mn1;i++)
{
    faidx2[i]=dx(dx(fai[i]));
    faidx2m=dx(dx(fai[m]));
    faidxdy[i]=dx(dy(fai[i]));
}

for (int m=0;m<mn1;m++)
for (int l=m;l<mn1;l++) //Only the upper triangular.
{
    CPK1[Allocate(m,l,mn1)]=(1./(A*A)*faidx2[m]
    +1./(B*B)*faidx2[l])*(1./(A*A)*faidx2[l]+1./(B*B)*faidx2[l]);
    CPK2[Allocate(m,l,mn1)]=faidx2[m]*faidx2[l]+faidx2[m]*faidx2[l];
    CPM[Allocate(m,l,mn1)]=fai[m]*fai[l];
}

if (fai) delete [] fai;
if (faidx2) delete [] faidx2;
if (faidxdy) delete [] faidxdy;
if (faidxdy) delete [] faidxdy;
}

void CPlate::stiffmass()//to generate KO and MO
{
    double D=E*H*H*(12.0*(1-Niu*Niu));
    //Begin to calculate the sliffness and mass matrix.
    for (int m=0;m<mn1;m++)
    for (int l=m;l<mn1;l++)
    {
        K0[Allocate(m,l,mn1)]=4*D*A*B
        *CPK1[Allocate(m,l,mn1)].intexy(1.0,-1.0,1.0,-1.0)
        -4*D*(1-Niu)/(A*B)
        *CPK2[Allocate(m,l,mn1)].intexy(1.0,-1.0,1.0,-1.0);
        // for simply supported, the second part can be omitted.
        M0[Allocate(m,l,mn1)]=A*B*Rou*H/4
        *(CPM[Allocate(m,l,mn1)]).intexy(1.0,-1.0,1.0,-1.0);

        for (m=0;m<mn1;m++)
        for (int l=0;l<m;l++)
        {
            K0[Allocate(m,l,mn1)]=K0[Allocate(l,m,mn1)];
            M0[Allocate(m,l,mn1)]=M0[Allocate(l,m,mn1)];
        }
    }
    //Finished calculating the sliffness and mass matrix.
}

void CPlate::stiffmass(CChromosome* cc)//to generate K and M,
{
    double D,Dd;
    D=E*H*H*(12.0*(1-Niu*Niu));
    Dd=0;
    double Cxd=cc->getvariable(0);
    double Cyd=cc->getvariable(1);
    double Rad=cc->getvariable(2);
    double ksi,yita;
    ksi=2*Cxd/A-1;yita=2*Cyd/B-1;

    //Begin to calculate the stiffness and mass matrix.
    for (int m=0;m<mn1;m++)
    for (int l=m;l<mn1;l++)
    {
        K[Allocate(m,l,mn1)]=K0[Allocate(m,l,mn1)]
        -16*(D-Dd)*CPK1[Allocate(m,l,mn1)].intecircle(ksi,yita,Rad,A,B);
        *CPK2[Allocate(m,l,mn1)].intecircle(ksi,yita,Rad,A,B);
        M[Allocate(m,l,mn1)]=M0[Allocate(m,l,mn1)];
    }

    for (m=0;m<mn1;m++)
    for (int l=0;l<m;l++)
    {
        K[Allocate(m,l,mn1)]=K[Allocate(l,m,mn1)];
        M[Allocate(m,l,mn1)]=M[Allocate(l,m,mn1)];
    }
    //Finished calculating the sliffness and mass matrix.
}

void CPlate::Relang_each_mode()
{
    double ksi1,ksi2,yita1,yita2;
    ksi1=2*Lxp/A-1;ksi2=2*Uxp/A-1;yita1=2*Lyp/B-1;yita2=2*Uyp/B-1;
    double ksi0=(ksi1+ksi2)/2;double yita0=(yita1+yita2)/2;
    for (int k=0;k<NNterm;k++)//for each mode.
    {
        Betax[k]=0.0;Betay[k]=0.0;
        for (int i=0;i<Np;i++)
        {
            for (int j=0;j<=i;j++)
            {
                int m=i*(i+1)/2+j;
                double d1=0.0;double t2=0.0;
                for (int ij=0;ij<faiw.getNterm();ij++)
                {
                    double a=faiw.getCoeff(ij);
                    int b=faiw.getXpower(ij)+j;
                    int c=faiw.getYpower(ij)+j;
                    if (b!=0) t1+=a*b*(pow(ksi1,b-1)
                    *pow(ksi2,b-1))*pow(yita0,c);
                    if (c!=0) t2+=a*c*(pow(yita1,c-1)
                    *pow(yita2,c-1))*pow(ksi0,b);
                }
                Betax[k]=Evect[Allocate(k,m,mn1)]*t1;
                Betay[k]=Evect[Allocate(k,m,mn1)]*t2;
            }
            Betax[k]=Betax[k]*2/A;
            Betay[k]=Betay[k]*2/B;
        }
    }

    void CPlate::factor()
    {
        double *t1,*t2,*t3;
        t1=new double[NNterm];
        if (t1 == NULL)
        {
            cout << "Fail to allocate memory for t1[]"!<<endl;
            exit(-1);
        }
        t2=new double[NNterm];
        if (t2 == NULL)
        {
            cout << "Fail to allocate memory for t2[]"!<<endl;
            exit(-1);
        }
        t3=new double[NNterm];
        if (t3 == NULL)
        {
            cout << "Fail to allocate memory for t3[]"!<<endl;
            exit(-1);
        }
        //t1 t2 and t3 are temporary variables
        for (int i=0;i<NNterm;i++){t1[i]=0.0;t2[i]=0.0;t3[i]=0.0;}
        double ksi1,ksi2,yita1,yita2;
        ksi1=2*Lxp/A-1;ksi2=2*Uxp/A-1;yita1=2*Lyp/B-1;yita2=2*Uyp/B-1;
        double ksi0=(ksi1+ksi2)/2;double yita0=(yita1+yita2)/2;//point of the
        PZT center

        for (int k=0;k<NNterm;k++)
        {
            for (int i=0;i<mn1;i++)
            {
                for (int j=0;j<mn1;j++)
                {
                    t1[k]=Evect[Allocate(k,i,mn1)]*M[Allocate(i,j,mn1)]
                    *Evect[Allocate(k,j,mn1)];
                }
            }
            //t1=XT*M*X
            //Mx=M*(delt(x-x1)-delt(x-x2))*H(y-y1)-H(y-y2));
            //t2=intxy(X*(-dx(Mx)))
            for (i=0;i<Np;i++)
            {
                for (int j=0;j<=i;j++)
                {
                    int m=i*(i+1)/2+j;
                    double d1=0.0;double d2=0.0;
                    for (int ij=0;ij<faiw.getNterm();ij++)
                    {
                        double a,b,c;
                        a=faiw.getCoeff(ij);
                        b=faiw.getXpower(ij)+j;
                        c=faiw.getYpower(ij)+j;
                        d1+=a*(pow(ksi1,b-1)-pow(ksi1,b-1))
                        *(pow(yita2,c+1)-pow(yita1,c+1))/(c+1);
                        d2+=a*(pow(ksi2,b+1)-pow(ksi1,b+1))/(b+1)
                        *c*(pow(yita2,c-1)-pow(yita1,c-1));
                    }
                    //d=inte(faix[m]*Fx);
                    t2[k]=Evect[Allocate(k,m,mn1)]*d1;
                    t3[k]=Evect[Allocate(k,m,mn1)]*d2;
                }
            }
            t2[k]=t2[k]*B/2*A;t2=inte(Faix*Fx)
            t3[k]=t3[k]*A/2*B;
            //+++++
            Vyitax[k]=t2[k]/t1[k];
            Vyitay[k]=t3[k]/t1[k];
        }
        if (t1) delete [] t1;
        if (t2) delete [] t2;
        if (t3) delete [] t3;

        double CPlate::Conduct(double Omega)
        {
            for (int i=0;i<4;i++) Z[i]=0;
            for (i=0;i<NNterm;i++)

```



```

{
    Z[0]=Z[0]+Vyitax[i]/(Eval[i]/(1+CCj)*Damping)
        -Omega*Omega)*Betax[i];
    Z[1]=Z[1]+Vyitay[i]/(Eval[i]/(1+CCj)*Damping)
        -Omega*Omega)*Betay[i];
    Z[2]=Z[2]+Vyitax[i]/(Eval[i]/(1+CCj)*Damping)
        -Omega*Omega)*Betax[i];
    Z[3]=Z[3]+Vyitay[i]/(Eval[i]/(1+CCj)*Damping)
        -Omega*Omega)*Betay[i];
}

//Inverse Z
CComplex a,b,c,d,e;
a=Z[0];b=Z[1];c=Z[2];d=Z[3];e=a*d-b*c;
Z[0]=d/e;Z[1]=-b/e;Z[2]=-c/e;Z[3]=a/e;

//Force impedance from moment impedance
Z[0]=2*(Uyp-Lyp)/((H+Hp)*(H+Hp)*CCj*Omega)*Z[0];
Z[1]=2*(Uyp-Lyp)/((H+Hp)*(H+Hp)*CCj*Omega)*Z[1];
Z[2]=2*(Uxp-Lxp)/((H+Hp)*(H+Hp)*CCj*Omega)*Z[2];
Z[3]=2*(Uxp-Lxp)/((H+Hp)*(H+Hp)*CCj*Omega)*Z[3];

for (i=0;i<4;i++) Z[i]=Z[i]*2;
CComplex CplxK=csqrt(Roup*(1-Niup*Niup/CplxY)*Omega
CComplex CplxX,CplxY;
CComplex CplxK1=CCj*Omega/(CplxY*CplxK*Hp);
CComplex N[4];
N[0]=1+CplxK1*(Z[0]/Bp-Niup*Z[2]/Ap)*tan(CplxK*Ap/2);
N[1]=CplxK1*(Z[1]/Bp-Niup*Z[3]/Ap)*tan(CplxK*Ap/2);
N[2]=CplxK1*(Niup*Z[0]/Bp+Z[2]/Ap)*tan(CplxK*Ap/2);
N[3]=1+CplxK1*(Niup*Z[1]/Bp+Z[3]/Ap)*tan(CplxK*Ap/2);

a=N[0];b=N[1];c=N[2];d=N[3];e=a*d-b*c;
N[0]=d/e;N[1]=-b/e;N[2]=-c/e;N[3]=a/e;//inv(N);
CplxX=(N[0]+N[1])/CplxK;CplxY=(N[2]+N[3])/CplxK;

CComplex Admit=CplxY*CCj*Omega/((1-Niup*Niup)*Hp)
*(2*(1+Niup)*Strcoe*Strcoe*(Bp*CplxX*tan(CplxK*Ap/2)
+Ap*CplxC*tan(CplxK*Bp/2)-Ap*Bp))
+Ap*Bp*Permit*CCj*Omega/Hp;

return Admit.real();
}

void CPlate::IOutput(char* Infile,ofstream& StrOutfile)
{
    ifstream StrInfile(Infile);
    if (StrInfile.fail())
    {
        cout<<"Error in CPlate: Unable to open input.dat!"<<endl<<endl;
        char a;
        cin>>a;
        exit (-1);
    }
    StrInfile>>SNp>>SBondCond;
    StrOutfile<<setw(10)<<SNp<<setw(10)<<SBondCond<<endl;
    StrInfile>>Np>>BondCond;
    StrOutfile<<setw(20)<<Np<<setw(20)<<BondCond<<endl;
    Initialize();

    StrInfile>>SE>>SRou>>SNiu>>SH>>SA>>SB>>SDamping;
    StrInfile>>E>>Rou>>Niu>>H>>A>>B>>Damping;
    StrOutfile<<setw(10)<<SE<<setw(10)<<SRou<<setw(10)<<SNiu
        <<setw(10)<<SH<<setw(10)<<SA<<setw(10)<<SB<<setw(10)
        <<SDamping<<endl;
    StrOutfile<<setw(10)<<E<<setw(10)<<Rou<<setw(10)<<Niu
        <<setw(10)<<H<<setw(10)<<A<<setw(10)<<B<<setw(10)
        <<Damping<<endl;

    StrInfile>>SUxp>>SLxp>>SUyp>>SLyp;
    StrInfile>>Uxp>>Lxp>>Uyp>>Lyp;
    StrOutfile<<setw(10)<<SUxp<<setw(10)<<SLxp<<setw(10)<<SUyp
        <<setw(10)<<SLyp<<endl;
    StrOutfile<<setw(10)<<Uxp<<setw(10)<<Lxp<<setw(10)<<Uyp
        <<setw(10)<<Lyp<<endl;

    StrInfile>>SEp>>SYita>>SRoup>>SNiup
        >>SHp>>SStrcoe>>SPermit>>SDieLoss;
    StrInfile>>Ep>>Yita>>Roup>>Niup
        >>Hp>>Strcoe>>Permit>>DieLoss;
    StrOutfile<<setw(10)<<SEp<<setw(10)<<SYita<<setw(10)<<SRoup
        <<setw(10)<<SNiup<<setw(10)<<SHp<<setw(10)<<SStrcoe
        <<setw(10)<<SPermit<<setw(10)<<SDieLoss<<endl;
    StrOutfile<<setw(10)<<Ep<<setw(10)<<Yita<<setw(10)<<Roup
        <<setw(10)<<Niup<<setw(10)<<Hp<<setw(10)<<Strcoe
        <<setw(10)<<Permit<<setw(10)<<DieLoss<<endl;

    normalize();
    setBoundary();

    StrInfile>>SUF>>SLF;
    StrInfile>>UF>>LF;
    StrOutfile<<setw(10)<<SUF<<setw(10)<<SLF<<endl;
    StrOutfile<<setw(10)<<UF<<setw(10)<<LF<<endl;

    StrInfile>>SMUPeak>>SMDPeak>>SPUPeak;
    StrOutfile<<setw(20)<<SMUPeak<<setw(20)<<SMDPeak
        <<setw(20)<<SPUPeak<<setw(20)<<"ReferPeak"<<endl;
    inti=0;
    do
    {
        StrInfile>>MUPeak[i]>>MDPeak[i]>>PUPeak[i];
        Refer_Peak[i]=PUPeak[i]-MUPeak[i]+MDPeak[i];
        i++;
    }
    while (MUPeak[i-1]>0);
    InPeakNum=i-1;
    for (i=0; i<InPeakNum; i++)
        StrOutfile<<setw(20)<<MUPeak[i]<<setw(20)<<MDPeak[i]
            <<setw(20)<<PUPeak[i]<<setw(20)<<Refer_Peak[i]<<endl;
    }
    //===== Evaluation.cpp =====
    #include "Evaluation.h"
    #include "Random.h"

    CEvaluation::CEvaluation()
    {
        NPop_size=1;
        fitness= new double [2*NPop_size];
        if (fitness == NULL)
        {
            cout << "Fail to allocate memory for fitness[]" <<endl;
            exit(-1);
        }
        weight = new double [2*NPop_size];
        if (fitness == NULL)
        {
            cout << "Fail to allocate memory for weight[]" <<endl;
            exit(-1);
        }
    }

    CEvaluation::CEvaluation(int npop_size)
    {
        NPop_size=npop_size;
        fitness = new double [2*NPop_size];
        if (fitness == NULL)
        {
            cout << "Fail to allocate memory for fitness[]" <<endl;
            exit(-1);
        }
        weight = new double [2*NPop_size];
        if (fitness == NULL)
        {
            cout << "Fail to allocate memory for weight[]" <<endl;
            exit(-1);
        }
    }

    CEvaluation::CEvaluation(CParameter& cp)
    {
        if (NPop_size!=cp.getPop_size())
        {
            delete [] fitness;
            delete [] weight;
            NPop_size=cp.getPop_size();
        }
        fitness = new double [2*NPop_size];
        if (fitness == NULL)
        {
            cout << "Fail to allocate memory for fitness[]" <<endl;
            exit(-1);
        }
        weight = new double [2*NPop_size];
        if (weight == NULL)
        {
            cout << "Fail to allocate memory for weight[]" <<endl;
            exit(-1);
        }
    }

    CEvaluation::~CEvaluation()
    {
        if (NPop_size!=0)
        {
            delete [] fitness;
            delete [] weight;
        }
    }

    void CEvaluation::cal_fitness(int Iflag,CChromosome** cc,CStruct*
        cstru)
    {
        i ti;

        if (Iflag == 1)
        {
            //Compute the objectives fitness[i] of Chromosome[i]
            for (i=0;i<NPop_size;i++)
                fitness[i]=cstru->fitness(cc[i]);
        }
        else if (Iflag == 2)
        {

```

```

//Compute the objectives fitness[j+NPop_size] of
//Chromosome[j+NPop_size].
for (i=0; i<NPop_size; i++)
    fitness[j+NPop_size]=ccstru->fitness(cc[j+NPop_size]);
}
else cout<<"Error in lflag"<<endl;
return;
}

void CEvaluation::put_fitness(int i, double fit)
{
    try
    {
        if ((i<2*NPop_size)&&(i>=0))
            fitness[i]=fit;
        else throw "Error in CEvaluation::put_fitness!";
    }
    catch (char *e)
    {
        cout<<e<<endl;
        cout<<"i="<<i<<endl;
        exit(-1);
    }
}

void CEvaluation::cal_weight(int num_gen, CChromosome**
                             cc, CParameter& cp)
{
    int i, j, k, jj;
    double T0, dweight, beta=0.972582, betaj;

    CRandomCr;

    //Set initial temperature T0=median of the fitness values.
    T0=fabs((get_fitness(0)-get_fitness(2*NPop_size-1))/2.0);
    //Compute weight by comparison.
    for (i=0; i<2*NPop_size; i++)
    {
        weight[i]=0.0;

        j=0;
        do
        {
            jj=int(cr.random_dbl(0.0,1.0)*(2*NPop_size-1));
            if (jj!=i)
            {
                if (((cp.getMaxMinID()==1)
                    &&(get_fitness(i)>=get_fitness(jj))
                    ||(cp.getMaxMinID()==2)
                    &&(get_fitness(i)<=get_fitness(jj))))
                {
                    //cc[i] is better than or equal to cc[jj]
                    weight[i] += 1.0;
                }
                else
                {
                    //Compute annealing value
                    dweight=fabs(get_fitness(i)-get_fitness(jj));
                    betaj=beta;
                    for (k=2; k<=num_gen; k++)
                        betaj*=beta;
                    weight[i] += exp(-(dweight/(T0*betaj)));
                }
            }
        }
        while (j<cp.getNTourna_size());
        j++;
    }
}

void CEvaluation::dsort(int lflag, char type, CChromosome**
                        cc, CParameter& cp)
{
    int N;
    if (lflag==1) N=NPop_size;
    else if (lflag==2) N=2*NPop_size;
    else
    {
        cout<<"Error in lflag"<<endl;
        exit(-1);
    }

    if (type=='f' || type=='F')
    {
        if (cp.getMaxMinID()==1)
        {
            for (int j=1; j<N; j++)
            {
                double temp=fitness[j];
                int i=j-1;
                while (i>=0&&fitness[i]<temp)
                {
                    CChromosome* tempcc;
                    tempcc=cc[i]; cc[i]=cc[i+1]; cc[i+1]=tempcc;
                    fitness[i+1]=fitness[i]; fitness[i]=temp;
                }
                i--;
            }
        }
        else
        {
            for (int j=1; j<N; j++)
            {
                double temp=fitness[j];
                int i=j-1;
                while (i>=0&&fitness[i]>temp)
                {
                    CChromosome* tempcc;
                    tempcc=cc[i]; cc[i]=cc[i+1]; cc[i+1]=tempcc;
                    fitness[i+1]=fitness[i]; fitness[i]=temp;
                }
                i--;
            }
        }
    }

    // After sort, best=fitness[0]; worst=fitness[2*NPop_size-1];

    //sort weight.
    else if (type=='w' || type=='W')
    {
        for (int j=1; j<N; j++)
        {
            double temp=weight[j];
            double tempf=fitness[j];
            int i=j-1;
            while (i>=0&&weight[i]<temp)
            {
                CChromosome* tempcc;
                tempcc=cc[i+1]; cc[i+1]=cc[i]; cc[i]=tempcc;
                weight[i+1]=weight[i]; weight[i]=temp;
                fitness[i+1]=fitness[i]; fitness[i]=tempf;
            }
            i--;
        }
    }
}

double CEvaluation::get_fitness(int i)
{
    try
    {
        if (i<2*NPop_size) return fitness[i];
        else throw "Error in CEvaluation::get_fitness ";
    }
    catch (char *e)
    {
        cout<<e<<endl;
        cout<<" i="<<i<<endl;
        cout<<" i="<<setw(2)<<i<<">= NVariable"<<endl;
        cout<<" Program returns -1."<<endl;
        return -1.0;
    }
}

double CEvaluation::get_weight(int i)
{
    try
    {
        if (i<2*NPop_size) return weight[i];
        else throw "Error in CEvaluation::get_weight ";
    }
    catch (char *e)
    {
        cout<<e<<endl;
        cout<<" i="<<setw(2)<<i<<">= NVariable"<<endl;
        cout<<" Program returns -1."<<endl;
        return -1.0;
    }
}

//End of Class CEvaluation
//===== Main.cpp =====
#include "BuildinHeader.h"

#include "Evaluation.h"
#include "Random.h"
#include "Complex.h"
#include "Struct.h"
#include "Plate.h"
#include "Beam.h"

//Begin of Application dependent function.
bool checkinequality (CChromosome* cc, CParameter& cp)
{
    int nvariable=cp.getNVariable();
    for (j=0; j<nvariable; j++)
    {
        if (((cc->getvariable(j))<cp.getpara_sca1(j, 'L'))
            || ((cc->getvariable(j))>cp.getpara_sca1(j, 'U')))
            return false;
    }
    return true;
}

```

```

//Description : To generate a newcc by applying MM on cc0
void Macro_Mutation(CChromosome* cc0, CChromosome*
newcc, CParameter& cp)
{
    int nvariable=cc0->getNVariable();
    CRandom cr;
    double AM=1.0;
    double *randr=new double[nvariable];
    int j;

    //produce a random move d = randr[]
    for (j=0;j<nvariable;j++)
    {
        double scal;
        if (fabs(cp.getpara_scal(j,'L'))>fabs(cp.getpara_scal(j,'U')))
            scal=fabs(cp.getpara_scal(j,'L'));
        else scal=fabs(cp.getpara_scal(j,'U'));
        randr[j]=cr.random_dbl(-scal,scal);
    }

    do //Generate newcc with feasible values.
    {
        for (j=0;j<nvariable;j++)
            newcc->putvariable(j,cc0->getvariable(j)+AM*randr[j]);
        AM=cr.random_dbl(0,AM);
    }
    while (!checkinequality(newcc,cp));

    for (j=0;j<nvariable;j++)
        newcc->putdeviation(j,(cp.getpara_scal(j,'U')-cp.getpara_scal(j,'L')));
    delete []randr;
    return;
}

//Description : To generate the first generation
//by applying Global-search on a seed chromosome
int initialize(int pop_size, CChromosome** cc, CParameter& cp)
{
    cout<<" Initializing "<<endl;
    int nvariable=cp.getNVariable();
    CRandom cr;
    //loop to generate pop-size chromosomes.
    for (inti=0;i<pop_size;i++)
    {
        for (intj=0;j<nvariable;j++)
        {
            cc[i]->putvariable(j,cr.random_dbl(
                cp.getpara_scal(j,'L'),cp.getpara_scal(j,'U')));
            cc[i]->putdeviation(j,(cp.getpara_scal(j,'U')
                -cp.getpara_scal(j,'L')));
        }
    }
    return 0;
}

//Description : To generate newcc by applying LBS on oldcc
void Grad_search(CChromosome* oldcc, CChromosome* newcc,
CParameter& cp, CStruct* cstru)
{
    int nvariable=cp.getNVariable();
    int i,j;
    double step;
    CRandom cr;
    CChromosome* tempcc=new CChromosome(nvariable);
    CChromosome* gcc = new CChromosome(nvariable);

    //Initialize newcc=oldcc; tempcc=newcc; fit0=f(newcc);
    for (i=0;i<nvariable;i++)
    {
        newcc->putvariable(i,oldcc->getvariable(i));
        newcc->putdeviation(i,oldcc->getdeviation(i));
    }
    double fit0=cstru->fitness(newcc);

    for (i=0;i<nvariable;i++)//Loop for each variable
    {
        for (j=0;j<nvariable;j++)
            tempcc->putvariable(j,newcc->getvariable(j));
        double delt=1e-4;
        if (fabs(newcc->getvariable(i))>1e-10)
            delt=fabs(newcc->getvariable(i));
        for (j=0;j<nvariable;j++)
            gcc->putvariable(j,newcc->getvariable(j));
        do
        {
            gcc->putvariable(i,newcc->getvariable(i)+delt);
            delt*=cr.random_dbl(0,1);
        }
        while (!checkinequality(gcc,cp));
        double fit1=cstru->fitness(gcc);
        if (fit1!=fit0)
        {
            if (((cp.getMaxMinID()==1)&&(fit1>fit0))
                ||((cp.getMaxMinID()==2)&&(fit1<fit0)))
                //gcc better than
                newcc
                step=oldcc->getdeviation(i);
            if (((cp.getMaxMinID()==1)&&(fit1<fit0))
                ||((cp.getMaxMinID()==2)&&(fit1>fit0)))
                //gcc worse than newcc
                step=oldcc->getdeviation(i);
            double AM=1;
            do
            {
                step*=AM;
                tempcc->putvariable(i,newcc->getvariable(i)+step);
                AM*=cr.random_dbl(0,1);
            }
            while (!checkinequality(tempcc,cp));
            double tempfit=cstru->fitness(tempcc);
            while (((cp.getMaxMinID()==1)&&(tempfit<=fit0))
                ||((cp.getMaxMinID()==2)&&(tempfit>=fit0)))
                //tempcc worse than newcc
                &&(fabs(step)>=1e-5))
                step/=2;
            AM=1;
            do
            {
                step*=AM;
                tempcc->putvariable(i,newcc->getvariable(i)+step);
                AM*=cr.random_dbl(0,1);
            }
            while (!checkinequality(tempcc,cp));
            tempfit=cstru->fitness(tempcc);
        }
        //upgrad newcc for next loop
        newcc->putvariable(i,tempcc->getvariable(i));
        newcc->putdeviation(i,fabs(step));
        fit0=tempfit;
    }
    //End if (fit1!=fit0);
    //End for (i=0;i<nvariable;i++)

    delete tempcc;
    delete gcc;
    return;
}

//Description : To generate newcc by X from father and mother
void crossover(CChromosome* father, CChromosome* mother,
CChromosome* offspring, CParameter& cp)
{
    int nvariable=cp.getNVariable();
    CRandom cr;
    int i;
    double AM=1.0;
    double *r=new double[nvariable];
    for (i=0;i<nvariable;i++)
        r[i]=cr.random_dbl(0,1.0);
    do
    {
        for (i=0;i<nvariable;i++)
        {
            offspring->putvariable(i,father->getvariable(i)
                *(1-AM*r[i])+mother->getvariable(i)*AM*r[i]);
            offspring->putdeviation(i,father->getdeviation(i)
                *(1-AM*r[i])+mother->getdeviation(i)*AM*r[i]);
        }
        AM=cr.random_dbl(0,1.0)*AM;
    }
    while (!checkinequality(offspring,cp));
    delete [] r;
    return;
}

//=====
//End of Application dependent function.
//=====

int main()
{
    CRandom cr;

    int i,j,pop_size,num_gen,max_gen,nvariable;
    int MaxMinID;//=1:Max, =2:Min
    double Pgrad,Pxover;
    unsigned randomseed=(unsigned)time(NULL);
    srand(randomseed);

    //open input data file - 'input.dat'
    ifstream EPIfile("EPIInput.dat");
    if (EPIfile.fail())
    {
        cout<<"Error in Main: Unable to open input.dat"<<endl<<endl;
        exit(-1);
    }

    CParameter cp;
    EPIfile >> cp;
    CStruct *Struct; char *Infile,*Outfile;
    //Preparation for structure analysis
    switch (cp.getStrTyp())
    {
        case 1://Plate

```

```

Struct=new CPlate;
Infile="PlateInfile.dat";
Outfile="Plate_NDE_Res.dat";
break;
case 2://Beam
Struct=new CBeam;
Infile="BeamInfile.dat";
Outfile="Beam_NDE_Res.dat";
break;
default:
cout<<"Error in Structure Type."<<endl;
exit (-1);
break;
}

//open output data file - 'res.dat'
ofstream resfile(Outfile);
if (resfile.fail())
{
cout<<"Error in Main: Unable to open res.dat!"<<endl<<endl;
exit(-1);
}

//write head of output file.
resfile << setw(60) << " [ Control parameters for inEPIB ] "
<< endl<< setw(60)
<< "===== "
<< endl<< endl<<"Runtime Random Seed = "<<randomseed
<<endl <<endl;

resfile<<cp;
cout << cp;
max_gen =cp.getMax_gen();
pop_size=cp.getPop_size();
MaxMinID=cp.getMaxMinID();
nvariable=cp.getNVariable();
Pgrad =cp.getpara_prob(0);
Pxover =cp.getpara_prob(1);

CEvaluation ce(pop_size);
CChromosome **cc = new CChromosome *[2*pop_size+1];
if( cc == NULL)
{
cout << "Fail to allocate memory for cc[]"<<endl;
exit(-1);
}
for (i=0;i<2*pop_size+1;i++)
cc[i]=new CChromosome(nvariable);

resfile<<setw(60)<<" [ Known information for backcalculation ]"<<endl;
resfile<<setw(60)<<"===== "<<endl;
Struct->Oput(Infile,resfile);
resfile<<setw(80)<<"===== "<<endl;
resfile<<setw(80)<<" Optimization Process by EP-NDE.ver3 "<<endl;
resfile<<setw(80)<<"===== "<<endl;
resfile<<setw(10)<<"Num_Gen"<<setw(10)<<"NumEval"
<<setw(20)<<"Fitness";
for (i=0;i<cp.getNVariable();i++)
resfile<<setw(17)<<"X["<<setw(2)<<i<<"]";
resfile<<endl;
Struct->stiffmass();

//Initialization - initialize chromosomes.
//-----
//produce Pop_size number of initial chromosomes.
initialize(pop_size,cc,cp);
//-----
//End of Initializing chromosomes.
//=====
ce.cal_fitness(1,cc,Struct);

for (num_gen=1;num_gen<=max_gen;num_gen++)
{
//=====
//Mutation - mutate chromosomes.
//-----
j=pop_size;

for (i=0;i<pop_size;i++)
{
//mutate cc[i].
double m=cr.random_dbl(0.0,1.0);
if (m<Pgrad)
Grad_search(cc[i],cc[pop_size+i],cp,Struct);
else if (m<(Pgrad+Pxover))
{
if (j==pop_size) j=i;
else
{
crossover(cc[i],cc[j],cc[pop_size+i],cp);
crossover(cc[j],cc[i],cc[pop_size+j],cp);
j=pop_size;
}
}
else Macro_Mutation(cc[i],cc[pop_size+i],cp);
}
if (j==pop_size) Macro_Mutation(cc[j],cc[pop_size+j],cp);
//-----
//End of Mutation.
//=====

//=====
//Evaluation: Compute and sort the fitness.
//-----
//Compute the fitness function fitness[2*Pop_size],
//Output the best chromosome in this generation.
ce.cal_fitness(2,cc,Struct);
ce.dsort(2,*P,cc,cp);
cout <<setw(10)<<num_gen<<setw(10)<<Struct->NumEval
<<setw(20)<<ce.get_fitness(0);
resfile<<setw(10)<<num_gen<<setw(10)<<Struct->NumEval
<<setw(20)<<ce.get_fitness(0);
for (j=0;j<nvariable;j++)
{
cout <<setw(20)<<cc[0]->getvariable(j);
resfile<<setw(20)<<cc[0]->getvariable(j);
}
//-----
//End of Evaluation.
//=====

//=====
//Selection:
//-----
//Chromosomes automatically selected since sorting is carried
//in CE.cal_fitness.
//Put ce.cc[j] to cc[j]. Selection is automatically finished here.

//Tournament selection:
ce.cal_weight(num_gen,cc,cp);
ce.dsort(2,*w,cc,cp);
cout <<endl;
resfile<<endl;
//-----
//End of Selection.
//=====

//=====
//Termination check:
//-----
//According to No. of generations finished, which is
//automatically checked by the do loop.
//No need to check here.
//-----
//End of termination check.
//=====

} //End of loop over generation.
for (int ii=0;ii<2*pop_size+1;ii++) delete cc[ii];
delete [] cc;
delete [] Struct;
return 0;
}
//=====

```