Global optimization and biomedical applications

Ye, Hong

2005

Ye, H. (2005). Global optimization and biomedical applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3533>

<https://doi.org/10.32657/10356/3533>

Nanyang Technological University

Downloaded on 26 Apr 2023 15:32:11 SGT

GLOBAL OPTIMIZATION AND BIOMEDICAL APPLICATIONS

YE HONG

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

2005

Global Optimization and Biomedical Applications

Ye Hong

School of Electrical & Electronic Engineering

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

2005

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

Ye Hong

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Lin Zhiping, for his guidance, discussions, patience and many invaluable suggestions and comments throughout the whole research project.

I also take this opportunity to thank Nanyang Technological University for the award of a research scholarship and both Workstation Resource Laboratory and Software Engineering Laboratory and staff for the excellent facilities and service without which the present work would not have been possible.

I would like to express my appreciation to Professor Raimund J. Ober of the University of Texas at Dallas and Professor E. Sally Ward of the University of Texas Southwestern Medical Center, USA, for their suggestions and technical collaborations concerning the work presented in Chapter 6 of the thesis.

I wish to acknowledge my parents and all family members who showed so much concern on me during my research. Their encouragement and unyielding support have contributed to the completion of this project.

To Yuan Yu

Contents

Acknowledgements	i
Contents	iii
Summary	vii
List of Figures	ix
List of Tables	xiv
List of Abbreviations	xvi
1 Introduction	1
1.1 Background and Motivations	1
1.1.1 Global Optimization	1
1.1.2 Global Optimization in Biomedical Applications	4
1.1.3 Challenges in Solving Nonlinear Optimization Problems	13
1.2 Objectives and Major Contributions	15
1.3 Organization of the Thesis	17
2 A Novel Ripple Search Strategy for Global Optimization Problems	19
2.1 Some Existing Global Optimization Algorithms	19
2.2 Ripple Search Strategy	24
2.2.1 Algorithm	26
2.2.2 Convergence Analysis	30
2.2.3 Implementation	36

2.2.4	Illustration	37
2.3	Test Functions and Results	40
2.3.1	Judge Model	41
2.3.2	Rosenbrock Model	42
2.3.3	A Set of Test Benchmarks	44
2.4	Conclusions	46
2.5	Appendix A. Derivation of the Probability p_s	48
3	Improved Subenergy Tunneling Function and Ripple Search for Efficient Global Optimization	52
3.1	Subenergy Tunneling Function	53
3.2	Improved Subenergy Tunneling Function	54
3.3	The Algorithm and Implementation	57
3.4	Neural Networks and the Feedforward Neural-Network Learning Problems	60
3.4.1	Artificial Neural Networks	60
3.4.2	Supervised Training in Neural Networks	62
3.5	Neural Network Applications	64
3.5.1	XOR Problem	65
3.5.2	Function Approximation	65
3.5.3	The 4-bit and 10-bit Parity Problems	71
3.5.4	Sonar Problem	76
3.5.5	Two-spiral Problem	76
3.6	Conclusions	81
4	Function Extensions for Efficient Global Optimization	82
4.1	Function Extension Techniques	82
4.1.1	Non-iterative Function Extension	84
4.1.2	Iterative Function Extension	90
4.2	Function Extensions for Efficient Global Optimization	97
4.2.1	Implementation of Non-iterative Function Extension with Global Optimization	102

4.2.2	Implementation of Iterative Function Extension in Global Optimization	104
4.3	Test and Simulation Results	105
4.4	Conclusions	106
5	Parallel Coordinates for Moderate- and High-dimensional Global Optimization	108
5.1	Multidimensional Visualization Techniques and Parallel Coordinates	109
5.2	Developed Parallel Coordinates	109
5.3	Optimization with Parallel Coordinates	111
5.3.1	Optimization with General Parallel Coordinates	111
5.3.2	Optimization with Section Parallel Coordinates	114
5.4	Test Functions and Results	116
5.4.1	Test Results of Simulated Annealing with Parallel Coordinates . . .	116
5.4.2	Test Results of Ripple Search Strategy with Parallel Coordinates . .	123
5.5	Discussion: From Parallel Coordinates to Radiation Coordinates	125
5.6	Conclusions	127
6	Achievable Accuracy of Parameter Estimation for Two-Dimensional NMR Experiments	135
6.1	Fundamentals	135
6.2	Analytical Expression for the Fisher Information Matrix	138
6.3	Cramer Rao Lower Bound and Parameter Estimation	146
6.4	Experiment Design Using the Cramer Rao Lower Bound	149
6.4.1	Non-uniform Sampling	150
6.4.2	Averaging to Improve Parameter Estimation Accuracy	151
6.5	Global Optimization in Parameter Estimation of 2D NMR Data	154
6.6	Conclusions	160
6.7	Appendix A. The Entries of the Fisher Information Matrix	161
6.7.1	A.1. Derivation of Entries of Fisher Information Matrix	161
6.7.2	A.2. List of Entries of Fisher Information Matrix	166

7	Biomolecular Structure Determination from NMR Data	170
7.1	Procedure of Structure Determination from NMR Data	170
7.2	Definition of Potential Energy Function	174
7.3	Development of Potential Energy Function	178
7.3.1	Potential Energy Function Development	178
7.3.2	Transformation from Cartesian Coordinates to Internal Coordinates	179
7.4	Applications to Molecular Structure Determination	182
7.4.1	Organic Molecule 1-Propanol	184
7.4.2	Organic Molecule 2-Methyl-1-Propanol	187
7.4.3	Deoxyribonucleic Acid 3REC	189
7.5	Conclusions	191
8	Conclusions and Recommendations for Further Research	193
8.1	Conclusions	193
8.2	Recommendations for Further Research	197
	Author's Publications	199
	Bibliography	200

Summary

In this thesis, new global optimization algorithms are presented for solving nonlinear global optimization problems. These problems are difficult to resolve because the nonlinear objective functions contain local minima that may trap conventional local optimization algorithms. In order to find good solutions of nonlinear optimization problems, the thesis focuses on the following two key issues: how to escape from local minima and how to find global solutions efficiently and accurately.

To avoid getting trapped in local minima, a novel ripple search strategy is proposed, as a stochastic global optimization method combining a novel ripple type search method and a proper local optimization algorithm, to find the global solution of a given optimization problem starting from a random initial point. The ripple search strategy differs from some other global optimization methods that explore search space randomly, hence it does not suffer the disadvantages of these methods and is more efficient and accurate to find the global minimum. Its probabilistic convergence to the global solution is validated theoretically. The method requires only a few control parameters. Therefore it is conceptually simple and fairly easy to apply to diverse applications.

To achieve the goal of finding global solutions efficiently and accurately, four approaches are proposed. First, an improved subenergy tunneling function is designed over the original one used in the TRUST method. A new global optimization is then proposed based on the combination of the improved subenergy tunneling function and the ripple search strategy to inherit the advantages of both methods. Second, two function extension techniques, called the non-iterative function extension and iterative function extension, are proposed to transform the objective function while preserving the characteristics relevant for optimization. The above techniques are then used for global optimization algorithms

to identify promising regions worth developing and save the computational cost of unpromising ones. Third, inspired by multidimensional visualization techniques, Parallel Coordinates is developed and introduced into the optimization research area to speed up global optimization algorithms by transforming the original search domain into another search domain with lower dimensions. As computational cost increases exponentially with the size of the problem, the saving in the dimension of the search domain can improve the efficiency of the optimization process in a straightforward way.

Experimental results are shown on extensive benchmarks and neural network training problems by the proposed algorithms. In general, our methods achieve better solutions with lower computational cost than existing methods.

A large number of biomedical applications are global optimization problems, for example, parameter estimation of nuclear magnetic resonance (NMR) data and biomolecular structure determination, which are two powerful tools for research in structural biology. In the thesis, we first give the analytical expression for the Fisher information matrix and show how the Cramer Rao Lower Bound can be calculated for data obtained from two-dimensional NMR experiments, which can be generalized straightforwardly to multidimensional NMR experiments. The Cramer Rao Lower Bound is then compared with the variance of parameter estimates for simulated NMR data by ripple search strategy. Our results also show how the Cramer Rao Lower Bound can be used to analyze whether an experimental design can be improved to provide experimental data which can result in parameter estimates with higher accuracy. The concept of non-uniform averaging in the indirect dimension is introduced and studied in connection with non-uniform sampling of the data. Finally, we show the procedure of biomolecular structure determination from NMR data. The potential energy function is explained in detail and further developed. The ripple search strategy is applied to determine the 3D structures of two organic molecules, 1-Propanol and 2-Methyl-1-Propanol, and one deoxyribonucleic acid 3REC. The structures found by our proposed method are nearly the same as the standard ones in the literature. The practical applicability of the developed potential energy function and the proposed technique on biomolecular structure determination is verified.

List of Figures

1.1	A classification of optimization problems.	2
1.2	An atom structure.	5
1.3	Procedures of structure determination from NMR data	9
1.4	NMR spectra of a peptide: (a) the 1D spectrum; (b) contour plot of the 2D spectrum.	10
2.1	The combination mesh and contour plot of the MATLAB peaks function for illustration	25
2.2	The sketch of ripple search in 2D case	30
2.3	The sketch of ripple search in the 2D case for convergence analysis	34
2.4	Contour plot of minimization the peaks function by ripple search strategy .	38
2.5	Plot of the designed function viewed from two different angles.	39
2.6	Search result by the ripple type search.	39
2.7	Search result by the random search method.	40
2.8	Points generated on a single 'ripple' in 3-D.	40
2.9	The sketch of probability density function $f_l(l)$	51
3.1	Comparison of the behaviors of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$	56
3.2	Comparison of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ for one-dimensional func- tion $f(x) = [\sin(2x) - x - 1]^2$ [44].	56
3.3	Model of a neuron (left) and a three-layer feedforward neural network (right).	61
3.4	Result plot of the proposed method: 1 – 7 – 1 neural network, target accuracy is 0.1. (a) Training result at the function value of 0.0807 ; (b) Testing result at the function value of 0.0760.	67

3.5	Result plot of the proposed method: $1 - 7 - 1$ neural network, target accuracy is $1e-3$. (a) Training result at the function value of $9.7253e-004$; (b) Testing result at the function value of 0.0010.	67
3.6	Result plot of the proposed method: $1 - 10 - 1$ neural network, target accuracy is 0.025. (a) Training result at the function value of 0.0231; (b) Testing result at the function value of 0.0200.	69
3.7	Result plot of the proposed method: $1 - 10 - 1$ neural network, target accuracy is $1e-3$. (a) Training result at function value of $4.4695e-4$; (b) Testing result at function value of $4.4106e-4$	69
3.8	Results of using different objective functions and training process for $1 - 7 - 1$ neural network with 50 training data corrupted by Gaussian noise with standard deviation of 0.25: (a) The sum of squared-errors objective function; (b) The sum of logistic loss function and limiting the training process.	72
3.9	Results of using different objective functions and training process for $1 - 10 - 1$ neural network with 20 training data corrupted by Gaussian noise with standard deviation of 0.25: (a) The sum of squared-errors objective function; (b) The sum of logistic loss function and limiting the training process.	72
3.10	Two-Spiral problem: (a) Training samples ; (b) Decision region.	78
3.11	Decision regions of networks trained by the proposed method with different number of hidden units: (a) A $2 - 8 - 1$ neural network ; (b) A $2 - 9 - 1$ neural network.	78
3.12	Decision regions of $2 - 4 - 1$ feedforward network with shortcuts trained by the proposed method	79
3.13	Decision regions of $2 - 5 - 1$ feedforward network with shortcuts trained by the proposed method	80
3.14	Decision regions of $2 - 6 - 1$ feedforward network with shortcuts trained by the proposed method	80
4.1	Desired properties of the transformation.	83

4.2	Illustration of non-iterative function extension: $\mathbf{J}'_1$	87
4.3	Illustration of non-iterative function extension: $\mathbf{J}'_2$ and $\mathbf{J}$	88
4.4	Comparison of $f_{non_i_exe}(\mathbf{X})$ and $f(\mathbf{X})$ as an example of 1D function.	88
4.5	Function and contour plot of $f_1(\mathbf{X})$ as a 2D example	89
4.6	Function and contour plot of $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$	89
4.7	Function and contour plot of $f_{non_i_exe}(\mathbf{X})$ for $f_1(\mathbf{X})$	90
4.8	Function and contour plot of $f_{non_i_exe}(\mathbf{X})$ for $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$	90
4.9	Projection on x_1 - x_2 plane: comparison of $f_{non_i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$	91
4.10	Function and contour plot of $f_2(\mathbf{X})$ as 2D example	91
4.11	Function and contour plot of $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$	92
4.12	Function and contour plot of $f_{non_i_exe}(\mathbf{X})$ for $f_2(\mathbf{X})$	92
4.13	Function and contour plot of $f_{non_i_exe}(\mathbf{X})$ for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$	93
4.14	Projection on x_1 - x_2 plane: comparison of $f_{non_i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$	93
4.15	Comparison of $f_{i_exe}(\mathbf{X})$ and $f(\mathbf{X})$ for an example of 1D function.	96
4.16	Function and contour plot of $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$	97
4.17	Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_3(\mathbf{X})$	97
4.18	Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$	98
4.19	Projection on x_1 - x_2 plane: comparison of $f_{i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$	98
4.20	Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_2(\mathbf{X})$	99

4.21	Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$	99
4.22	Projection on x_1 - x_2 plane: comparison of $f_{i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$	100
4.23	Function and contour plot of $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$	100
4.24	Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$	101
4.25	Projection on x_1 - x_2 plane: comparison of $f_{i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$	101
5.1	Parallel Coordinates display of a 9-dimensional data element by: (a) a polygonal line; (b) a smooth curve.	110
5.2	Illustration of Section Parallel Coordinates.	114
5.3	Radiation coordinates display of a multi-dimensional data element by curve.	126
6.1	The real part of the first trace of a 1024×32 2D data set with noise level at -20 dB.	148
6.2	Comparison of lower bound obtained for CRLB for the variance estimate of ω_{11} (solid line 'o-') with estimated variance based on nonlinear least-squares parameter estimate ('x') for five noise levels ($-10, -20, -30, -40, -50$ dB).	149
6.3	CRLB and the estimated variance for parameter ω_{22} in the case of $K = 2$, $L = 2$	159
7.1	Procedure of structure determination from NMR data	171
7.2	Four atoms connected in the order A - X - Y - B	172
7.3	Definition of the positive or negative value of a dihedral angle	172
7.4	The PDB file of a simple compound ethanol including atom coordinates.	175
7.5	Display in Rasmol and the command line window.	175
7.6	The sketch of $y = \frac{1}{2} (\tanh 300 (x - b) - \tanh 300 (x - a) + 2)$	179

7.7	Illustrations of the bond length L_{12} , the bond angle θ_{234} , the atom distance r_{13} and the torsion angle φ_{5432}	182
7.8	Ball and stick display in Rasmol for the molecule 1-Propanol.	184
7.9	Primary structure of the molecule 1-Propanol.	185
7.10	Standard structure of the 1-Propanol in display of sticks.	185
7.11	Structure of the 1-Propanol found by ripple search strategy in display of sticks.	186
7.12	Ball and stick display in Rasmol for the molecule 2-Methyl-1-Propanol. . .	188
7.13	Primary structure of the molecule 2-Methyl-1-Propanol.	188
7.14	Standard structure of the 2-Methyl-1-Propanol in display of sticks.	188
7.15	Structure of the 2-Methyl-1-Propanol found by ripple search strategy in display of sticks.	189
7.16	Ball and stick display in Rasmol for the biomolecule 3REC.	189
7.17	Standard structure of the 3REC in display of sticks.	190
7.18	Structure of the 3REC found by ripple search strategy in display of sticks. .	190

List of Tables

2.1	Solutions of the Ripple Search Strategy	37
2.2	Data for the Judge Model	42
2.3	Comparison of Three Optimization Algorithms on Judge Model	42
2.4	Comparison of Four Global Optimization Algorithms on Rosenbrock Model with Four Variables	43
2.5	Results and Comparison on Standard Global Optimization Problems . . .	45
3.1	Results for the XOR Problem	66
3.2	Results for the Function Approximation Problem Using 50 Training Patterns	68
3.3	Results for the Function Approximation Problem Using 20 Training Patterns	70
3.4	Results for the Function Approximation Problem Using 50 Noisy Training Patterns	73
3.5	Results for the Function Approximation Problem Using 20 Noisy Training Patterns	73
3.6	Comparison Results for the 4-Bit Parity Problem	74
3.7	Results for Problems with Smaller Network Structures	75
3.8	Comparison Results for the 10-Bit Parity Problem	76
3.9	Comparison Results for Sonar Problem	77
3.10	Comparison Results for Spiral Problem	80
4.1	Comparison Results on Cost Function	107
5.1	The Global Minima $\mathbf{P}$ and the Starting Points of the Quasi-Rosenbrock Functions	119

5.2	The Global Minima $\mathbf{P}$, the Optimal Function Values f^* and the Starting Points of the Quasi-COST Functions	120
5.3	Tests on Quasi-Rosenbrock Function with 9 Variables	128
5.4	Tests on Quasi-Rosenbrock Function with 18 Variables	128
5.5	Tests on Quasi-Rosenbrock Function with 27 Variables	129
5.6	Tests on Quasi-Rosenbrock Function with 56 Variables	129
5.7	Tests on Quasi-Rosenbrock Function with 82 Variables	130
5.8	Tests on Quasi-COST Function with 30 Variables	130
5.9	Comparisons of Different Sets of Values for k , u and v on Quasi-COST Function with 30 Variables	131
5.10	Tests on Quasi-COST Function with 50 Variables	131
5.11	Test Results Averaged over 10 Runs for Each Problem	132
5.12	Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 9 Variables	132
5.13	Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 18 Variables	133
5.14	Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 27 Variables	133
5.15	Test Results Averaged over 10 Runs for Each Problem	134
6.1	CRLB for Different Sampling Methods. $u u$ ($n n$) stands for (non-)uniform sampling in both dimensions; $n u$ stands for non-uniform sampling in the indirect dimension and uniform sampling in the direct dimension. The precise sampling scheme is explained in the text.	152
6.2	CRLB for Different Sampling and Weighting Methods. $u u$ ($n n$) stands for (non-)uniform sampling in both dimensions; $n u$ stands for non-uniform sampling in the indirect dimension and uniform sampling in the direct dimension. UA (NA) stands for (non-)uniform averaging in the first dimension. The precise sampling and weighting scheme is explained in the text. .	154
6.3	Simulation Results and Comparisons for Noiseless 2D NMR Data	157

List of Abbreviations

ABBREVIATIONS	FULL EXPRESSIONS
ASA	Adaptive simulated annealing
COSY	COrrrelated SpectroscopY
CRLB	Cramer-Rao lower bound
DEA	Differential evolution algorithm
DFT	Discrete Fourier transform
FID	Free induction decay
IT	Initial temperature
NMR	Nuclear magnetic resonance
NN	Neural network
NOE	Nuclear Overhauser effect
NOESY	Nuclear Overhauser Enhancement SpectroscopY
RMSD	Root-mean-square deviation
SA	Simulated annealing
SNR	Signal-to-noise ratio
SQP	Sequential quadratic programming
TAS	Temperature annealing scale
TOCSY	TOtal Correlation SpectroscopY
TRS	Temperature ratio scale
TRUST	Terminal repeller unconstrained subenergy tunneling

Chapter 1

Introduction

1.1 Background and Motivations

1.1.1 Global Optimization

A large variety of problems and applications, arising in science, engineering and applied science in many research areas such as molecular biology, economics and operational research, can be perceived and modeled as optimization problems. Optimization problems are composed of three basic components: a set of unknowns or variables, a set of constraints that specify feasible values of the variables and a function to be minimized or maximized. In this thesis, minimization is assumed in the general discussion of the optimization problems. A mathematical formulation of the minimization problem is defined as follows:

$$\min_{\mathbf{X} \in R^n} f(\mathbf{X}) \quad (1.1)$$

subject to m_c constraints

$$g_j(\mathbf{X}) \leq 0, j = 1, 2, \dots, m_c \quad (1.2)$$

where $\mathbf{X} = (x_1, x_2, \dots, x_n)$ is an n -dimensional vector of unknown variables, $g_j(\mathbf{X})$ is a vector function and R^n is the feasible domain of $\mathbf{X}$. The function $f(\mathbf{X})$ is called an objective function or a cost function. When $m_c = 0$, it is an unconstrained problem. The

global solution $\mathbf{X}^*$ is sought which satisfies all stated feasibility constraints and minimizes the value of the objective function $f(\mathbf{X})$ [1].

A) ***A Taxonomy of Optimization Problems***

There are many types of optimization problems. Variables can be continuous, discrete, or symbolic. The objective function can be continuous or discrete, and take linear or nonlinear forms. Constraints can also have linear or nonlinear expressions. Figure 1.1 shows a taxonomy of optimization problems.

In Figure 1.1, optimization problems are classified into continuous and discrete problems. A problem is continuous if the variables take continuous real values. A problem is discrete if the variables take discrete, usually integer values.

Continuous and discrete optimization problems are further classified into constrained optimization and unconstrained optimization based on the presence of constraints. Problems without constraints fall into the class of unconstrained optimization.

There are two types of optimal points of an optimization problem: local minimum and global minimum. A local minimum has the smallest value in a local feasible region surrounding itself, whereas a global minimum has the smallest value in the whole feasible domain. A optimization problem is unimodal if its objective function has one local minimum, which is the global minimum at the same time. A problem is multimodal if its objective function has more than one local minimum. General nonlinear functions are multimodal.

Figure 1.1: A classification of optimization problems.

B) *A Taxonomy of Optimization Algorithms*

A large number of optimization methods have been developed to solve nonlinear optimization problems. Optimization algorithms are classified into local optimization and global optimization methods. Global optimization methods have the ability to find global optimal solutions given sufficient time, while local optimization methods do not have the same ability.

In an optimization problem, if the objective function is unimodal in the domain of interest, one can choose solutions among many good local optimization algorithms, such as Gauss-Newton method [2], Levenberg-Marquardt method [3], Broyden, Fletcher, Goldfarb, and Shanno (BFGS) Quasi-Newton method [4]-[7] and sequential quadratic programming (SQP) algorithm [8]. These local optimization methods can always find the minimum by moving downhill in an iterative manner.

However, if the objective function is multimodal in the domain of interest, which is visualized by ‘hilly landscapes’ corresponding to projections of the objective function into selected subspaces, local optimization algorithms tend to stop at the first minimum encountered. They may only give local solutions and cannot easily find the global one, depending on the starting search point. It is to be noted that the number of local solutions is typically unknown and can be quite large. Furthermore, the objective function values of various local and global solutions may differ significantly from each other. These remarks indicate the extreme difficulty of optimization problems.

The emerging field of Global Optimization deals with mathematical programming problems in the presence of multiple local optima. Especially during the last three decades, there has been a significant amount of research in developing algorithms for finding the global minimum of optimization problems. The explosive growth of the computing capabilities shows great promise towards addressing these issues [9].

Global optimization algorithms can be categorized into two types: stochastic methods [10, 13] and deterministic methods [14, 15]. Stochastic methods involve function evaluations at random trial points in some prescribed way. Examples of this kind of

methods are random search, simulated annealing, genetic algorithms, tabu search etc. Under mild conditions on the test functions, these stochastic methods guarantee asymptotic convergence to the global optimum as the number of trial points increases. Deterministic methods involve no element of randomness. Examples of these methods are branch and bound, interval analysis method, homotopy method and so on. Some of them apply deterministic heuristics to bring a search out of a local minimum, while others, like covering methods, try to partition the search space into subregions by using bounds. Deterministic methods can guarantee absolute success in theory, but only by making certain restrictive assumptions on the objective functions.

Deterministic methods are recognized as not feasible for practical applications, because when problems are with a few more dimensions, they will be fairly computational expensive and complex to implement. Though some approximate strategies are usually applied to high dimensional cases, the trade-off between efficiency and accuracy exists. Thus, deterministic methods are only suited for small problems with limited size of dimensions. In contrast, stochastic methods assume little knowledge on the objective functions and are often simpler and easier to implement. Therefore, they can be adopted to handle large-scale and complex problems without much difficulty. However, most stochastic methods are inefficient in the way they explore the search space. Moreover, many searches may lead to the same local minimum, resulting in the waste of previous searches. Therefore, there is still a need to develop new stochastic global optimization methods, particularly for higher dimensional problems, such as some biomedical applications which will be discussed next.

1.1.2 Global Optimization in Biomedical Applications

A significant effort has been devoted in the last five decades toward theoretical and algorithmic studies in Biology. Over the past ten years, the area of global optimization has received a lot of attention from a variety of disciplines including biomedical applications, and this surge of interest is attributed to two main reasons. Firstly, a large number of

biology and medicine problems are indeed global optimization problems. Secondly, the existing local nonlinear optimization approaches may either fail to obtain even a feasible solution or are trapped to a local optimum solution. This thesis presents global optimization algorithms with biomedical applications in nuclear magnetic resonance (NMR) data processing and biomolecular structure determination with NMR data. Some relevant background knowledge concerning these applications are introduced as follows [16].

A) *Introduction to Biomolecules*

Most of the biologic matter examined with NMR are composed of biomolecules. Biomolecules, such as organic compounds, DNA, RNA, peptides, nucleic acids and proteins, are composed of atoms. An atom consists of protons, neutrons and electrons. The nucleus of the atom consists of protons and neutrons and is located right in the center of the atom where both protons and neutrons are stuck together. The proton is a subatomic particle that is positively charged and is found in the nucleus of the atom. The neutron is the subatomic particle that is neutral in charges and is also found in the nucleus of an atom. The electron is a negatively charged subatomic particle and is found outside of the nucleus. The structure of an atom is shown in Figure 1.2.

Figure 1.2: An atom structure.

Biomolecular structures are an important source of information for understanding biological function at the molecular level and provide the basis for many studies in research areas such as structure-based drug design and homology modeling. Accordingly, a number of disciplines and techniques are currently being employed towards achieving these goals: X-ray crystallography, nuclear magnetic resonance

(NMR), electron paramagnetic resonance (EPR), computational biology and chemistry, various laser-based spectroscopic techniques, protein expression, cryo-electron microscopy, synthetic chemistry and chemical engineering. The two main techniques for determining the three-dimensional (3D) structures of biological molecules are X-ray crystallography and solution NMR. In the past 20 years, NMR spectroscopy has proved itself as a potentially powerful alternative to X-ray crystallography for the determination of macromolecular 3D structures. NMR has the advantage over crystallographic techniques in that experiments are performed in aqueous solution as opposed to a crystal lattice. Moreover, every NMR active nucleus gives rise to an individual signal in the spectrum that can be resolved by multidimensional NMR techniques, hence NMR method provides complementary information about biomolecular structures to that from X-ray crystallography.

B) *Principle of NMR*

NMR is a phenomenon which occurs when the nuclei of certain atoms are immersed in a static magnetic field and exposed to a second oscillating magnetic field, usually at radio frequency (RF) [16]. These nuclei will resonate, or absorb energy at specific frequencies. Some nuclei experience this phenomenon, and others do not, depending upon whether they possess a property called spin. The spin can be thought of as a small magnetic field, and will cause the nucleus to produce an NMR signal. Spin comes in multiples of $1/2$ and can be $+$ or $-$. Individual unpaired electrons, protons, and neutrons each possesses a spin of $1/2$. Two or more particles with spins having opposite signs can pair up to eliminate the observable manifestations of spin. In nuclear magnetic resonance, it is unpaired nuclear spins that are of importance.

In a strong magnetic field with high magnetic field strength, the spins align themselves with two energy states: with or against the magnetic field of the NMR instrument. When energy is applied at exactly the right frequency, resonance occurs, and the spin flips from one state to the other. As the spins are flipping, they absorb and then radiate energy. The exact frequency of the spin flips, called Larmor frequency, identifies the kind of atom that is involved and the other atoms to which

it is connected in the biomolecule. If the magnetic field strength is B , then the Larmor frequency ν in frequency units is

$$\nu = -\frac{1}{2\pi}\gamma B, \quad (1.3)$$

where γ is the gyromagnetic ratio [17]. The Larmor frequency is a signed quantity indicating the rotating direction of its corresponding angular frequency. The frequency of the RF energy emitted by each nucleus in the biomolecules is reported in units of chemical shift. Chemical shift is actually proportional to frequency and the proportionality constant depends on the magnetic field strength of the particular NMR instrument used in the experiment. The chemical shift of a nucleus is the difference between the Larmor frequency ν of the nucleus and a standard ν_{REF} , divided by ν_{REF} . This quantity is reported in ppm and given the symbol δ ,

$$\delta = (\nu - \nu_{REF}) \times 10^6 / \nu_{REF} \quad (1.4)$$

In NMR spectroscopy, the magnetic field strength B normally appears in Equation (1.3). Conventionally, Larmor frequency of the hydrogen atom is referred to as the magnetic field strength expressed in megahertz (MHz) [17]. Higher magnetic field strength (up to 900 MHz) leads to higher information content and increased sensitivity, allowing the detection and characterization of smaller amounts of material and more complex biomolecules. For a given magnetic field strength, each type of nuclear species resonates at a different frequency. The chemical shift plays a very important role in the structure determination by NMR.

The basic phenomenon of NMR was discovered in 1945 [16]. Since 1960 the field of NMR has seen an explosive growth which started with the development of pulsed Fourier-transform NMR and multidimensional NMR spectroscopy and still continues today. NMR is one of the few non-destructive methods for analyzing structure and molecular dynamics. It is also information rich in terms of primary biomolecular structure (which atoms are attached to each other), secondary structure (which type of basic 3D structures are there, i.e. ring structures, layered molecular segments, etc.) and tertiary structure (how the biomolecule is folded or twisted in 3D space).

NMR spectroscopy is the use of the NMR phenomenon to study physical, chemical, and biological properties of matters. As a consequence, NMR spectroscopy finds applications in several areas of science. NMR spectroscopy is routinely used by chemists to study chemical structure using simple one-dimensional (1D) techniques. Two-dimensional (2D) techniques are used to determine the structure of more complicated biomolecules.

C) ***Biomolecular structure Determination from NMR Data***

Understanding the relationship between biomolecular structure and function is a central biomedical research area within structural biology, and is also crucial to the emerging field of structure-based drug discovery and design [18][19]. With the completion of the human genome project, biomolecular structure/function determination is now considered to be a major opportunity for harnessing the wealth of new information about the genome to achieve improvements in human health.

Nuclear magnetic resonance (NMR) is one of the promising techniques capable of determining three-dimensional (3D) structure of molecules at atomic resolution in nearly physiological conditions. It does not directly produce an image of a molecule including the positions of all atoms. Rather, it is able to yield a wealth of indirect structural information from which the 3D structure can be revealed by extensive data analysis and computer calculations. For example, the resonance frequency and many other experimentally accessible parameters are easily related to biomolecular structure and dynamics. Therefore NMR has attracted a lot of interests. In the following, a short introduction is devoted to the NMR methodology for biomolecular structure determination.

The procedure of determining 3D structures of biomolecules by NMR spectroscopy can be summarized as four steps: NMR experiments and signal processing, sequential resonance assignments, nuclear Overhauser effect (NOE) assignments and structure determination, as shown in Figure 1.3.

Step one: NMR experiments and signal processing

NMR experiments are performed on a NMR spectrometer consisting of two compo-

Figure 1.3: Procedures of structure determination from NMR data

nents: a high field superconductive magnet which produces an extremely homogeneous, strong static magnetic field and a console that can generate electromagnetic waves in any desired combination. For an NMR experiment a glass tube containing the solution of the biomolecule of interest is placed in the room temperature bore of the superconductive magnet. After a perturbation of the equilibrium state of the spins by one or several RF pulses an NMR signal can be observed. The signal consists of RF waves with frequencies that matches the energy difference between the spin-up and spin-down states of the individual nuclei involved. Since the system tends to return to its equilibrium state, the signal will decay (relax) and typically vanish within hundreds of milliseconds. The signal is detected by mounting a small coil of wire around the glass tube containing the sample. The whole process is analogous to the way in which oscillating voltage can be induced by a magnet rotating inside a coil. The voltage vs time signal is called free induction decay (FID). The FID consists of a sum of decaying cosine waves whose frequencies represent the resonance frequencies of the nuclei in the sample. The voltage is amplified and digitized by the receiver and recorded by the NMR spectrometer. The NMR data is usually buried in the noise caused by various factors, such as the imperfections in spectrometer and the condition of samples during the NMR data collection [17]. From this data the NMR frequency spectrum can then be obtained and processed

by a Fourier transformation.

Each one-dimensional (1D) NMR experiment consists of two sections: preparation and detection. During preparation the spin-system is set to a defined state. During detection the resulting signal is recorded. Usually 1D NMR spectra are far too complex for interpretation as most of the signals overlap heavily, see Figure 1.4(a) for an example. By the introduction of an additional spectral dimension to the 1D spectrum, as the two-dimensional (2D) spectrum shown in Figure 1.4(b), these overlapped signals are separated making the interpretation of NMR signals possible and extra information can be obtained.

Figure 1.4: NMR spectra of a peptide: (a) the 1D spectrum; (b) contour plot of the 2D spectrum.

In addition to preparation and detection which are already known from 1D experiments, a two-dimensional (2D) experiment has an indirect evolution time and a mixing sequence. There are three 2D spectra which are widely used for the structure determination: 2D COSY (COrelated SpectroscopY), 2D TOCSY (TOtal Correlation SpectroscopY) and 2D NOESY (NUclear Overhauser Enhancement SpectroscopY). In the COSY experiment, the magnetization is transferred by scalar coupling between two nuclei which are connected by one to three chemical bonds formed by the

electrons. The scalar coupling occurs because of the spin interaction of the nuclei with bonding electrons. Normally, couplings over one bond (1J), two bonds (2J) and three bonds (3J) apart are visible in the COSY spectrum. The three-bond coupling 3J , also called vicinal coupling, depends on the torsion angle between the coupled nuclei. This is described by the famous Karplus equation [17]:

$$^3J = A \cos^2(\phi - 60) - B \cos(\phi - 60) + C \quad (1.5)$$

where ϕ is torsion angle, A , B and C are constants, 3J is given in Hertz. Therefore, it is possible to derive parts of the torsion angle from the knowledge of scalar couplings.

In the TOCSY experiment, magnetization is dispersed over a complete spin-system of biomolecule by successive scalar coupling. The TOCSY experiment correlates all members of a spin system. Thus a characteristic pattern of signals results for each atom from which the atom can be identified. The TOCSY experiment can be used to identify the primary structure of a biomolecule.

In COSY and TOCSY experiments, magnetization transfer between spins is mediated by the through-bond scalar couplings. The NOESY, on the other hand, takes advantage of the through-space dipolar couplings. Dipolar coupling arises from the interaction of nuclear magnetic moments of two different nuclear spins. Magnitude of the coupling depends on the nuclear distance, hence this is a through-space interaction, unlike scalar coupling which takes place through chemical bonds [17]. The NOESY experiment is crucial for the determination of biomolecular structure as it provides distance constraints. It uses the dipolar interaction of spins, the nuclear Overhauser effect (NOE) for correlation of protons. The correlation between two protons depends on the distance between them, but normally a signal is only observed if their distance is smaller than 5 Angstroms. The NOESY experiment correlates all protons which are close enough. It also correlates protons which are distant in the atom sequence but close in space due to tertiary structure. This is the most important information for the determination of biomolecular structures.

Since spectral overlap is proportional to the size of the biomolecule being studied, 2D spectra of large biomolecules are often crowded with signals. Therefore, these

spectra are spreading out in a third dimension. There are, however, several disadvantages of using 3D NMR. The time required to acquire a spectrum increases with the increase of dimensionality. The sensitivity drops with increasing of dimensions. More detailed information on NMR experiments is described in [16].

Step two: Sequential resonance assignments

In NMR spectroscopy, once the NMR experiments are acquired, individual peaks in the spectra have to be assigned to sequence-specific locations in the chemical structure of the biomolecule before the distance information in the NOESY spectrum can be fully interpreted. The process of assigning chemical shifts to nuclei in the biomolecule is called sequential resonance assignment. A unique chemical shift value is assigned to individual atoms in the biomolecule because a chemical shift value depends on the electron density around the corresponding nucleus. Sequential resonance assignment plays a pivotal role in the structure determination process.

Step three: Nuclear Overhauser effect (NOE) assignments

Although a variety of NMR parameters contain structural information, the crucial information comes from NOE measurements which provide many distance constraints between the hydrogen atoms of a biomolecule. Consider two spins I and S which are close enough to have a through-space dipolar coupling but there is no scalar coupling. If the S spin magnetization deviates from the equilibrium, there will be a change in the I spin magnetization. The change in the I spin magnetization will manifest itself as a change in the intensity of the corresponding peak in the spectrum. It is this change in the intensity of the I spin when the S spin is perturbed which is termed the nuclear Overhauser effect. NOE experiment measures the changes in the intensity of the peaks and shows the difference spectrum compared to a reference spectrum in which the intensities are unperturbed [20]. Assignment of distance limits for pairs of protons in a biomolecule is based on observation of nuclear Overhauser effect between the protons. The intensity of the

NOE between a pair of protons can be related to the distance of separation between the proton pair. Generally speaking, an NOE peak with strong intensity may indicate that two protons are within 2.5 Angstroms of each other while a weak NOE peak corresponds to an upper limit of 5 Angstroms.

For a high resolution (high quality) structure determination, the maximal possible number of NOE constraints must be collected as input for the calculation of the complete 3D biomolecular structure.

Step four: Structure determination

In general, NMR data alone would not be sufficient to determine the positions of all atoms in a biological macromolecule. It has to be supplemented by information about the covalent structure of the molecule - bond lengths, bond angles, chiralities and planar groups - as well as by steric repulsion between non-bonded atom pairs. A molecular potential energy function is a convenient compilation of these information. The problem of finding correct molecular conformations is then to find conformations that minimize the potential energy function. The general approaches are based on global minimization of the potential energy function by simulated annealing protocols with biomolecular dynamics in several computer program packages, such as X-PLOR and DYANA [11][12]. All packages calculate in the end the Cartesian coordinates of the spatial biomolecular structures which are consistent with the set of constraints obtained from the NMR experimental data. More detailed information about structure determination with NMR data is described in Chapter 7.

1.1.3 Challenges in Solving Nonlinear Optimization Problems

Except for trivial cases, nonlinear optimization problems usually do not have closed-form solutions and cannot be solved analytically. Numerical methods have been developed to handle these problems. The process of solving an optimization problem becomes a process of searching for optimal solutions in the corresponding search space. Generally, the global optimization problems are considered very hard and complicated to resolve, because the

nonlinear objectives have many local minima that make global minima difficult to find and verify. There may exist terrains with large shallow or small but deep basins in the search space. Furthermore, the dimensions of optimization problems are large in many interesting applications, such as neural network training, NMR data processing and biomolecular structure determination.

The accuracy of the estimated parameters is always very important, especially in NMR spectroscopy because it plays an important role in determining the biomolecular structure. The difficulties of NMR parameter estimation result from the low inherent sensitivity of the technique, the high complexity and information content of NMR spectra and the frequent occurrence of overlapping or nearly overlapping peaks. Progress in the theoretical analysis of NMR is necessary to estimate the signal parameters under low signal-to-noise ratio (SNR) by using few data points which is of particular significance for multidimensional NMR experiments, because any savings in data in the indirect dimensions can lead to major savings of expensive experimental time.

One of the most significant and challenging problems in molecular biophysics and biochemistry is the determination of a 3D stable biomolecular structure. The solution probably requires the use of experimental or knowledge-based methods to provide initial guesses for subsequent numerical refinement. The numerical methods require: (a) suitable energy functions that can reliably model biomolecular behavior and (b) optimization methods that can find global minima of energy functions given many degrees of freedom available to biomolecular structures. At present significant progress in defining appropriate energy functions has been made. The primary impediment is the lack of effective global minimization methods for these large-dimensional problems. Due to the complex nature of biomolecules, computing the global minimum of the energy function is very difficult because it typically has a very large number of local minima which may grow exponentially with the size of the biomolecule.

Hitherto numerous global optimization algorithms have been proposed and investigated in the literature. However many of them have difficulties and disadvantages in either solving high-dimensional problems or lack of efficiency. Many problems still need to be exploited, improved and settled. There is an obvious need for developing effective and

accurate global optimization techniques.

1.2 Objectives and Major Contributions

As most applications in biomedicine can be naturally expressed as an unconstrained, continuous nonlinear global optimization problem, such as NMR data processing and biomolecular structure determination, one objective of this research is to propose some new approaches to tackle unconstrained continuous global optimization problems efficiently and accurately. Our proposed methods can also be used to constrained optimization problems by some transformational approaches to convert a constrained problem to an unconstrained one, such as penalty methods, barrier methods and Lagrangian methods [13]. Another objective of this thesis is to apply our proposed methods to NMR data processing and biomolecular structure determination applications.

The major contributions of this thesis can be summarized as follows:

- A new ripple search strategy is proposed for nonlinear global optimization problems. Its probabilistic convergence to the global optimum is then validated theoretically. The proposed algorithm is an improvement over some previous optimization techniques and is more efficient and accurate to find the global optimum. The new method is largely independent of the initial values and works well regardless of irregularities of the objective functions. The algorithm requires only a few control parameters hence it is conceptually simple and is fairly easy to apply to diverse applications.
- An improved subenergy tunneling function with ripple search strategy for efficient global optimization is proposed. It is based on the combination of the improved subenergy tunneling function developed from the original one of the TRUST method and the ripple search to inherit the advantages of both methods. The new subenergy tunneling function is introduced to identify and develop promising trial points which will certainly lead to a better solution than the previous ones, hence to save computational time on unpromising points. The performance on accuracy and efficiency of the new method is demonstrated to be better than that of the original

TRUST algorithm for extensive neural network training problems. It also compares favorably in general with other global optimization methods on the topic. The influence of noise injected in training signals on the proposed learning algorithm and the ways to reduce the noise corruption are also investigated.

- Two methods, non-iterative and iterative function extension techniques, exploiting function extension are proposed to improve the efficiency of global optimization algorithms. After applying the proposed function extension techniques, the objective function is transformed while preserving the characteristics relevant for optimization. By embedding the techniques into global optimization algorithms, such as ripple search strategy, it is more reliable than improved subenergy tunneling function for global search to identify promising trial points.
- Inspired by multidimensional visualization techniques, Parallel Coordinates is developed and introduced into optimization research area to speed up global optimization algorithms by transforming the original search domain into another search domain with lower dimensions. The proposed method is meant to address optimization problems of moderate and high dimensions. As the computational cost by global optimization algorithms may grow exponentially with the size of the problem, the reduction on the dimension of the search domain by Parallel Coordinates can reduce the cost of global search, local search and function evaluation during optimization process and consequently speed up the convergence rate of global optimization algorithms.
- The Cramer Rao Lower Bound provides a method to assess the best achievable accuracy of parameter estimates resulting from an unbiased estimation procedure. We give the analytical expression for the Fisher information matrix and show how the Cramer Rao Lower Bound can be calculated for data obtained from two-dimensional NMR experiments. The Cramer Rao Lower Bound is compared to the variance of parameter estimates for simulated data using ripple search strategy. Our results also show how the Cramer Rao Lower Bound can be used to analyze whether an experimental design can be improved to provide experimental data which can result

in parameter estimates with higher accuracy. The concept of non-uniform averaging in the indirect dimension is introduced and studied in connection with non-uniform sampling of the data.

- We show the procedure of biomolecular structure determination from NMR data. Then the potential energy function is further developed and the detailed transformation formulae from Cartesian coordinates to internal coordinates are provided. Finally, the ripple search strategy is applied to determine the 3D structures of two organic molecules, 1-Propanol and 2-Methyl-1-Propanol, and one deoxyribonucleic acid 3REC. The structures found by our proposed method are nearly the same as the standard ones in the literature. The practical applicability of the developed potential energy function and the proposed technique on biomolecular structure determination is verified.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapters 2 to 5 deal with the proposed global optimization algorithms. The applications of proposed methods in NMR data processing are considered in Chapter 6. The applications of global optimization algorithms to biomolecular structure determination are discussed in Chapter 7. More details are given below.

- In Chapter 2, a new ripple search strategy is proposed and illustrated for nonlinear global optimization problems. Its probabilistic convergence to the global optimum is then analyzed. Test results on extensive benchmarks are also addressed. The proposed method has the potential for parallel processing.
- In Chapter 3, we propose an improved subenergy tunneling function with ripple search for efficient global optimization. The performance of the new method is demonstrated and compared with other optimization methods on neural network training problems, with particular emphasis on small-sized neural networks. The

influence of noise injected in training signals on the proposed learning algorithm and the ways to reduce the noise corruption are also investigated.

- In Chapter 4, two methods, non-iterative and iterative function extension techniques, exploiting function extension are proposed to improve the efficiency of global optimization algorithms. Test results demonstrate the performance of the proposed approaches.
- In Chapter 5, the concept of Parallel Coordinates, a coordinate system originating from multidimensional visualization, is developed and introduced into optimization research area to speed up the convergence rate of global optimization algorithms. Extensive numerical studies are given to show the superiority of the proposed method.
- In Chapter 6, we give the analytical expression for the Fisher information matrix and show how the Cramer Rao Lower Bound can be calculated for data obtained from two-dimensional NMR experiments and used to analyze whether an experimental design can be improved to provide experimental data which can result in parameter estimates with higher accuracy. The concept of non-uniform averaging in the indirect dimension is introduced and studied in connection with non-uniform sampling of the data. Additionally, ripple search strategy is applied to globally optimize the parameters of 2D exponentially damped sinusoids in NMR signal. The CRLB is also shown by numerical simulations to match the variance of NMR spectroscopy parameters estimated by our proposed optimization algorithm.
- In Chapter 7, potential energy function is explained in detail and further developed. The ripple search strategy is applied to determine the 3D structures of two organic molecules and one biomolecule. The structures found by our proposed method are nearly the same as the standard ones in the literature. The practical applicability of the developed potential energy function and the proposed technique on biomolecular structure determination is verified.

Chapter 2

A Novel Ripple Search Strategy for Global Optimization Problems

Nonlinear optimization problems are generally difficult to solve due to the complicated terrain and the large number of local minima in the search space. This chapter focuses on how to develop efficient global optimization algorithm to escape from these local minima. First, some existing global optimization algorithms are reviewed. Then a novel ripple search strategy is proposed and illustrated in detail. Finally, the new method is tested on benchmarks to demonstrate its performance and is compared with other global optimization methods in the literature.

2.1 Some Existing Global Optimization Algorithms

Many global optimization algorithms have been developed to solve nonlinear optimization problems. In this section, the global optimization algorithms used as comparisons in this and later chapters are introduced.

A) *Random Search and Adaptive Random Search*

The class of random search algorithms uses the basic strategies of random sampling with or without local optimization. Three of the simplest random search methods are pure random search, single-start and multistart random search. In all algorithms, the trial points are generated randomly, chosen from a uniform distribution

on a feasible region. In pure random search no local optimization is performed. Normally explorations would be made in the neighborhood of the best point found in the sampling phase. In single-start random search, a single local search is performed starting from the best point found in the sampling phase. In multistart random search, each random point is taken as a starting point for a local optimization. All three algorithms are simple but rather inefficient, because effort is spent equally on both unpromising and promising subregions. In order to improve on their efficiency these methods have been modified in different ways, giving rise to new random search methods, such as random line search and adaptive random search. Random line search may be considered a modification of pure random search. Instead of a random point, a random line is generated uniformly in feasible region. In most cases the method itself is inefficient because the probability of penetrating the neighborhood of the global minimum is normally very small. Adaptive random search may be considered as modifications of pure random search by changing the uniform distribution into a more complicated one and by taking into account the results of previous minimization steps. Adaptation can be one of two types: structural or parametric. The first one normally means switching from one algorithm to another. For parametric adaptation, parameters of some density function for distributing trial points are determined and heuristically justified. Therefore the real efficiency of the algorithm depends on a successful choice of many parameters [13].

B) *Simulated Annealing*

Simulated annealing is a stochastic optimization technique for obtaining good solutions to difficult optimization problems. It has received much attention over the past twenty years. The interest in simulated annealing began with the work of Kirkpatrick et al. [21] in 1983 and independently of Cerny [22] in 1985. They showed how a model for simulating the annealing of solids, as proposed by Metropolis et al. in 1953 [23] could be used for optimization problems, where the objective function to be minimized corresponds to the energy of the states of the solid.

Simulated annealing is one of many heuristic approaches and a Markov chain Monte

Carlo method designed to give a good though not necessarily optimal solution, within a reasonable computing time. Starting from an initial solution perhaps chosen at random, simulated annealing generates a new solution $\mathbf{X}'$ in the neighborhood of the current solution $\mathbf{X}$. Then, calculate the change in the objective function, i.e. $\Delta E = f(\mathbf{X}') - f(\mathbf{X})$. In minimization problems, if $\Delta E < 0$, transition to the new solution, which is a downhill move, is accepted. If $\Delta E \geq 0$, then transition to the new solution is accepted only with a specified probability obtained by the function $e^{-\Delta E/T}$, where T is a control parameter called the temperature. The acceptance of uphill moves statistically allows the algorithm to escape from local minima. Simulated annealing repeats this process M times at each temperature, where M is a control parameter called the epoch length. The value of T is gradually decreased by a cooling function, such as $T_t = r_t T_{t0}$, where T_{t0} is the initial temperature, r_t is the temperature reduction factor.

Note that simulated annealing first builds up a rough view of the surface by moving with large step lengths. As the temperature falls and the step length decreases, it slowly focuses on the most promising area. Lundy and Mees [24] proved that the simulated annealing algorithm converges asymptotically with probability close to 1 provided that certain conditions are met, including an infinitely long cooling schedule.

Simulated annealing plays an important role in practice because of its generality and robustness. Early simulated annealing algorithm considers combinatorial systems, where the system's state depends on the configuration of variables. Perhaps the best known is the traveling salesman problem [22, 25], in which one tries to find the shortest trip distance connecting a number of cities. Simulated annealing has also been extended to optimization problems for continuous variables. Implementation of simulated annealing for continuous variable by Corana et al. [26] appears to offer the best combination of ease of use and robustness. Additionally, there has been progress on the theoretical results from a mathematical analysis of the method [27], as well as many computational experiments comparing the performance of simulated annealing with other methods for a range of problems [26, 28].

Simulated annealing has been used successfully in job shop scheduling, image processing, computer and circuit design, pollution control, neural networks, reconstruction of polycrystalline structures, etc [29]-[31].

Considering that simulated annealing is a heuristic algorithm for obtaining good, though not necessarily optimal solutions to optimization problems, it has several attractive features. First of all, once a neighborhood structure has been devised, the algorithm only occupies a few lines of code to perform. Secondly it can be generally applied to a wide range of problems. Thirdly simulated annealing can provide high quality solutions to many problems. Finally, it explores the objective function's entire surface and tries to optimize the function while moving both downhill and uphill. Thus, it is largely independent of the starting values, often a critical input in local optimization algorithms. Furthermore, it can escape from local minima and find the global solution by the downhill and uphill moves. Simulated annealing also makes less stringent assumptions regarding the function than do local optimization algorithms.

Simulated annealing must not be regarded as a panacea. The selection of such control parameters as the initial value of temperature is very important since parameter values may have a significant influence on the performance of the algorithm. However, it is not easy to find out good parameter values. Experience and careful selection are needed to devise an appropriate neighborhood structure and cooling schedule to obtain an efficient algorithm [32, 33]. Another problem is that convergence may take a long time and also a great deal of computational effort. If the temperature drops sharply or the initial value of temperature is not high enough, the execution time will be short, but simulated annealing will be easily trapped to local minima. If the initial value of temperature is high enough and the temperature drops very slowly, good solution may be obtained, but the execution time will be much longer.

C) ***Adaptive Simulated Annealing***

Adaptive simulated annealing (ASA) is a global optimization algorithm based on an

associated proof that the parameter space can be sampled much more efficiently than by using other previous simulated annealing algorithms [34]. Among its numerous control parameters, there are three more essential than others, they are temperature ratio scale (TRS), temperature annealing scale (TAS) and initial temperature (IT). ASA is developed to find the best global fit of a nonlinear constrained nonconvex cost function over a d -dimensional space. The introduction of reannealing permits adaptation to changing sensitivities in the multidimensional parameter space.

Although there are a lot of similarities between SA and ASA, the main difference resulting in ASA being faster than SA is the cooling function $T_t = T_{t0} \exp(-C_1 K^{1/d})$, where T_{t0} is the initial temperature, C_1 is a constant, K is the annealing time. The new temperature T_t decreases exponentially in annealing time K .

D) ***Genetic Algorithm***

Genetic Algorithm (GA) is a family of computational models based on evolution. The idea of evolution involves reproduction of offspring with better genes and where mutation and crossover takes place. The parents, who have better genes, are chosen to procreate the children who will inherit these genes.

GA starts with a current population (solutions). A pre-selection from this population is done to form an intermediate population. From here, crossover, that is, inheriting the similar set of solutions from the parent solution, and mutation, where there is random improvement manipulation of the solution, takes place to create a new generation. Hence crossover and mutation operators are important components of the GA.

E) ***Differential Evolution Algorithm***

Differential evolution algorithm (DEA) possesses similar mechanisms to GA, such as, the crossover and mutation, hence both are classified as evolutionary algorithm. However, they are search techniques employing different strategies. The basic idea of DEA is to use the difference of two randomly selected parameter vectors, within the search space, to obtain the third parameter vector via a trial vector. The

parameter vector is also called individual or genome. Among its many control parameters, there are three more essential than others. They are population size NP , amplifying differential factor F and crossover constant CR . In the initialization stage, DEA creates NP parameter vectors, known as a population. The value of NP does not change during the minimization process. The amplifying differential factor F is a real and constant factor which controls the amplification of the differential variation between two randomly selected parameter vectors. Choosing a subgroup of parameter vectors for mutation is similar to a process known as crossover in the evolution theory. The crossover constant $CR \in [0, 1]$ denotes the crossover probability. When a random number generated on the interval $[0, 1]$ is smaller than the value of CR , crossover operation happens to merge the genetic information of two or more parent individuals for producing one or more descendants. This technique is reported possessing parallel search nature and can find the global minimum in reasonable convergence speed. Unlike other evolution techniques which need a high number of function evaluations to find the optimal solution, differential evolution algorithm could converge more efficiently to the solution [35].

F) *Terminal Repeller Unconstrained Subenergy Tunneling*

Terminal Repeller Unconstrained Subenergy Tunneling (TRUST) formulates the global optimization problem as the solution of a deterministic dynamical system incorporating terminal repellers and a subenergy tunneling function. Benchmark tests comparing this method to other global optimization procedures are presented with favorable results. The TRUST formulation leads to a simple stopping criterion. More detailed introduction to TRUST is presented in Chapter 3.

2.2 Ripple Search Strategy

Based on the aspects of implementation and computational expense, local optimization algorithms are superior to all of the global optimization algorithms. A major disadvantage of these methods is that the local minimum found may be far from the global optimum. This is a consequence of the fact that local optimization algorithms can only get the global

solution when the starting point, which is one of the most important factors for the accuracy of the results, is in the area nearby enough to the global minimum. Unfortunately, in practice the possibility of knowing such a suitable starting point is rather low.

For example, the MATLAB peaks function [36] with two variables is defined by:

$$\begin{aligned} f(\mathbf{X}) &= f(x_1, x_2) \\ &= 3(1 - x_1)^2 e^{(-x_1^2 - (x_2 + 1)^2)} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{(-x_1^2 - x_2^2)} - \frac{1}{3} e^{(-(x_1 + 1)^2 - x_2^2)} \end{aligned} \quad (2.1)$$

It has two local minima at $(-1.3474, 0.2045)$ and $(0.2965, 0.3201)$ respectively and one global minimum at $(0.2283, -1.6255)$. By local optimization algorithms, it is very easy to get trapped in the local minima if the starting point is not near enough to the global solution. Moreover, these algorithms are unable to distinguish the global minimum among the three minima. The plot of the function and its contour is shown in Figure 2.1.

Figure 2.1: The combination mesh and contour plot of the MATLAB peaks function for illustration

To eliminate the drawbacks and at the same time inherit the merits of local optimization algorithms, a new ripple search strategy is proposed. It is the combination of a traditional local optimization method for finding local minima fast and an innovative ripple type search to avoid getting trapped in these local minima. The new global optimization algorithm is presented, analyzed, illustrated and tested in the rest of this chapter.

2.2.1 Algorithm

As defined in Chapter 1, the unconstrained global minimization problem over a feasible region $D \in R^{n_d}$ is defined as follows,

$$\min_{\mathbf{X} \in R^{n_d}} f(\mathbf{X})$$

where, n_d is the number of unknown variables of the objective function $f(\mathbf{X})$.

The framework of ripple search strategy consists of global and local search phases. The global search phase can eliminate the drawback of local optimization algorithms by our novel ripple type search. The ripple type search is realized by generating new trial points located on concentric ripples in the hyperspace to form a global search trajectory that explores the feasible region D and brings the search out of local minima once it gets there. The m -th trial point on the r -th ripple, which is centered at point $\mathbf{X}_0$ in the hyperspace, is denoted as $\mathbf{S}_{rm}$ and generated as follows:

For $m = 1, 2, \dots, m_0 + p_a(r - 1)n_d$ and $r = 1, 2, \dots$, let

$$\mathbf{S}_{rm} = r\Delta r T_{rm} \frac{\mathbf{X}_0}{|\mathbf{X}_0|} + \mathbf{X}_0 \quad (2.2)$$

where,

$$T_{rm} = \prod_{p=1}^{n_d-1} \prod_{q=p+1}^{n_d} U_{pq} \quad (2.3)$$

and the entries of the $n_d \times n_d$ matrix U_{pq} are defined as

$$U_{pq}(p, p) = U_{pq}(q, q) = \cos \theta_{pq} \quad (2.4)$$

$$U_{pq}(p, q) = -U_{pq}(q, p) = \sin \theta_{pq} \quad (2.5)$$

$$U_{pq}(i, i) = 1 \quad (i \neq p, q; \quad i = 1 \dots n_d) \quad (2.6)$$

$$U_{pq}(i, j) = 0 \quad (i, j \neq p, q; \quad i \neq j; \quad i, j = 1 \dots n_d) \quad (2.7)$$

where,

m_0 , p_a and stepsize Δr : predefined fixed scalars. For example, they can be set as $m_0 = 30$, $p_a = 1$ and $\Delta r = 0.5$. The total number of trial points generated on the r -th ripple is $m_0 + p_a(r - 1)n_d$;

$\mathbf{S}_{rm}$: the m -th trial point generated on the r -th ripple;

T_{rm} : the transformation matrix defined by Equations (2.3)-(2.7) for rotation around the point $\mathbf{X}_0$ in the hyperspace;

n_d : the number of variables in the objective function $f(\mathbf{X})$;

U_{pq} : an $n_d \times n_d$ matrix defined by Equations (2.4)-(2.7);

θ_{pq} : a rotation angular variable on the p - q plane, which is formed by the p -axis and the q -axis. θ_{pq} is generated randomly, chosen from a uniform distribution on the interval $[0, 2\pi]$. Note that the total number of angular variables needed for a transformation matrix T_{rm} is $n_d(n_d - 1)/2$.

For example, when $n_d = 2$, T_{rm} can be obtained by:

$$T_{rm} = U_{12} = \begin{bmatrix} \cos \theta_{12} & \sin \theta_{12} \\ -\sin \theta_{12} & \cos \theta_{12} \end{bmatrix};$$

when $n_d = 3$, T_{rm} equals to:

$$T_{rm} = U_{12}U_{13}U_{23} = \begin{bmatrix} \cos \theta_{12} & \sin \theta_{12} & 0 \\ -\sin \theta_{12} & \cos \theta_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_{13} & 0 & \sin \theta_{13} \\ 0 & 1 & 0 \\ -\sin \theta_{13} & 0 & \cos \theta_{13} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_{23} & \sin \theta_{23} \\ 0 & -\sin \theta_{23} & \cos \theta_{23} \end{bmatrix}.$$

The trial points generated by Equations (2.2)-(2.7) are then fed to the local search phase as initial points to find local minima respectively by traditional local optimization methods. Note that from the mathematical point of view, the total number of trial points on the r -th ripple in the hyperspace should be proportional to r^{n_d-1} . However in practice, it is verified through extensive test problems that setting the number as $m_0 + p_a(r - 1)n_d$ is generally proper and sufficient for achieving the global optimum.

The ripple search strategy and its termination criteria are described as below.

Ripple Search Strategy: Set a random initial point $\mathbf{X}_0$ as the center to enlarge the explorative area in the search space by applying Equations (2.2)-(2.7) repeatedly, generating each time an m -th new trial point on the r -th ripple in the hyperspace, denoted by $\mathbf{S}_{rm}$. This process creates many new trial points located on concentric ripples. Start from

each point and employ a proper local optimization algorithm, such as Levenberg-Marquardt method or Quasi-Newton method, to minimize the objective function meanwhile obtain the solution $\mathbf{X}_{rm}$. Record the best one as the final solution $\mathbf{X}^*$, for which the objective function has the smallest value.

Termination Criteria:

Termination Criteria can be any of the following:

- maximum of the number of function evaluations achieved;
- maximum of the number of generated trial points $\mathbf{S}_{rm}$ achieved;
- maximum of the number of ripples achieved;
- an approximate solution is achieved when the objective function value is known.

The existence of local minima and the unsolvability in general are important characteristics of global optimization problems [13]. The unsolvability means that the global solution cannot be guaranteed in a finite time. For the tests on benchmarks in the literature, the true global minimum is always known. In practice, the objective function value of the true global minimum is known to be zero for some applications, such as the supervised training of neural networks. This makes it possible to tell for sure that the global solution has been found or estimate how close the best solution obtained by optimization algorithms is to the true global solution. For a global optimization problem without any prior knowledge on the objective function value of the global minimum, normally the global optimization methods can not identify if the solution obtained is the true global minimum, especially for high-dimensional and complex problems, unless the optimization process is allowed to continue infinitely. There is no means to estimate how close the solution obtained is to the true global minimum. We have to give up the solvability requirement that guarantees to find the global solution in a finite time and only try to obtain an approximation of the true global minimum [13]. In such a case, the best solution with the smallest objective function value obtained when the termination criteria are met, such as a limited time allowed for the optimization process, is regarded as the global minimum. The same situation applies to the ripple search strategy and other proposed global optimization methods to be introduced in the thesis.

This search method is called ripple type search, or ripple search in short, because in the 2D case it acts like ripples to explore the feasible region, as shown in Figure 2.2. When the ripple enters the region which is in the nearby area for a local optimization algorithm to achieve the global minimum, the search will slide to the global minimum. Note that if the objective function is simple enough, the transformation matrix T_{rm} can be set as $T_{rm} = \pm I$ to simplify the optimization procedure.

Ripple search strategy has the following key properties that are superior to some other stochastic global optimization algorithms:

- Though trial points on each ripple are generated stochastically, the ripples spread in a deterministic way. The ripple search differs from those random search methods which explore the feasible region randomly. Hence the proposed method does not suffer from the disadvantages of random search methods, such as the easiness of leading to the same local minimum and the inefficiency of implementation.
- The proposed algorithm requires only three control parameters, m_0 , p_a and Δr . Unlike some stochastic global optimization algorithms, such as simulated annealing and genetic algorithm, having many crucial control parameters, the new algorithm is fairly easy to apply to diverse industrial applications.
- The proposed method has the inherent potential for parallel processing which can improve the efficiency of an algorithm. For example, starting from a single initial point $\mathbf{X}_0$, trial points can be generated and developed on parallel-processing computers simultaneously meanwhile these points together still form the ripples centered at $\mathbf{X}_0$ in the hyperspace. Many existing methods do not have the property, such as the simulated annealing, as the successive trial points are generated by taking into account the results from previous iterations during the minimization. GA and DEA are reported possessing the parallel search nature. However they start from a group of initial individuals and their parallel processing is generally implemented by dividing these individuals into smaller groups that are allowed to reproduce amongst themselves respectively. Furthermore, the descendants have to be produced by their parent individuals.

Figure 2.2: The sketch of ripple search in 2D case

2.2.2 Convergence Analysis

Convergence property of the ripple search strategy is analyzed and discussed as follows. For a global optimization problem with objective function $f(\mathbf{X})$, the region of attraction A_{rm} (or attraction region) for each local minimum $\mathbf{X}_{rm}$ is defined as the set of points in D starting from which the local search will converge to $\mathbf{X}_{rm}$ [13].

For a global optimization problem without any prior knowledge nor investigation on the terrain of the objective function, the local minima can be anywhere in the feasible region D and the location of the global solution is unknown. It is reasonable to assume that the objective function $f(\mathbf{X})$ satisfies the following conditions:

Assumption 2.2.1.

- (1) the feasible region D is spherical with radius d and the attraction region A_{rm} is also spherical with radius L ;
- (2) local minima $\mathbf{X}_{rm}$ are distributed randomly over the feasible region D with a uniform distribution;
- (3) the total number of minima is h for objective function $f(\mathbf{X})$ and the global minimum $\mathbf{X}^*$ is distributed randomly over all minima with a uniform distribution (assume the sole global minimum case).

The first assumption in Assumption 2.2.1 is made for simplicity. When the feasible region

D and the attraction region A_{rm} are not spherical, the convergence analysis is similar with slight modifications on the current definitions of the 2α region and L which are shown in Figure 2.3 for the 2D case.

The aim of global optimization is to find the global minimum among those local minima. Because the probability of a location being with a local minimum is equal in the feasible region, optimization algorithms are very easy to be trapped in these local minima. The second assumption in Assumption 2.2.1 implies one of the challenges in solving global optimization problems. Similarly, the third assumption in Assumption 2.2.1 is also reasonable.

Definition 2.2.1.

Considering initial point $\mathbf{X}_0 = \mathbf{X}_{n+1}(x_{n+1_1}, \dots, x_{n+1_{n_d}})$ and local minimum $\mathbf{X}_{rm}(x_{rm_1}, \dots, x_{rm_{n_d}})$, we define:

Event A: a trial point $\mathbf{S}_{rm}$ generated inside the 2α region, which is illustrated in Figure 2.3 for the 2D case;

Event B: step $r\Delta r$ of trial point $\mathbf{S}_{rm}$ satisfies:

$(\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 - L) < r\Delta r < (\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 + L)$, where $\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2$ is the Euclidean norm of $(\mathbf{X}_{n+1} - \mathbf{X}_{rm})$;

Event C: a trial point $\mathbf{S}_{rm}$ generated inside the attraction region A_{rm} .

The probabilities for events A, B and C are denoted as $p(A) = p_s$, $p(B)$ and $p(C)$ respectively. The number of trial points N on the $\text{Int}(\frac{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}{\Delta r})$ -th ripple, where $\text{Int}(x)$ means taking the integer part of x , is $N = m_0 + p_a(\text{Int}(\frac{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}{\Delta r}) - 1)n_d$ with m_0 , p_a and Δr the control parameters of the ripple search strategy. Suppose the total number of minima for objective function $f(\mathbf{X})$ is h , we have the following Theorem.

Theorem 2.2.1. When stepsize $\Delta r < 2L$, the ripple search strategy converges to the global minimum with probability $P = 1 - (1 - \frac{p_s}{h})^N$.

Proof:

The probability of a trial point $\mathbf{S}_{rm}$ generated inside the attraction region A_{rm} is

$$p(C) \approx p(AB) = p(B)p(A|B) \quad (2.8)$$

Note that as long as stepsize $\Delta r < 2L$, $p(B) = 1$ as the ripple expands. Moreover, event A and event B are independent. Thus Equation (2.8) becomes:

$$p(C) \approx p(B)p(A) = p(A) = p_s \quad (2.9)$$

Since all the events C are identical and independent, the probability of success follows a binomial distribution. The probability of trial points generated inside k attraction regions can be described as:

$$P'_k = \binom{N}{k} p(C)^k (1 - p(C))^{N-k} \approx \binom{N}{k} p_s^k (1 - p_s)^{N-k} \quad (2.10)$$

From (3) of Assumption 2.2.1 the total number of minima is h for the objective function and the global minimum $\mathbf{X}^*$ is distributed randomly over all minima with a uniform distribution. With total N trial points, the probability of k trial points inside the attraction region of the global minimum can be described as:

$$P_k = \binom{N}{k} \left(\frac{p_s}{h}\right)^k \left(1 - \frac{p_s}{h}\right)^{N-k} \quad (2.11)$$

The probability of no trial point generated inside the attraction region of the global minimum therefore is:

$$P_0 = \binom{N}{0} \left(1 - \frac{p_s}{h}\right)^N = \left(1 - \frac{p_s}{h}\right)^N \quad (2.12)$$

thus the probability of having at least one trial point generated inside the attraction region of the global minimum is:

$$P = 1 - \left(1 - \frac{p_s}{h}\right)^N \quad (2.13)$$

That is, the ripple search strategy converges to the global optimum with probability of $P = 1 - \left(1 - \frac{p_s}{h}\right)^N$, regardless of the initial point.

Since $0 < p_s < 1$ (as to be shown in Section 2.5, Appendix A) and $h \geq 1$, $1 - \frac{p_s}{h} < 1$. We have

$$\lim_{N \rightarrow \infty} P = 1 \quad (2.14)$$

Therefore, the ripple search strategy guarantees asymptotic convergence to the global optimum as the number of trial points N increases.

□

Consider a simple example. For the 2D case, let $p_s = 0.0971$, $h = 6$ and P is expected to be 0.98. From Equation (2.13) we require $N = 240$. This instance helps to emphasize the convergence property of the algorithm as it converges within finite time. More detailed information about the probability p_s of a trial point $\mathbf{S}_{rm}$ generated inside the 2α region is given in Section 2.5, Appendix A.

The example shows that, with prior knowledge or investigations on the terrain of the objective function, we can estimate the values of p_s (as to be shown in Section 2.5, Appendix A) and h . Therefore N can be calculated by Equation (2.13). As $N = m_0 + p_a(\text{Int}(\frac{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}{\Delta r}) - 1)n_d$, we can get useful information to set the control parameters m_0 , p_a and Δr of the ripple search strategy. Proper setting of control parameters can help speed up the convergence.

Here is another example. If we decrease N and let $N = 100$, from Equation (2.13) we have $P = 0.8047$. We can improve P by the following method: applying the ripple search strategy independently three times starting from the same initial point. Thus from Equation (2.12) we have

$$P_0 = \left(1 - \frac{p_s}{h}\right)^{3N} = 0.0074$$

and $P = 1 - P_0 = 0.9926$.

Therefore ripple search strategy will converge to the global optimal solution with probability higher than 99% . This example shows that the proposed method has the potential to be implemented simultaneously on parallel-processing computers, which can also help speed up the convergence.

Remark 2.2.1:

Many stochastic global optimization algorithms, such as random search methods and simulated annealing, have the feature that their convergence is guaranteed asymptotically as the number of trial points N increases, similar to Equation (2.14). Their starting points,

Figure 2.3: The sketch of ripple search in the 2D case for convergence analysis

or the trial points generated during iterations, are usually randomly chosen. Therefore it is easy to pick an identical point repeatedly or points in the attraction region of the same local minimum. We call such points as a kind of unpromising points. Therefore lots of previous work are wasted and the convergence speed is low due to the inefficiency of implementation. As can be seen from Theorem 2.2.1 and Equation (2.14), in order to increase the probability of convergence in finite time and to speed up the convergence, it is necessary to avoid taking unpromising points and let more new points to be considered in certain time. Therefore our proposed algorithm is designed to have a different framework from those global optimization algorithms mentioned above. In ripple search strategy, though trial points on each ripple are generated stochastically, the ripples spread in a deterministic way and explore new search region constantly. Hence, the probability of encountering the unpromising points mentioned previously is decreased. Consequently the proposed method, in essence, can consider more new trial points within certain time and is superior to those algorithms mentioned above on the convergence speed.

Theorem 2.2.1 and Equation (2.14) also indicate some other ways to speed up the convergence. For example, developing techniques to identify promising trial points which lead to better solutions than the current best one or decreasing the number of trial points required for convergence to a good solution. These will be considered in the following chapters and new algorithms can be proposed to speed up the convergence process based on the analysis.

Discussions:

Before the ripple search strategy is implemented in practice, discussions on how to determine the control parameters m_0 , p_a and Δr are necessary. To analyze theoretically, Equation (2.13) and the first sample followed show that the number of trial points N required for an expected probability of convergence can be calculated if the probability p_s and the total number of minima h can be estimated reasonably by prior knowledge of the objective function or experience. More detailed information on the probability p_s is to be given in Section 2.5, Appendix A. As $N = m_0 + p_a(\text{Int}(\frac{\|\mathbf{x}_{n+1} - \mathbf{x}_{rm}\|_2}{\Delta r}) - 1)n_d$, we can get useful information on setting the three control parameters. Note that the control parameter m_0 is much less important than the other two.

However, setting some control parameters by empirical study is a more practical way and is generally used in the implementation of global optimization algorithms [26]. The proposed algorithm totally has three control parameters, therefore it is fairly easy to apply to diverse applications compared with other algorithms having many control parameters. Through numerous empirical studies, we suggest the following intervals of all control parameters for general global optimization problems as $m_0 \subseteq [5, 50]$, $p_a \subseteq [0, 6]$ and $\Delta r \subseteq [0.05, 1]$. For simple problems with a few number of local minima, m_0 and p_a can be set small and Δr large, and vice versa for ill-conditioned or complex problems with numerous local minima. More detailed adjustment can take the number of unknown variables of the problem into considerations. Fine-tuning of the control parameters could help speed up the convergence. However, from our empirical investigations on diverse problems, it was found that the proposed algorithm generally can be superior to other methods under comparisons on convergence speed without carefully choosing the control

parameters. This will be shown in the following chapters of the thesis.

2.2.3 Implementation

The implementation of the ripple search strategy is given as follows.

Initialization

Step 1. Generate a random initial point $\mathbf{X}_0$.

Step 2. Start from the point $\mathbf{X}_0$, adopt a proper classical optimization algorithm to minimize the objective function. Get the solution $\mathbf{X}'$ and the objective function value $f(\mathbf{X}')$. Set $\mathbf{X}_{min} = \mathbf{X}'$ and $f_{min} = f(\mathbf{X}')$.

Ripple Search

Step 3. Set the values for m_0 , p_a and Δr . Set the termination criteria.

Step 4. Let $m = 0$, $T_{rm} = T_{r0} = I$ and $\mathbf{S}_{rm} = \mathbf{S}_{r0} = r\Delta r \frac{\mathbf{X}_0}{|\mathbf{X}_0|} + \mathbf{X}_0$. Let $\mathbf{S}_{rm}$ to be the starting point.

Step 5. If $\mathbf{S}_{rm}$ is out of the search region, go to step 7; Otherwise go to step 6.

Step 6. Call the same optimization algorithm as in step 2 to get corresponding solution $\mathbf{X}_{rm}$ and the value of the objective function $f(\mathbf{X}_{rm})$.

If $f(\mathbf{X}_{rm}) < f_{min}$, set $\mathbf{X}_{min} = \mathbf{X}_{rm}$ and $f_{min} = f(\mathbf{X}_{rm})$; Otherwise keep the values of $\mathbf{X}_{min}$ and f_{min} without updating.

Step 7. Increase m . Generate $n_d(n_d - 1)/2$ random rotation angles θ_{pq} for different p and q defined in Equation (2.3). Construct and calculate the transformation matrix T_{rm} by Equations (2.3)-(2.7);

Step 8. Set $\mathbf{S}_{rm} = r\Delta r T_{rm} \frac{\mathbf{X}_0}{|\mathbf{X}_0|} + \mathbf{X}_0$.

Step 9. Let $\mathbf{S}_{rm}$ be the starting point and repeat steps 5 to 9, till $m > m_0 + p_a(r - 1)n_d$ or one of the termination criteria is satisfied.

Step 10. Increase r and repeat steps 4 to 10, till termination criteria is satisfied.

Step 11. Let $\mathbf{X}^* = \mathbf{X}_{min}$. Therefore, $\mathbf{X}^*$ is the final solution of the optimization problem and $f(\mathbf{X}^*)$ is the corresponding objective function value.

2.2.4 Illustration

The MATLAB Peaks function (2.1) is minimized to illustrate the procedure of ripple search strategy, as shown in Figure 2.4. In this example, the local optimization algorithm adopted is the Sequential Quadratic Programming (SQP) method.

Beginning at the initial point $B(0, 2)$, a solution $C(0.0330, 4.0000)$ is obtained by performing the *SQP* method. Because this function is flat in some region and the initial point is relatively far from the global minimum, solution C is neither a global minimum nor a local minimum.

For the proposed method, set $m_0 = 5$, $\Delta r = 0.5$ and $p_a = 2$. By applying the ripple search to enlarge the search area, many solutions are obtained, such as points D and H , and also including the two local minima E and G (denoted by maker '*' in Figure 2.4). Rather than being trapped in the local minima, the global optimum F is obtained during the search and is kept as the final solution. Because the illustrative example is to minimize a simple and low-dimensional function, m_0 is set small and is able to resolve this problem. Simulation results are shown in Table 2.1.

Table 2.1: Solutions of the Ripple Search Strategy

	x_1	x_2	Final Function Value	Number of function evaluations
Global Solution	0.2283	-1.6255	-6.5511	
Starting Point (B)	0	2	5.8591	0
SQP (C)	0.0330	4	0.0012	9
Ripple Search (F)	0.2283	-1.6255	-6.5511	271

Consider another sample. In the thesis, a test function is designed by:

$$f(x_1, x_2) = \frac{400[(x_1 - 3)^2 + (x_2 - 3)^2 - 0.04]}{1 + e^{200[(x_1 - 3)^2 + (x_2 - 3)^2 - 0.04]}} + \frac{x_1^2 + x_2^2 - 4}{1 + e^{200(x_1^2 + x_2^2 - 4)}} \quad (2.15)$$

As shown in Figure 2.5, the above function has a large flat region, a local minimum at $(0, 0)$ in a shallow basin and the global minimum at $(3, 3)$ in a small region with steep slope. This ill-conditioned test function is very difficult for many global optimization methods. It is minimized to illustrate the performance of the proposed method as follows

Figure 2.4: Contour plot of minimization the peaks function by ripple search strategy

and the search process is shown in Figure 2.6. Starting from a random initial point (*), the ripple type search explores the 2D feasible region by generating many new trial points on concentric ripples. The search will be terminated once the total number of generated trial points achieves 200. We set the number at 200 to let the trial points generated by the random search method be not so sparse in the search space, as shown in Figure 2.7 for a comparison, and give the random search method more chances to locate trial points in the small region containing the global solution. Therefore the three control parameters of the ripple type search are set as $m_0 = 6$, $\Delta r = 0.3$ and $p_a = 6$. The value of p_a is intentionally set large to generate this number of trial points while the ripple search explores in the same search space as the random search method. It can be seen from Figure 2.6 that the ripple type search is able to locate two trial points in the small region, thus can find the global solution with the function value of -15.9946 , while the random search method failed using the same number of trial points. Compared with the random search method,

of which the search result is shown in Figure 2.7, the ripple type search is less likely to miss promising regions, or to pick identical points.

Figure 2.8 shows points generated on a single 'ripple' in 3D. Ripple search strategy is general to arbitrary dimensional problems.

Figure 2.5: Plot of the designed function viewed from two different angles.

Figure 2.6: Search result by the ripple type search.

Figure 2.7: Search result by the random search method.

Figure 2.8: Points generated on a single 'ripple' in 3-D.

2.3 Test Functions and Results

Numerical simulations on extensive benchmarks in the literature are presented to demonstrate the performance of the ripple search strategy.

2.3.1 Judge Model

The Judge model proposed by Judge et al. [37] is used as the first test function. This is a nonlinear statistical model as follows,

$$y_t = \theta_1 x_{t1} + \theta_2 x_{t2} + \theta_2^2 x_{t3} + e_t \quad (2.16)$$

where, $t = 1, \dots, 20$. The data used for this example are given in Table 2.2. In the table, $x_{t1} = 1$, x_{t2} and x_{t3} are pseudo-random numbers from a uniform distribution on the unit interval, and y_t are computed by adding a normal pseudo-random number to $\theta_1 + \theta_2 x_{t2} + \theta_2^2 x_{t3}$, where the true variables θ_1 and θ_2 are chosen to be $\theta_1 = \theta_2 = 1$. To determine the least squares estimates of θ_1 and θ_2 , the problem is defined by minimizing the following objective function:

$$f(\Theta) = f(\theta_1, \theta_2) = \sum_{t=1}^{20} (y_t - (\theta_1 + \theta_2 x_{t2} + \theta_2^2 x_{t3}))^2 \quad (2.17)$$

This nonlinear least squares problem is written in minimization form and has two minima. It has two variables and twenty observations. It serves as a convenient test from the literature for optimization algorithms because of the two local minima.

The algorithms are coded in the MATLAB 5.3 environment. Table 2.3 shows the results of the Levenberg-Marquardt method, the simulated annealing and the ripple search strategy respectively for the Judge model. Solutions obtained by other algorithms such as the simplex method and the Quasi-Newton method were presented in [38], which showed that these local optimization algorithms converged to both the local and global minimum. As can be seen in Table 2.3, although the local optimization algorithm Levenberg-Marquardt method requires less execution time, it cannot distinguish between the two minima and is trapped in a nearby local minimum at $(2.4986, -0.9826)$ with the function value of 20.482. On the other hand, both the simulated annealing and the ripple search strategy which combines ripple search with the Levenberg-Marquardt method have no difficulty of achieving the global minimum at $(0.8648, 1.2357)$ with the value of 16.082. Comparing the number of function evaluations, it is obvious that the ripple search strategy is remarkably more efficient than the simulated annealing. Moreover, after combining ripple search with the Levenberg-Marquardt method, the accuracy is improved significantly when compared to using the Levenberg-Marquardt method alone.

Table 2.2: Data for the Judge Model

t	y_t	x_{t1}	x_{t2}	x_{t3}
1	4.284	1.000	0.286	0.645
2	4.149	1.000	0.973	0.585
3	3.877	1.000	0.384	0.310
4	0.533	1.000	0.276	0.058
5	2.211	1.000	0.973	0.455
6	2.389	1.000	0.543	0.779
7	2.145	1.000	0.957	0.259
8	3.231	1.000	0.948	0.202
9	1.998	1.000	0.543	0.028
10	1.379	1.000	0.797	0.099
11	2.106	1.000	0.936	0.142
12	1.428	1.000	0.889	0.296
13	1.011	1.000	0.006	0.175
14	2.179	1.000	0.828	0.180
15	2.858	1.000	0.399	0.842
16	1.388	1.000	0.617	0.039
17	1.651	1.000	0.939	0.103
18	1.593	1.000	0.784	0.620
19	1.046	1.000	0.072	0.158
20	2.152	1.000	0.889	0.704

Table 2.3: Comparison of Three Optimization Algorithms on Judge Model

Method	Solution for Parameters		Optimal Function Value	Number of function evaluations
	θ_1	θ_2		
A Random Starting Point: $\theta_1 = -13.7220$; $\theta_2 = -77.3140$				
Levenberg-Marquardt method	2.4986	-0.9826	20.4823	84
Simulated Annealing	0.8648	1.2357	16.0817	868001
Proposed Method	0.8648	1.2357	16.0817	291

2.3.2 Rosenbrock Model

We have chosen the Rosenbrock classical test function [26] in 4 variables since it is notorious in optimization examples because of the way the curvature bends around the origin. The optimization problem is defined by minimizing the following objective function (2.18):

$$f(\mathbf{X}) = f(x_1, x_2, x_3, x_4) = \sum_{k=1}^3 (100(x_{k+1} - x_k^2)^2 + (1 - x_k)^2) \quad (2.18)$$

This long curved narrow valley function is ill-conditioned and is indeed a hard test for every minimization algorithm since a small change in the variables can give unpredictably

large changes in the value of the objective function.

The tests have been made starting from eleven different initial points. All of the initial points are quite far from the optimum which is obviously $(1, 1, 1, 1)$. The results of the ripple search strategy are reported in Table 2.4, comparing with three global optimization algorithms: adaptive random search, simulated annealing [26] and Torus algorithm[39].

Table 2.4: Comparison of Four Global Optimization Algorithms on Rosenbrock Model with Four Variables

	Starting Point	Adaptive Random Search		Simulated Annealing		Torus four processors		Ripple Search Strategy	
		Number of Function evaluations	Final function value	Number of function evaluations	Final function value	Number of function evaluations	Final function value	Number of function evaluations	Final function value
1	101,101,101,101	519632	1.9E-6	1288001	5.0E-7	9540	3.8E-10	5287	1.07E-9
2	101,101,101,-99	194720	1.7E-6	1328001	1.8E-7	8980	2.4E-10	5371	1.21E-10
3	101,101,-99,-99	183608	3.8E-6	1264001	5.9E-7	9680	6.6E-10	3036	1.56E-10
4	101,-99,-99,-99	195902	2.3E-6	1296001	7.4E-8	9880	1.4E-10	3353	1.21E-10
5	-99,-99,-99,-99	190737	2.7E-6	1304001	3.3E-7	9560	1.8E-10	3181	3.33E-10
6	-99,101,-99,101	4172290	2.6E-6	1280001	2.8E-7	9190	5.7E-10	2979	1.30E-9
7	101,-99,101,-99	53878	3.70	1272001	2.3E-7	10070	4.4E-10	4747	3.30E-10
8	201,0,0,0	209415	1.1E-6	1288001	7.5E-7	9340	2.4E-10	5058	3.29E-10
9	1,201,1,1	215116	1.2E-6	1304001	4.6E-7	9540	1.9E-10	3468	3.30E-10
10	1,1,1,201	29069006	2.2E-6	1272001	5.2E-7	9170	2.5E-9	5482	1.98E-9
11	1010,1010,1010,1010			1384000	3.2E-11			15760	4.94E-10

The accuracy and efficiency of the four algorithms have been measured by the final function value and the number of function evaluations respectively. As can be seen in Table 4, adaptive random search sometimes prematurely converges and is computationally expensive. Ripple search strategy is the most efficient one among the four algorithms and has high accuracy. It has never failed to reach the global minimum of the objective function. Simulated annealing is also relatively accurate but it takes about 300 times more function evaluations than the ripple search strategy. It is worth noting that when the initial point is set to be $(1010, 1010, 1010, 1010)$ which is very far from the optimum, the ripple search strategy remains highly accurate and robust with just an increase of the number

of function evaluations.

2.3.3 A Set of Test Benchmarks

Table 2.5 shows the minimization results of the proposed method compared with standard solutions and several global optimizers on a set of test functions [40, 41, 42]. Brief description of the functions is as below:

- (1) CA: six-hump camel-back function in the search space $-3 \leq x_1 \leq 3$, $-2 \leq x_2 \leq 2$:

$$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

The function has five local and global minima in the search space.

- (2) GP: function Goldstein-Price in the search space $-2 < x_1, x_2 < 2$:

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ * [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

The function has five local and global minima in the minimization region.

- (3) BR: Branin function in the search space $-5 < x_1 \leq 15$, $0 \leq x_2 \leq 15$:

$$f(x_1, x_2) = [x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6]^2 + 10(1 - 1/8\pi)\cos x_1 + 10$$

- (4) H3: function Hart3 in the search space $0 < x_1, x_2, x_3 < 1$:

$$f(x_1, x_2, x_3) = - \sum_{i=1, \dots, 4} C_i * \exp(- \sum_{j=1, \dots, 3} a_{ij} * (x_j - p_{ij})^2)$$

- (5) RA: Rastrigin function in the search space $-2 \leq x_1, x_2 \leq 2$:

$$f(x_1, x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$$

The function has about 50 local minima arranged in the feasible region.

Table 2.5: Results and Comparison on Standard Global Optimization Problems

Method	Objective Function	CA	GP	BR	H3	RA
Standard results	X^*	(0.0898, -0.7126)	(0,-1)	(9.42, 2.475)	(0.115, 0.555, 0.852)	(0,0)
	$f(X^*)$	-1.0316	3.00	0.3979	-3.86	-2
GRDT	X^*	(0.0901, -0.7128)	(0,-1)	(9.4250, 2.4790)	(0.3083, 0.5547, 0.8530)	(0,0)
	$f(X^*)$	-1.0316	3.00	0.3979	-3.8615	-2
Ripple Search Strategy	X^*	(0.0898, -0.7127)	(0,-1)	(9.4248, 2.4750)	(0.1146, 0.5556, 0.8525)	(0,0)
	$f(X^*)$	-1.0316	3.00	0.3979	-3.8628	-2
Comparison on number function evaluations						
Simulated Annealing		10822	5439	2700	3416	
GRDT		290	5175	466	2811	822
ASA						2279
DEA						1450
Ripple Search Strategy		88	156	86	178	103

The results of ripple search strategy are averaged over 100 runs starting from random initial points. It is observed from Table 2.5 that in each case ripple search strategy has no difficulty to reach the global minimum. In the case of functions *CA*, *BR* and *H3*, the proposed method is more accurate than the hybridization of gradient descent algorithm with dynamic tunneling method (GRDT) reported in [41] on finding global optimal point. In term of the number of function evaluations, ripple search strategy is more efficient than simulated annealing, GRDT, ASA and DEA. For the results of ASA, the setting of three most important control parameters is as follows: $TRA = 1e-5$, $TAS = 10000$ and $IT = 1$. For the results of DEA, the setting of three essential control parameters, the population size NP , amplifying differential factor F and crossover constant CR , is 6, 0.9 and 0.35 respectively.

2.4 Conclusions

In this chapter, a novel global optimization algorithm for unconstrained continuous optimization problem is proposed. First, the ripple search strategy, as a stochastic global search method, is presented. The proposed method combines a new ripple search method and a proper local optimization algorithm to find the global optimal solution of a given optimization problem starting from random initial point. Its probabilistic convergence to the global optimum is validated theoretically. Test results on extensive benchmarks demonstrate the performance and robustness of the proposed approach and show that the ripple search strategy is the most accurate and efficient method among those global optimization algorithms under comparisons.

The ripple search strategy possesses the following merits which are of considerable importance:

- (1) The proposed algorithm is an improvement over traditional optimization techniques, and is more efficient and accurate than other global optimization algorithms under comparisons in finding the global optimum. The method can be generally applied to a wide range of problems and has the potential for parallel computing.
- (2) The accuracy of the proposed method is largely independent of the choice of initial values, as verified through tests in this chapter that the proposed method can always obtain the global solution starting from many different initial points. The proposed method works regardless of irregularities of the objective functions. Further, it can avoid the entrapment in local minima and continue to find the global solution by the ripple search.

Note that for any global optimization algorithm, the efficiency of finding the global solution very much depends on the choice of initial point $\mathbf{X}_0$. That is, the computational cost may be different starting from different initial points for the same problem. To eliminate the effect of different initial points on the efficiency, test results of benchmarks in Section 2.3.3 by the proposed method are averaged over 100 runs starting from random initial points for each problem.

- (3) The proposed algorithm is conceptually simple and straightforward to implement. It requires only three control parameters, m_0 , p_a and Δr . Hence, unlike some stochastic global optimization algorithms, the new algorithm is fairly easy to apply to diverse industrial applications. It does not require much statistical background and deep theoretical knowledge of optimization theory for engineers.

2.5 Appendix A. Derivation of the Probability p_s

The derivation of the probability p_s of a trial point $\mathbf{S}_{rm}$ generated inside the 2α region for the convergence analysis of the ripple search strategy is given in this appendix.

The radius L of attraction region A_{rm} can be considered according to a reasonable distribution within $[0, 2d]$, for example, a so-called quasi-normal distribution which is similar to the normal distribution. To simplify the calculation, its probability density function $f_l(l)$ is defined as follow:

Definition 2.5.1.

$$f_l(l) = -\frac{a(l-d)^2 - ad^2}{(l-R)^2 + b} = -\frac{a(l^2 - 2ld)}{(l-R)^2 + b} \quad (2.19)$$

where R is chosen to approximate the most possible value l^* of radius L which is decided by local optimization algorithm and objective function. The constants a and b are selected to satisfy $f_l(l) \geq 0$ and $\int_0^{2d} f_l(l) dl = 1$. Note that the function $f_l(l)$ satisfies $f_l(0) = f_l(2d) = 0$.

The relationship between R and l^* is:

$$\begin{aligned} \frac{df_l(l)}{dl} \Big|_{l=l^*} &= 0 \Rightarrow \\ R &= \begin{cases} \frac{l^*b + dl^{*2} - db}{l^{*2}} = l^* & (l^* = d) \\ \frac{l^{*2} - \sqrt{l^{*4} - 4l^{*2}b - 4l^{*3}d + 8l^*db + 4l^{*2}d^2 - 4d^2b}}{2(l^* - d)} & (l^* \neq d) \end{cases} \end{aligned} \quad (2.20)$$

Obviously there are many other functions to choose, but the one defined by Equation (2.19) is relatively easy for integration. The sketch of the probability density function $f_l(l)$ is shown in Figure 2.9.

From (2) of Assumption 2.2.1 that the local minima $\mathbf{X}_{rm}$ are distributed randomly over D with a uniform distribution, the probability density at $\mathbf{X}_{rm}$ is:

$$f_{\mathbf{x}}(\mathbf{X}_{rm}) = \frac{1}{V(D)} \quad (2.21)$$

where $V(D)$ is the volume of the feasible region D .

Considering initial point $\mathbf{X}_0 = \mathbf{X}_{n+1}(x_{n+1_1}, \dots, x_{n+1_{n_d}})$ and local minimum $\mathbf{X}_{rm}(x_{rm_1}, \dots, x_{rm_{n_d}})$, the angle α as shown in Figure 2.3 can be obtained by:

$$\alpha = \arcsin\left(\frac{L}{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}\right) \quad (2.22)$$

Thus the distribution function $F(u)$ of angle α can be calculated as:

$$\begin{aligned} F(u) &= \int_0^u f_\alpha(\alpha) d\alpha = \int_0^{2d \sin u} F_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2} \left(\frac{l}{\sin u} \right) f_l(l) dl \\ &= \int_0^{2d \sin u} \int_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 \leq \frac{l}{\sin u}} dF_{\mathbf{X}}(\mathbf{X}_{rm}) f_l(l) dl \\ &= \int_0^{2d \sin u} \int_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 \leq \frac{l}{\sin u}} f_{\mathbf{X}}(\mathbf{X}_{rm}) d\mathbf{X}_{rm} f_l(l) dl \\ &= \int_0^{2d \sin u} \int \cdots \int_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 \leq \frac{l}{\sin u}} f_{\mathbf{X}}(\mathbf{X}_{rm}) dx_{rm_1} \cdots dx_{rm_{n_d}} f_l(l) dl \\ &= \frac{1}{V(D)} \int_0^{2d \sin u} \int \cdots \int_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 \leq \frac{l}{\sin u}} dx_{rm_1} \cdots dx_{rm_{n_d}} f_l(l) dl \\ &= \frac{1}{V(D)} \int_0^{2d \sin u} \frac{\left(\frac{l}{\sin u}\right)^{n_d} \cdot \pi^{\frac{n_d}{2}}}{\Gamma\left(\frac{n_d}{2} + 1\right)} f_l(l) dl \\ &= \frac{\pi^{\frac{n_d}{2}}}{V(D)(\sin u)^{n_d} \Gamma\left(\frac{n_d}{2} + 1\right)} \int_0^{2d \sin u} l^{n_d} f_l(l) dl \\ &= -\frac{a\pi^{\frac{n_d}{2}}}{V(D)(\sin u)^{n_d} \Gamma\left(\frac{n_d}{2} + 1\right)} \int_0^{2d \sin u} \frac{(l - 2d)l^{(n_d+1)}}{(l - R)^2 + b} dl \end{aligned} \quad (2.23)$$

where, $F_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}(\frac{l}{\sin u})$ and $F_{\mathbf{X}}(\mathbf{X}_{rm})$ are the distribution functions of $\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2$ and $\mathbf{X}_{rm}$ respectively. $f_{\mathbf{X}}(\mathbf{X}_{rm})$ is the probability density function of $F_{\mathbf{X}}(\mathbf{X}_{rm})$. Note that $\int \cdots \int_{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2 \leq \frac{l}{\sin u}} dx_{rm_1} \cdots dx_{rm_{n_d}}$ is the volume of a hyper sphere with n_d dimensions. $\Gamma(\cdot)$ is the Gamma function.

The expectation of the angle α is:

$$\begin{aligned}
E(u) &= \int_0^{\frac{\pi}{2}} u dF(u) = uF(u)|_0^{\frac{\pi}{2}} - \int_0^{\frac{\pi}{2}} F(u) du \\
&= \left\{ -\frac{a u \pi^{\frac{n_d}{2}}}{V(D)(\sin u)^{n_d} \Gamma(\frac{n_d}{2} + 1)} \int_0^{2d \sin u} \frac{(l-2d)l^{(n_d+1)}}{(l-R)^2 + b} dl \right\} \Big|_0^{\frac{\pi}{2}} \\
&\quad - \int_0^{\frac{\pi}{2}} -\frac{a \pi^{\frac{n_d}{2}}}{V(D)(\sin u)^{n_d} \Gamma(\frac{n_d}{2} + 1)} \int_0^{2d \sin u} \frac{(l-2d)l^{(n_d+1)}}{(l-R)^2 + b} dl du \\
&= \frac{a \pi^{\frac{n_d}{2}}}{V(D) \Gamma(\frac{n_d}{2} + 1)} \int_0^{\frac{\pi}{2}} \int_0^{2d \sin u} \frac{(l-2d)l^{(n_d+1)}}{(\sin u)^{n_d} (l-R)^2 + b} dl du \\
&\quad - \frac{a \pi^{(\frac{n_d}{2} + 1)}}{2V(D) \Gamma(\frac{n_d}{2} + 1)} \int_0^{2d} \frac{(l-2d)l^{(n_d+1)}}{(l-R)^2 + b} dl \\
&= K_1 - \frac{\pi}{2} K_2
\end{aligned} \tag{2.24}$$

with

$$K_1 = \frac{a \pi^{\frac{n_d}{2}}}{V(D) \Gamma(\frac{n_d}{2} + 1)} \int_0^{\frac{\pi}{2}} \int_0^{2d \sin u} \frac{(l-2d)l^{(n_d+1)}}{(\sin u)^{n_d} (l-R)^2 + b} dl du$$

and

$$K_2 = \frac{a \pi^{\frac{n_d}{2}}}{V(D) \Gamma(\frac{n_d}{2} + 1)} \int_0^{2d} \frac{(l-2d)l^{(n_d+1)}}{(l-R)^2 + b} dl$$

Then the probability of a trial point $\mathbf{S}_{rm}$ generated inside the 2α region as shown in Figure 2.3 is:

$$p_s = \frac{E(2u)}{2\pi} = \frac{2K_1 - \pi K_2}{2\pi} \tag{2.25}$$

The detailed expression of p_s could be a useful reference when considering the setting of control parameters of the algorithm.

Consider a 2D example, where $n_d = 2$. We have $\Gamma(2) \approx 1$. Let $a = 0.0267$, $b = 200$, $R=2$ and $d = 25$. Therefore,

$$K_1 = \frac{a}{d^2} \int_0^{\frac{\pi}{2}} \int_0^{2d \sin u} \frac{(l-2d)l^3}{(\sin u)^2 (l-R)^2 + b} dl du = -0.8894$$

and

$$K_2 = \frac{a}{d^2} \int_0^{2d} \frac{(l-2d)l^3}{(l-R)^2 + b} dl = -0.7604$$

We have

$$p_s = \frac{E(2u)}{2\pi} = \frac{2K_1 - \pi K_2}{2\pi} = 0.0971.$$

By Equation (2.13), the value of N can be calculated as shown in the first example. Because $N = m_0 + p_a(\text{Int}(\frac{\|\mathbf{X}_{n+1} - \mathbf{X}_{rm}\|_2}{\Delta r}) - 1)n_d$, we can get useful information to set the control parameters m_0 , p_a and Δr for the ripple search strategy.

Figure 2.9: The sketch of probability density function $f_l(l)$

Chapter 3

Improved Subenergy Tunneling Function and Ripple Search for Efficient Global Optimization

The ripple search strategy presented in Chapter 2 is a good choice to handle global optimization problems because it can decrease the number of unpromising points generated and let more new points to be considered in finite time. However, as it develops all trial points taken in the search space without considering their worth for further investigation, the efficiency of ripple search strategy can be further improved.

In this chapter, a global optimization algorithm is developed from the ripple search strategy. The algorithm is based on a new subenergy tunneling function to identify promising trial points, which will certainly lead to a better solution than previous ones, and the ripple type search to explore the feasible region. The good performance of the proposed algorithm is demonstrated through some frequently used neural network learning applications. The neural networks used are smaller than or equal to the smallest structures found in the literature for each problem under investigation, having the advantages of reduced computational cost of training and better generalization, but making the training task extremely difficult. The influence of noise injected in training signals on the proposed learning algorithm and ways to reduce the noise corruption are also investigated.

Our discussion begins with the existing subenergy tunneling function and our improved

subenergy tunneling function before the mean idea of proposed algorithm is presented.

3.1 Subenergy Tunneling Function

The terminal repeller unconstrained subenergy tunneling (TRUST) is a deterministic global optimization method proposed by Barhen *et al.* [43][44]. The TRUST is an algorithm based on the combination of a novel subenergy tunneling function and the non-Lipschitz terminal repellers. It was reported to be superior to previous tunneling based methods and many other global optimization algorithms [43][45]. The TRUST method has been applied to some optimization applications including neural network training [46]. For the convenience of discussion and comparison, the TRUST algorithm is reviewed as follows [43].

Beginning with the first local minimum at $\mathbf{X}_{k'}^* = \mathbf{X}_0^*$ found by a traditional local search algorithm, the objective function $f(\mathbf{X})$ is transferred to the following virtual objective function:

$$E_{TRUST}(\mathbf{X}, \mathbf{X}_{k'}^*) = E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*) + E_{rep}(\mathbf{X}, \mathbf{X}_{k'}^*) \quad (3.1)$$

where, the subenergy tunneling function $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ is defined as follows:

$$E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*) = \log \left(\frac{1}{1 + \exp(-(f(\mathbf{X}) - f(\mathbf{X}_{k'}^*) + a))} \right) \quad (3.2)$$

and the terminal repeller term $E_{rep}(\mathbf{X}, \mathbf{X}_{k'}^*)$ is

$$E_{rep}(\mathbf{X}, \mathbf{X}_{k'}^*) = -\frac{3}{4}\rho(\mathbf{X} - \mathbf{X}_{k'}^*)^{\frac{4}{3}}u(f(\mathbf{X}) - f(\mathbf{X}_{k'}^*)) \quad (3.3)$$

In the above equations, $\mathbf{X}$ is a vector of n_d variables, $u(\cdot)$ is the Heaviside step function and a is a constant. Typically $a = 2$, as it leads to the most desirable asymptotic behavior of the subenergy tunneling transformation. The parameter $\rho > 0$ is referred to as the power of the repeller. $\mathbf{X}_{k'}^*$ is a fixed vector of $\mathbf{X}$ which can be any one of the total k local minima obtained heretofore or an initial point $\mathbf{X}_0$. In the thesis, if these vectors are put into a $n_d \times k$ matrix one by one, the matrix is $[\mathbf{X}_{k_1}^*, \dots, \mathbf{X}_{k'}^*, \dots, \mathbf{X}_k^*]$ and $E_{sub}(\mathbf{X}_{k_1}^*) > \dots > E_{sub}(\mathbf{X}_{k'}^*) > \dots > E_{sub}(\mathbf{X}_k^*)$. $\mathbf{X}_k^*$ is defined as the local minimum with the current smallest function value, while $\mathbf{X}_{k'}^*$ is the minimum with k' -th smallest

function value. The subenergy tunneling function has the following properties:

$$E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*) \approx \begin{cases} 0, & f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*) \\ f(\mathbf{X}) - f(\mathbf{X}_{k'}^*) + a, & f(\mathbf{X}) < f(\mathbf{X}_{k'}^*) \end{cases} \quad (3.4)$$

$$\frac{\partial E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)}{\partial \mathbf{X}} \approx \begin{cases} 0, & f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*) \\ \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}, & f(\mathbf{X}) < f(\mathbf{X}_{k'}^*) \end{cases} \quad (3.5)$$

Therefore, when $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$, $E_{TRUST}(\mathbf{X}, \mathbf{X}_{k'}^*) = E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*) \approx f(\mathbf{X}) - f(\mathbf{X}_{k'}^*) + a$ is a transformation of $f(\mathbf{X})$ which preserves characteristics relevant for optimization. The TRUST is in the minimization phase to find a local minimum by gradient decent method in the promising region only, where the local minimum exists with smaller function value than $E_{sub}(\mathbf{X}_{k'}^*)$. Whereas $E_{TRUST}(\mathbf{X}, \mathbf{X}_{k'}^*) \approx E_{rep}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and the system $\dot{E}_{TRUST}(\mathbf{X}, \mathbf{X}_{k'}^*)$ switched to the second phase represents a terminal repeller to escape the current local minimum until a new point $\mathbf{X}_{new}$ satisfying $f(\mathbf{X}_{new}) < f(\mathbf{X}_{k'}^*)$ is found. That is, the portions of $f(\mathbf{X})$ where $f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*)$ will be passed through without further exploration. The two phases repeat sequentially before the stopping criteria are met.

3.2 Improved Subenergy Tunneling Function

To achieve expected behavior of the transformation, the subenergy tunneling function should quickly switch between the minimization and repeller phases at the critical value $f(\mathbf{X}) = f(\mathbf{X}_{k'}^*)$ and leave the original function $f(\mathbf{X})$ nearly unmodified for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$ [44]. For the subenergy tunneling function $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ defined by Equation (3.2), the so-called transition region exists and its width is affected by the value of constant a , as shown in Figure 3.1. That is, $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ cannot quickly approach zero for $f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*)$ and $f(\mathbf{X}) - f(\mathbf{X}_{k'}^*) + a$ for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. Therefore a is a crucial factor to the performance of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$. Further more, the original function $f(\mathbf{X})$ in the transition regions is modified asymptotically for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. These deficiencies are unavoidable to any value of a including the recommended value $a = 2$ by the authors of TRUST.

To overcome the above limitations, an improved subenergy tunneling function is designed

as follows:

$$E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*) = \frac{1}{a} \ln \left(\frac{1}{1 + \exp[-a(f(\mathbf{X}) - f(\mathbf{X}_{k'}^*))]} \right) \quad (3.6)$$

The derivative of $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ with respect to $\mathbf{X}$ is:

$$\frac{\partial E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)}{\partial \mathbf{X}} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \left(\frac{1}{1 + \exp[a(f(\mathbf{X}) - f(\mathbf{X}_{k'}^*))]} \right)$$

Because $\exp[a(f(\mathbf{X}) - f(\mathbf{X}_{k'}^*))] > 0$, we have

$$\frac{\partial E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)}{\partial \mathbf{X}} = 0 \Leftrightarrow \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} = 0$$

Therefore $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ is a transformation of $f(\mathbf{X})$ which preserves characteristics relevant for optimization. The improved subenergy tunneling can lead to satisfactory behavior of the transformation, which is described as follows:

$$E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*) = \begin{cases} 0, & f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*) + \epsilon_1 \\ f(\mathbf{X}) - f(\mathbf{X}_{k'}^*), & f(\mathbf{X}) < f(\mathbf{X}_{k'}^*) - \epsilon_2 \end{cases} \quad (3.7)$$

$$\frac{\partial E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)}{\partial \mathbf{X}} = \begin{cases} 0, & f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*) + \epsilon_1 \\ \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}, & f(\mathbf{X}) < f(\mathbf{X}_{k'}^*) - \epsilon_2 \end{cases} \quad (3.8)$$

where, ϵ_1 and ϵ_2 are very small positive scalars and can be ignored for arbitrary big value of a , for example $a = 15$. Note that the sum of $\epsilon_1 + \epsilon_2$ is the width of the transition region.

Figures 3.1 and 3.2 plot the comparisons of the behaviors of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$. The curve with maker '*' shows $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ at $a = 15$, while another is for $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ at $a = 2$ which is the most desirable one suggested by [44].

As shown in Figure 3.1, the transition region of the improved subenergy tunneling function $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ can be ignored thus $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ approaches zero quickly for $f(\mathbf{X}) \geq f(\mathbf{X}_{k'}^*)$ and $f(\mathbf{X}) - f(\mathbf{X}_{k'}^*)$ for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. Any big value of a , for example $a = 15$, can lead to satisfactory behavior of the transformation. Therefore the constant a is no longer a crucial factor to $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$. Figure 3.2 shows that $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*) + f(\mathbf{X}_{k'}^*)$ matches the original function $f(\mathbf{X})$ very well for the whole region of $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. The original subenergy tunneling function $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ does not have the properties

mentioned above. Note that $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*) + f(\mathbf{X}_{k'}^*)$ is unable to keep the original function $f(\mathbf{X})$ unmodified in the transition region when $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. The improvement by the proposed subenergy tunneling function $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ is obvious.

Figure 3.1: Comparison of the behaviors of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$

Figure 3.2: Comparison of $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ and $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ for one-dimensional function $f(x) = [\sin(2x) - x - 1]^2$ [44].

Remark 3.2.1:

As discussed above, the subenergy tunneling function of the original TRUST method has the transition region of which the width is affected by the value of constant a . The original

function $f(\mathbf{X})$ in the transition regions is modified asymptotically for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$ even with the recommended value $a = 2$. However in practice, this will cause a problem only when an attraction region of the objective function is smaller than the corresponding transition region for $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$, making the function value of the minimum changed. For the improved subenergy tunneling function, the transition region can be ignored for any big value of a . $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*) + f(\mathbf{X}_{k'}^*)$ matches the original function $f(\mathbf{X})$ very well in the whole region of $f(\mathbf{X}) < f(\mathbf{X}_{k'}^*)$. On the other hand, this makes the $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ not so smooth as the $E_{sub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ in the transition region. However it does not cause any problems to the minimization process. Because as shown in Figure 3.1, the transition region can be ignored and only the region left to the transition region of $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ is to be optimized, while other regions are excluded from further exploration, as they are already identified as unpromising regions.

3.3 The Algorithm and Implementation

In TRUST, the terminal repellers in the second phase perform well for low dimensional cases. When the number of dimensions for a problem is moderate or large, such as general neural network learning problem, the second phase is complex and computationally expensive. Though two possible strategies for multidimensional case have been presented in [45], the trade-off between efficiency and accuracy exists.

To further advance the applicability of TRUST to high dimensional problems, a stochastic tunneling method, by replacing terminal repellers with the ripple search, is proposed in this chapter. Its framework includes the improved subenergy tunneling function to identify promising points, the local optimization method to develop the identified promising points and the global search algorithm, such as the ripple type search method, to explore the feasible region.

For clarity, the pseudo-algorithm of the proposed method is given as follows:

—Initialization phase

Step 1.

Generate a random starting point $\mathbf{X}_0$.

Step 2.

Set the control parameters m_0 , p_a and Δr for ripple search strategy. Set the termination criteria.

Step 3.

Start from the point $\mathbf{X}_0$, adopt a proper local search algorithm to minimize the objective function. Get the solution $\mathbf{X}_0^*$ and the objective function value $f(\mathbf{X}_0^*)$.

Step 4.

Set $\mathbf{X}_{min} = \mathbf{X}_0^*$, $f_{min} = f(\mathbf{X}_0^*)$. Select the value of k' such that the ripple search and local minimization phases switch at the threshold value $f(\mathbf{X}_{k'}^*)$. For example, $k' = 2$; Set initial threshold value $f(\mathbf{X}_{k'}^*) = f(\mathbf{X}_0^*)$.

Step 5.

Set ripple $r = 1$.

Step 6.

Set $m = 1$.

Step 7.

Transform $f(\mathbf{X})$ into $E_{isub}(\mathbf{X}, \mathbf{X}_{k'}^*)$ by improved subenergy tunneling function in Equation (3.6).

— Ripple search phase

Step 8.

Generate $n_d(n_d - 1)/2$ random values of θ_{pq} for different p and q defined in Equation (2.3). Construct and calculate T_{rm} by Equations (2.3)-(2.7);

Step 9.

Generate a new trial point $\mathbf{S}_{rm}$ by Equation (2.2).

Step 10.

Calculate $E_{isub}(\mathbf{S}_{rm}, \mathbf{X}_{k'}^*)$.

If $(0 - \xi) < E_{isub}(\mathbf{S}_{rm}, \mathbf{X}_{k'}^*) < (0 + \xi)$, where ξ is a small scalar and can be set, e.g. $\xi = 0.2$, go to step 13; otherwise go to step 11.

— Local minimization phase

Step 11.

Let $\mathbf{S}_{rm}$ be the starting point. Apply the same local search method as in step 3 to $f(\mathbf{X})$

and get the corresponding solution $\mathbf{X}_{rm}$ with the value of the objective function $f(\mathbf{X}_{rm})$.

Step 12.

If $f(\mathbf{X}_{rm}) < f_{min}$, set $\mathbf{X}_{min} = \mathbf{X}_{rm}$ and $f_{min} = f(\mathbf{X}_{rm})$, update $\mathbf{X}_{k'}^*$ and $f(\mathbf{X}_{k'}^*)$; otherwise only update $\mathbf{X}_{k'}^*$ and $f(\mathbf{X}_{k'}^*)$.

Step 13.

Increase m by 1 and repeat steps 8 to 13, till $m > m_0 + p_a(r - 1)n_d$ or one of the termination criteria is satisfied.

Step 14.

Increase r by 1 and repeat steps 6 to 14, till the termination criteria is satisfied.

Step 15.

Let $\mathbf{X}^* = \mathbf{X}_{min}$. Therefore, $\mathbf{X}^*$ is the final solution of the optimization problem and $f(\mathbf{X}^*)$ is the corresponding objective function value.

Remark 3.3.1:

As can be seen from Theorem 2.2.1, Equation (2.14) and the analysis in Remark 2.2.1, in order to increase the probability of convergence in finite time and speed up the convergence process, it is necessary to identify promising points worth developing and exclude unpromising ones from consideration to save computational time. The proposed algorithm is designed right for this purpose and therefore it can serve as an efficient algorithm in global optimization. The proposed method has four control parameters, k' , m_0 , Δr and p_a , and is easy to be applied to diverse applications compared with some other methods which have many critical control parameters. To verify its performance, we will apply the proposed algorithm to neural network applications in the following sections and make comparisons with other methods. The suggested ranges of the control parameters for general neural network training problems are $k' \subseteq [1, 20]$, $m_0 \subseteq [10, 50]$, $\Delta r \subseteq [0.05, 0.5]$ and $p_a \subseteq [1, 3]$. For simple problems using large neural networks, k' , m_0 and p_a can be set small and Δr large, and vice versa for complex problems using small-sized neural networks.

3.4 Neural Networks and the Feedforward Neural-Network Learning Problems

3.4.1 Artificial Neural Networks

Artificial neural networks, or neural networks (NNs) for short in this thesis, have been widely used in a variety of signal processing applications, such as prediction, pattern recognition, image processing and mobile communication system [47]-[49]. The concept of neural networks was inspired by biological neural networks. Biological neurons, believed to be the structural constituents of the brain, are much slower than silicon logic gates. The brain compensates for the relatively slower operation of biological neurons by having an enormous number of them that are massively interconnected. The result is that inferencing in biological neural networks is much faster than the fastest computer.

NNs attempt to mimic some characteristics of biological neural networks. In NNs, information is stored in the synaptic connections. Each neuron is an elementary processor with primitive operations, like summing the weighted inputs and then amplifying or thresholding the sum.

Figure 3.3 (left) shows an artificial neuron. This neuron computes a weighted sum of n input signals x_i , $i = 1, \dots, n$, and passes it through an activation function $\varphi(\cdot)$ to produce output y :

$$y = \varphi\left(\sum_{i=1}^n w_i x_i - \sigma\right) \quad (3.9)$$

where w_i is the synapse weight associated with the i th input. The activation function $\varphi(\cdot)$ can take the forms of threshold, piecewise linear, sigmoid or Gaussian functions.

A neural network is characterized by the network topology, the connection strength between pairs of neurons (weights), the node properties, and the state-updating rules. The updating rules control the states of the processing elements (neurons).

According to their architectures, neural networks can be classified into feedforward and feedback networks. In feedforward neural networks, connections between nodes (neurons) are uni-directional, that is, from the input nodes to the output nodes through possibly

Figure 3.3: Model of a neuron (left) and a three-layer feedforward neural network (right).

several layers of the hidden nodes. The most popular family of feedforward networks are the multilayer perceptrons, in which neurons are organized into layers that have unidirectional connections between them. Figure 3.3 (right) shows a typical structure of this type of network. On the other hand, in feedback neural networks, there are connections in the reverse direction, i.e., from the output nodes to the input nodes. These feedback connections feed information back from output nodes to input or hidden nodes.

An alternative classification of neural networks is based on learning paradigms, by which neural networks are classified into supervised-, reinforcement-, and unsupervised-learning networks. In supervised learning, the target outputs are known ahead of time, and the error between the desired outputs and the actual outputs during learning is fed back to the network to adjust its weights. Reinforcement learning differs in that its feedback only distinguishes between correct and incorrect outputs but not the amount of error. In unsupervised learning, no target output is supplied during training, and the network self-organizes and classifies the training data on its own.

Learning in neural network is accomplished by adjusting weights between connections in response to the inputs of training patterns. In feedforward neural networks, this involves mapping from an input space to an output space. Despite various forms of error functions, such as mean-square error, arctangent error, and cross-entropy error, training a neural network is in fact a global optimization problem. Getting trapped in local minima is a common problem in neural network learning and current learning methods are far from

optimal.

3.4.2 Supervised Training in Neural Networks

Supervised training in neural networks can be formulated as an unconstrained nonlinear optimization problem of minimizing the objective function which is usually defined by the total sum of squared-errors (TSSE) of the output over all training patterns, that is:

$$E(\mathbf{w}^*) = \min_{\mathbf{w} \in R^{n_d}} E(\mathbf{w}) = \min_{\mathbf{w} \in R^{n_d}} \sum_{i=1}^n (o_i(\mathbf{w}) - d_i)^2 \quad (3.10)$$

where $o_i(\mathbf{w})$ and d_i are the actual and desired outputs of the network for the i^{th} pattern respectively, n is the number of training patterns, $\mathbf{w}$ is a vector of weights of the neural network and n_d is the number of weights. The task is to find the appropriate values of connection weights $\mathbf{w}^*$ so that the given objective function (3.10) is minimized. The search region is the weight space. Training the weights in neural networks is a hard optimization problem because the highly nonconvex objective function usually contains numerous local minima [50]. Global optimization algorithms are good alternatives to traditional local search methods, such as the backpropagation (BP) algorithm, since the latter cannot avoid getting stuck at local minima.

Efficient and accurate training of neural networks is an ongoing research subject of considerable interest and a number of methods have been proposed. These approaches can be divided into two broad categories. One is gradient-based methods (traditional methods) [51]-[56], the other is global optimization algorithms [46][50][57]-[61].

Traditional gradient-based methods, such as the backpropagation (BP) algorithm, can easily get stuck at local minima. To overcome this shortcoming, they normally adopt large neural network architectures which may cause a training process to be excessively long. An enormous amount of effort has been made to accelerate the training speed by traditional methods, for instance, the modified standard backpropagation (MBP) algorithm which minimizes a modified form of the criterion used in the standard backpropagation algorithm [51], the trust-region-based error-aggregated training algorithm (TREAT) that uses a different Hessian matrix approximation to reduce the iterative computational cost [52],

the Levenberg-Marquardt backpropagation [53] and the efficient second-order algorithms like LMAM and OLMAM [54]. In general, gradient-based methods do not aim at very high accuracy on results and work well for simple applications. However, when the problems encountered are complex and large networks are needed for gradient-based methods to obtain a solution within required accuracy, the disadvantage mentioned above is difficult to overcome. Furthermore, the optimal network size that can guarantee a good solution within reasonable training time is hard to determine in practice.

Global optimization algorithms are a good alternative to traditional methods. They attempt to use smaller-sized neural networks and obtain higher accuracy of solutions than traditional methods do. Because there are fewer weights to adjust, smaller size neural networks are not so costly in training compared with larger ones and their generalization ability is better [50]. Moreover, the decision of optimal network size is no longer so crucial. However, the landscape of the objective function for a small network can be very rugged and ill-conditioned, such as having large flat regions, shallow basins and steep slopes. Together with the reduced number of global minima, they make the training of small-sized neural networks extremely difficult. Therefore research results on small neural network training are rare in the literature and many global optimization algorithms still work on general-sized neural networks for many applications [46][57]-[61]. However this topic has attracted more and more attentions [50].

There are two kinds of global optimization algorithms which can be used for neural network training. One is deterministic methods, for example, the NOVEL [50] and the TRUST methods [43]-[46]. Another is stochastic methods, such as random search methods, simulated annealing and genetic algorithm [26][34][62][63]. Deterministic methods can guarantee absolute success in theory, but only by making certain restrictive assumptions on the objective functions. When applied to neural network training which generally is a high dimensional problem, deterministic methods are complex and computationally expensive. Usually to implement this kind of algorithms, the trade-off between efficiency and accuracy exists. In contrast, stochastic methods assume little knowledge on the objective functions and are often simple to implement. Therefore, they can be adopted to handle large-scale and complex problems without much difficulty. However, some stochas-

tic methods, such as the random search method, are inefficient in the way they explore the weight space. They may easily miss promising regions which have better solutions than the previous ones or lead to the same local minimum, resulting in failure in finding the global solution or waste of previous local searches. Some stochastic algorithms, for example, simulated annealing based methods, have many crucial control parameters to choose carefully for different problems. Therefore it is not easy for them to get satisfactory solutions for diverse applications.

Real-world training data obtained are usually contaminated by noises. Because of their intrinsic globality, global learning algorithms may perfectly fit the given data with noises, leading to the phenomenon of overfitting. An example can be seen in Figure 3.9(a). In overfitting situations, the learning may be able to achieve low training errors because the learned neural network has tried to fit all data as close as possible. However, the testing errors may significantly increase because the testing patterns are different with the training patterns. Various robust learning algorithms have been proposed to resolve the problems with noises [66][67]. The basic idea is to replace the squared-errors term in the objective error function by the so-called loss function. These reported algorithms can improve the learning performance to some extent when training data sets contain noises. In the next section, improved subenergy tunneling function and ripple search strategy is applied to NN training problems, with particular emphasis on small-sized neural networks. Additionally, the effect of noises on the proposed algorithm and ways to improve the training and generalization results are also investigated.

3.5 Neural Network Applications

The proposed method is tested and applied to several neural network benchmarks. In all experiments, Levenberg-Marquardt backpropagation [53] is adopted as the traditional method in the local search phase. The topologies of neural networks chosen are smaller than or equal to the smallest structure found in the literature for each problem making training tasks difficult. All the simulation results of the proposed method are generated using the neural network toolbox of MATLAB 6.5 [53] on a PC with 2.4GHz clock speed.

In tables, the training and generalization errors are the total sum of squared-errors defined by Equation (3.10). Test results of the proposed method are carried out without fine-tuning the control parameters. It is worth noting that different traditional methods with different settings may lead to different number of training epochs and CPU time.

3.5.1 XOR Problem

In this test four samples of the XOR input-output patterns are used for training. Following [46][61], the network chosen is a feedforward neural network with 2 inputs, 2 neurons in the hidden layer and 1 neuron in the output layer, denoted as a $2 - 2 - 1$ network. The total number of weights including bias is 9.

For the proposed method, $k' = 2$, $m_0 = 30$, $\Delta r = 0.5$ and $pa = 1$. The allowed maximal number of training epochs is 500. The number of Monte-Carlo runs is 100.

Table 3.1 shows the output values by the proposed method after training, and the comparison with the TRUST method [46] and two other global descent algorithms (FLGO and LSGO) [61] under the same test conditions. The target sum of squared-errors is first set to be 0.025. As the accuracy is not sufficient to obtain a global solution, note the target values in the table, it is then set to be $1e - 9$. As observed from the number of epochs, the proposed method is the most accurate and efficient one among those under comparisons and is much faster than the TRUST method, where one epoch is typically defined as a single presentation of all input vectors to the network and a flop means a floating point operation. In addition, the proposed method also performs better than the recursive least squares (RLS) method which needs around 60 training epochs to converge globally [55].

3.5.2 Function Approximation

The function in this simulation is given by

$$h(x) = \sin(2\pi x) \cos(4\pi x), x \in [0, 1] \quad (3.11)$$

Table 3.1: Results for the XOR Problem

Pattern #	x_1	x_2	Target	TRUST	Proposed Method		Fast Deterministic Method	
							FLGO	LSGO
1	0.0	0.0	0.0	0.030	0.0004	0.0000	N/A	
2	0.0	1.0	1.0	0.963	0.9862	1.0000	N/A	
3	1.0	0.0	1.0	0.964	0.9806	1.0000	N/A	
4	1.0	1.0	0.0	0.031	0.0219	0.0000	N/A	
Target sum of squared error				N/A	0.025	1e-9	0.025	
Success Rate				N/A	100%	100%	100%	100%
Mean of epochs				>800	8.3500	11.2857	26.23	39.40
Deviation of epochs				N/A	4.8646	5.8008	4.12	15.76
Minimum epochs				N/A	3	5	19	28
Maximum epochs				N/A	31	33	47	132
Mean of point numbers				N/A	30.9100	32.5408	N/A	N/A
Deviation of point numbers				N/A	68.2324	56.4493	N/A	N/A
Mean of Error E(w)				0.0044	0.0049	3.8824e-011	<0.025	<0.025
Deviation of Error E(w)				N/A	0.0067	1.1844e-010	N/A	N/A
Mean of flops				N/A	3.5140e+005	3.8089e+005	N/A	N/A
Deviation of flops				N/A	5.4908e+005	4.3629e+005	N/A	N/A

Noiseless Case

The first network chosen is a $1 - 7 - 1$ feedforward neural network. The total number of weights including biases is 22, the same as in [52]. The activation functions are hyperbolic tangent sigmoid for the hidden layer and linear transfer function for the output layer. The 50 input-output training patterns are uniformly chosen in the interval $[0, 1]$. To examine the performance of the proposed method, 50 new samples are randomly picked and tested after training. The number of Monte-Carlo simulation runs for the test is set as 100 using different initial weights generated randomly with uniform distribution on the range $[-1, 1]$, the same as in [52]. For the proposed method, $k' = 2$, $m_0 = 30$, $\Delta r = 0.5$ and $p_a = 1$. The maximal allowed number of training epochs is 1000. According to [52], the target total sum of the squared-errors is first set to be 0.1. Training and testing results of one trial using the proposed method are plotted in Figure 3.4. For this simulation in figures, the solid curve is the original function, points (*) are training results and points (o) are testing results. As the accuracy is not sufficient to obtain a global solution, observed from

Figure 3.4, it is then set to be $1e-2$ and $1e-3$ respectively. Training and testing results of one trial using the proposed method at target accuracy of $1e-3$ are plotted in Figure 3.5.

Figure 3.4: Result plot of the proposed method: 1 – 7 – 1 neural network, target accuracy is 0.1. (a) Training result at the function value of 0.0807 ; (b) Testing result at the function value of 0.0760.

Figure 3.5: Result plot of the proposed method: 1 – 7 – 1 neural network, target accuracy is $1e-3$. (a) Training result at the function value of $9.7253e-004$; (b) Testing result at the function value of 0.0010.

Table 3.2 shows the detailed simulation results compared with other training methods. It

can be seen from the table that the proposed method can achieve all required accuracies. Our method has higher success rate and less training epochs than the TREAT and EBPM methods [52] under the same test conditions.

Table 3.2: Results for the Function Approximation Problem Using 50 Training Patterns

Neural Network Structure 1-7-1; Number of Training Patterns: 50					
	Proposed Method			TREAT	EBPM
Target sum of the squared-errors	1e-1	1e-2	1e-3	1e-1	
Success Rate	100%	100%	100%	99%	62%
Mean of epochs	92.75	127.85	165.7	97.72	3173.4
Deviation of epochs	52.9515	115.8991	78.1351	32.85	1939.4
Minimum epochs	46	55	90	N/A	N/A
Maximum epochs	412	994	785	N/A	N/A
Mean of training error	0.0898	0.0093	9.7715e-4	N/A	N/A
Deviation of training error	0.0112	9.4707e-4	2.3354e-5	N/A	N/A
Mean of generalization Error	0.0797	0.0083	9.8951e-4	0.0868	0.4705
Deviation of generalization error	0.016	0.0015	1.1540e-4	0.0133	0.0976
Mean of CPU time (seconds)	5.7339	6.1207	6.6883	N/A	N/A
Deviation of CPU time	3.5443	3.6826	4.3521	N/A	N/A

The second network used for comparison is a $1-10-1$ feedforward neural network with 31 weights including biases. The number of training patterns uniformly chosen in the interval $[0, 1]$ is 20. Another 20 new samples are randomly picked and tested after training. The number of Monte-Carlo simulation runs for the test is set as 100 using different initial weights generated randomly with uniform distribution on the range $[0, 1]$, the same as in [59][61]. For the proposed method, $k' = 2$, $m_0 = 30$, $\Delta r = 0.5$ and $p_a = 1$. According to [59][61], the target total sum of the squared-errors is first set to be 0.025. Training and testing results of one trial using the proposed method are plotted in Figure 3.6. As the accuracy is not sufficient enough to obtain a global solution, observed from the figure, it is then set to be $1e-1$ and $1e-3$. Training and testing results of one trial using the proposed method at target accuracy of $1e-3$ are plotted in Figure 3.7.

As shown in Table 3.3, the proposed method can achieve all required accuracies. The computational cost of epochs of the proposed method is between FLGO and LSGO under the same test conditions [61]. The maximum number of epochs of the proposed method is

Figure 3.6: Result plot of the proposed method: 1–10–1 neural network, target accuracy is 0.025. (a) Training result at the function value of 0.0231; (b) Testing result at the function value of 0.0200.

Figure 3.7: Result plot of the proposed method: 1–10–1 neural network, target accuracy is $1e-3$. (a) Training result at function value of $4.4695e-4$; (b) Testing result at function value of $4.4106e-4$.

much smaller than that of FLGO. Note that the experiment results of FLGO and LSGO were carried out on a different computer hardware from ours, therefore the CPU execution times are incomparable.

The proposed method has also been tested successfully on the same problems mentioned above using 1 – 5 – 1 feedforward network with 16 weights and biases. Detailed results are shown in Table 3.7. Compared with those results in Tables 3.2 and 3.3, the number of epochs increases indicating that the problems are more difficult. However, the amount of CPU time on the same PC is reduced because the network sizes are smaller.

Table 3.3: Results for the Function Approximation Problem Using 20 Training Patterns

Neural Network Structure 1-10-1; Number of Training Patterns: 20					
	Proposed Method			FLGO	LSGO
Target sum of the squared-errors	1e-1	0.025	1e-3	0.025	
Success Rate	100%	100%	100%	100%	100%
Mean of epochs	80.97	94.09	123.5	108.88	71.96
Deviation of epochs	39.0644	41.2614	47.2182	41.48	23.68
Minimum epochs	42	49	58	64	27
Maximum epochs	348	357	433	1460	192
Mean of training error	0.0821	0.0199	9.3142e-4	<0.025	<0.025
Deviation of training error	0.0151	0.0055	1.0154e-4	N/A	N/A
Mean of generalization Error	0.082	0.0165	0.001	N/A	N/A
Deviation of generalization error	0.0379	0.0073	2.4501e-4	N/A	N/A
Mean of CPU time (seconds)	6.4218	6.8753	7.7652	11.8679	8.1315
Deviation of CPU time	3.3354	3.7722	4.513	N/A	N/A

Noisy Case

Many obtained training data are subject to noise. When the noise is small, it may not always have a big influence. When the noise becomes large, the global learning algorithms may try to fit those corrupted data and thus the phenomenon of overfitting occurs. In the following, the influences of noise injected in training data on the proposed learning algorithm and the improvement by using a different objective function and by so-called limiting the training process are investigated.

For the same function approximation problem in Equation (3.11), noisy training data are generated by adding Gaussian white noise with zeros-mean and with three different standard deviations 0.05, 0.15 and 0.25, respectively. For the noise with standard deviation 0.25, overfitting is frequently observed after training by the proposed method, as shown

in Figures 3.8(a) and 3.9(a), where the dotted curve is the original function, the solid curve displays the function approximated by the trained neural network, and points (+) are training patterns with noise.

To eliminate the phenomenon of overfitting, the sum of logistic loss function [66] is introduced to replace the sum of squared-errors objective function as follows:

$$E(\mathbf{w}) = \sum_{i=1}^n \frac{\beta}{2} \ln \left[1 + \left(\frac{(o_i(\mathbf{w}) - d_i)^2}{\beta} \right) \right] \quad (3.12)$$

where the scale estimator β is defined by

$$\beta = 1.4826 \times \text{mean}(|(o_i(\mathbf{w}) - d_i) - \text{mean}(o_i(\mathbf{w}) - d_i)|) \quad (3.13)$$

Detailed results of the noise case for 100 trials are shown in Tables 3.4 and 3.5. From a statistical point of view, little improvement is obtained by changing the objective error function to the sum of loss function, especially for the 1 – 10 – 1 neural network using 20 training data, which may due to that the number of training data is too less. However, it is found that limiting the training process, that is, moderately terminating each local minimization during the whole training process, is also helpful to reduce the noise corruption. As can be seen from Figures 3.8(b) and 3.9(b), the overfitting does not occur under the absolutely same test and training conditions by applying the new objective function and terminating each gradient descent in advance during the training process.

3.5.3 The 4-bit and 10-bit Parity Problems

The parity problem is essentially a generalization of the XOR problem. The single output unit is required to be on if an odd number of inputs are on, and off otherwise. It is often used for evaluating network performances and considered as a challenging problem.

The proposed method is first tested on the 4-bit parity problem. Following [46][59], the network chosen is a 4 – 4 – 1 feedforward neural network with 25 weights including biases. The training set has 16 input-target patterns. For the proposed method, $k' = 2$, $m_0 = 30$, $\Delta r = 0.35$ and $p_a = 1$. The number of Monte-Carlo simulation runs for the test is set as 100 using different initial weights generated randomly with uniform distribution on the

Figure 3.8: Results of using different objective functions and training process for 1 – 7 – 1 neural network with 50 training data corrupted by Gaussian noise with standard deviation of 0.25: (a) The sum of squared-errors objective function; (b) The sum of logistic loss function and limiting the training process.

Figure 3.9: Results of using different objective functions and training process for 1 – 10 – 1 neural network with 20 training data corrupted by Gaussian noise with standard deviation of 0.25: (a) The sum of squared-errors objective function; (b) The sum of logistic loss function and limiting the training process.

Table 3.4: Results for the Function Approximation Problem Using 50 Noisy Training Patterns

Neural Network Structure 1-7-1; Number of Training Patterns with noise: 50				
Objective function	Sum of squared-errors			Sum of logistic loss function
Standard deviation of noise	0.05	0.15	0.25	0.25
Success Rate	100%	100%	100%	100%
Mean of epochs	362.6400	213.6400	249.9800	376.1616
Deviation of epochs	440.3725	251.4323	267.3573	328.4068
Minimum epochs	58	41	37	69
Maximum epochs	1601	1190	1600	1600
Mean of training error	0.0951	0.8897	2.6164	2.3376
Deviation of training error	0.0137	0.1663	2.0011	1.0790
Mean of generalization Error	0.0466	0.4103	1.2116	1.0946
Deviation of generalization error	0.0194	0.2378	1.3943	0.7585
Mean of flops	7.6527e+007	6.4051e+007	8.5262e+007	1.5049e+008
Deviation of flops	5.8006e+007	4.1397e+007	4.2089e+007	2.7846e+007

Table 3.5: Results for the Function Approximation Problem Using 20 Noisy Training Patterns

Neural Network Structure 1-10-1; Number of Training Patterns with noise: 20				
Objective function	Sum of squared-errors			Sum of logistic loss function
Standard deviation of noise	0.05	0.15	0.25	0.25
Success rate	100%	100%	100%	100%
Mean of epochs	917.2929	608.8469	474.5918	420.1010
Deviation of epochs	545.3109	454.8502	462.6889	434.8998
Minimum epochs	131	102	54	17
Maximum epochs	1604	1602	1604	1616
Mean of training error	0.0118	0.1107	0.6588	0.7049
Deviation of training error	0.0218	0.2318	0.8737	0.8948
Mean of generalization Error	0.0988	0.5559	1.3955	1.3252
Deviation of generalization error	0.4199	0.6633	1.1949	1.1675
Mean of flops	6.0962e+008	6.0657e+008	6.0812e+008	5.9750e+008
Deviation of flops	1.5384e+008	1.2069e+008	8.4715e+007	9.5523e+007

range $[-5, 5]$, the same as in [51]. The maximum allowed number of training epochs is 500.

Table 3.6 shows the performance comparison among the TRUST [46], LSGO [59], the MBP algorithm [51] and the proposed approach. The target sum of squared-errors is first set to be 0.025. As the accuracy is not sufficient to obtain a global solution, observed

from the target values after training, it is then set to be $1e-7$. The proposed method can achieve both required accuracies. The TRUST method uses more than 30,000 epochs while the LSGO method is slightly faster than our method on this problem using initial weights on the range $[0, 1]$. The MBP training algorithm, which is a traditional gradient-based method and developed to reduce the computational cost, has to use a $4 - 8 - 1$ feedforward neural network to achieve a comparable precision and costs more epochs than the proposed method.

It was reported that with a feedforward network and no shortcut connections, there must be at least N hidden units for a network with N inputs on the parity problem using backpropagation [64]. The proposed method has been tested to resolve the 4-bit parity problem using the $4 - 3 - 1$ feedforward network with 19 weights and biases. The success rate of obtaining global solution keeps unchanged over 100 runs. Detailed results are shown in Table 3.7.

Table 3.6: Comparison Results for the 4-Bit Parity Problem

	Proposed Method		TRUST	LSGO	MBP	
Target sum of squared-errors	0.025	1e-7	N/A	0.025	N/A	
Success rate	100%	100%	N/A	100%	N/A	N/A
Mean of epochs	56.08	66.21	>30000	39.82	130	>300
Deviation of epochs	47.5734	53.1815	N/A	24.45	N/A	N/A
Minimum epochs	9	16	N/A	21	N/A	N/A
Maximum epochs	300	273	N/A	179	N/A	N/A
Mean of Error	0.007	1.9757e-8	0.0088	<0.025	1.6e-2	1.6e-6
Deviation of Error	0.0077	2.8665e-8	N/A	N/A	N/A	N/A
Mean of CPU time (seconds)	9.9349	14.6476	N/A	4.14	N/A	N/A
Deviation of CPU time	7.4737	11.6779	N/A	N/A	N/A	N/A

The proposed method is also applied to the 10-bit parity problem, where there are 1024 training patterns and no test patterns. Following [50], the networks chosen are $10 - 5 - 1$ and $10 - 6 - 1$ feedforward neural networks with 61 and 73 weights including biases respectively. For the proposed method, $k' = 10$, $m_0 = 30$, $\Delta r = 0.3$ and $p_a = 1$. The initial weight is the origin, the same as in [50]. The small number of hidden units makes the problem extremely challenging.

Table 3.7: Results for Problems with Smaller Network Structures

Application Problems	Function Approximation		4-Parity
	50 Training patterns	20 Training patterns	
Neural Network Structure	1-5-1	1-5-1	4-3-1
Target sum of the squared-errors	1e-3	1e-3	1e-6
Success rate	100%	100%	100%
Mean of epochs	222.9184	200.27	106.8352
Deviation of epochs	134.544	138.6143	76.7901
Minimum epochs	89	73	22
Maximum epochs	757	761	444
Mean of training error	9.8546e-4	9.7336e-4	4.2714e-7
Deviation of training error	1.4512e-5	1.8835e-5	1.5307e-8
Mean of generalization Error	0.001	0.0011	N/A
Deviation of generalization error	1.0409e-4	2.1072e-4	N/A
Mean of CPU time (seconds)	3.4267	1.8981	11.7232
Deviation of CPU time	2.1417	1.4712	9.5464

Table 3.8 shows the performance comparison among the Truncated Newton's method with multistarts (TN-MS) [50], NOVEL [50] and the proposed approach. The experiment results of TN-MS and NOVEL were carried out on different computer hardware and software platform by adopting different traditional methods in the phase of local search from ours, therefore the CPU execution times are incomparable. However the number of restarts of TN-MS and the number of time units of NOVEL have similar meaning and are useful in providing a measure on relative computational complexity among the algorithms. Note that the NOVEL executed all three stages in its global-search phase in each time unit [50], thus the number of restarts is three times of the number of time units. It can be seen from the table that our method is able to achieve higher training corrections than the TN-MS and NOVEL methods [50] with less or similar computational costs. For the proposed method, the CPU time is basically spent on the local search phase by the traditional method and the computational costs of generating trial points by ripple type search can be ignored, especially for high-speed computers.

Table 3.8: Comparison Results for the 10-Bit Parity Problem

10-Parity Problem	No. of Hidden Units					
	5			6		
Results \ Methods	Proposed Method	TN-MS	NOVEL	Proposed Method	TN-MS	NOVEL
Training correction (%)	99.9	97.2	98.9	99.9	97.6	99.8
No. of restarts	78	148	153	150	108	186
CPU time (seconds)	87.7503	2000	2000	118.6086	3000	3000
No. of epochs	2195	N/A	N/A	2465	N/A	N/A

3.5.4 Sonar Problem

Sonar problem is the task to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock [65]. The problem has sixty real-valued inputs and one binary output.

The proposed method is tested on the sonar problem. Following [50], the networks chosen are $60 - 2 - 1$ and $60 - 3 - 1$ feedforward neural networks with 125 and 187 weights including biases respectively. For the proposed method, $k' = 20$, $m_0 = 40$, $\Delta r = 0.4$ and $p_a = 2$. The initial weight is the origin, the same as in [50].

Table 3.9 shows the performance comparison among the Truncated Newton's method with multistarts (TN-MS) [50], NOVEL [50] and the proposed approach. The number of restarts is useful in providing a measure of relative computational complexity among the algorithms. The proposed method is much faster than the TN-MS in getting the same or better results. For the sonar problem using $60 - 3 - 1$ neural network, our method can get the same corrections as NOVEL and the number of restarts is 412.

3.5.5 Two-spiral Problem

The two-spiral problem is to learn to discriminate between two sets of training points which lie on two distinct spirals in the x-y plane. These spirals coil a few times around the origin and around one another. In order to classify the two spirals, Lang and Witbrock [68] crafted a $2 - 5 - 5 - 1$ neural network with shortcuts. The total number of weights

Table 3.9: Comparison Results for Sonar Problem

Sonar Problem	No. of Hidden Units					
	2			3		
Results \ Methods	Proposed Method	TN-MS	NOVEL	Proposed Method	TN-MS	NOVEL
Training correction (%)	98.1	98.1	98.1	100	100	100
Testing correction (%)	91.3	90.4	94.2	91.3	91.3	92.3
No. of restarts	153	454	573	117	485	873
No. of epochs	1.4314e6	N/A	N/A	1.0561e6	N/A	N/A

is 138. Baum and Lang [69] tried to solve this problem using a $2 - 50 - 1$ conventional BP network, and did not get the required result. The Cascade-Correlation algorithm of Fahlman et.al. [70] was able to solve the problem with networks comprising 12 to 19 hidden nodes with shortcuts. The LMAM and OLMAM algorithms proposed by Ampazis et al. [54] resolved the problem using a $2 - 30 - 1$ feedforward network at success rate of 89% and 90% over 100 trials. The smallest feedforward network used for the problem in the literature with acceptable results has 4 hidden units with shortcuts and 25 weights, which was trained by NOVEL [50].

In this experiment, the proposed method is first applied to the two-spiral problem with the two spirals coil 1.5 times around the origin and around one another. We use 97 sample points with noise on each spiral for training set. The noises are Gaussian white noises with zeros-mean and variance 0.01. Figure 3.10(a) shows the training samples for the problem. The architecture used for this experiment is a no-shortcut $2 - 7 - 1$ feedforward network with 29 weights including biases. The activation function is asymmetric sigmoid for both hidden and output neurons. The error function is the sum of the squared-errors. In applying the proposed method to the two-spiral problem, we start from a point generated randomly in the weight space ranging between $[-1, 1]$ in each dimension. For the proposed method, set $k' = 6$, $m_0 = 50$, $\Delta r = 0.05$ and $p_a = 1$. The allowed maximal number of training epochs is 10000. The target sum of squared-errors is set to be $1e - 10$. The proposed method found the global optimal solution with 881 epochs at function value of $4.8990e-11$. The decision region according to the conventional $40 - 20 - 40$ criterion of the trained network is shown in Figure 3.10(b). To eliminate the bias in choosing

good initial weights, we select 10 different initializations. The proposed method achieves 100% success at training, while the traditional Levenberg-Marquardt algorithm failed at each time. Figure 3.11 displays the decision regions for the same problem by $2 - 8 - 1$ and $2 - 9 - 1$ neural networks, respectively, trained by the proposed method. Similar experiments were presented in [72], where the convergence was achieved by a R-GEM global optimization algorithm using a $2 - 15 - 1$ neural network with 61 weights.

Figure 3.10: Two-Spiral problem: (a) Training samples ; (b) Decision region.

Figure 3.11: Decision regions of networks trained by the proposed method with different number of hidden units: (a) A $2 - 8 - 1$ neural network ; (b) A $2 - 9 - 1$ neural network.

The proposed method is also applied to the two-spiral problem with spirals coiling 3 times around the origin and one another. It is well known that the problem is extremely hard for feedforward neural networks, especially when the network structure is small. In this experiment, the architectures we used are $2 - 4 - 1$, $2 - 5 - 1$ and $2 - 6 - 1$ feedforward neural networks with shortcuts, the same as in [50]. The numbers of weights including biases are 25, 33 and 42 respectively. The initial weight is the origin of the weight space to eliminate any bias in choosing good starting points in the search [50]. For the proposed method, set $k' = 20$, $m_0 = 30$, $\Delta r = 0.05$ and $p_a = 3$ for the $2 - 4 - 1$ neural network, $k' = 20$, $m_0 = 30$, $\Delta r = 0.1$ and $p_a = 3$ for the $2 - 5 - 1$ network and $k' = 20$, $m_0 = 30$, $\Delta r = 0.25$ and $p_a = 1$ for the $2 - 6 - 1$ network. The activation function is asymmetric sigmoid for both hidden and output neurons. The error function is the total sum of the squared-errors.

The decision regions according to the conventional $40 - 20 - 40$ criterion of the trained network are shown in Figures 3.12 - 3.14. The test results in Table 3.10 by the proposed method are comparable to those by NOVEL [50] under the same test conditions.

Figure 3.12: Decision regions of $2 - 4 - 1$ feedforward network with shortcuts trained by the proposed method

Figure 3.13: Decision regions of 2 – 5 – 1 feedforward network with shortcuts trained by the proposed method

Figure 3.14: Decision regions of 2 – 6 – 1 feedforward network with shortcuts trained by the proposed method

Table 3.10: Comparison Results for Spiral Problem

2-Spiral Problem	No. of Hidden Units					
	4		5		6	
Results \ Methods	Proposed Method	NOVEL	Proposed Method	NOVEL	Proposed Method	NOVEL
Training TEES	19.8	14.0	4	6.0	2	0.0
Training correction (%)	75.3	92.8	97.9	96.9	98.97	100
Testing correction (%)	87.1	85.6	96.4	94.8	97.4	95.4
No. of restarts	314	1200	372	1200	305	1200
No. of epochs	1.5316e7	N/A	1.8434e7	N/A	1.524e7	N/A

3.6 Conclusions

In this chapter, the subenergy tunneling function of the TRUST is improved over the original one. The enhancement is obvious and the improved subenergy tunneling function can lead to satisfactory behavior of the transformation. A new algorithm for efficient global optimization has been presented based on the combination of the improved subenergy tunneling function and the ripple search to inherit the advantages of both methods. The proposed approach has been applied to several neural network applications. The performance on accuracy and efficiency of the new method has been demonstrated better than that of the original TRUST algorithm for several benchmark problems. It also compares favorably in general with other global optimization methods on the topic. As for the specific problems discussed in this chapter, we have been able to train the neural networks which are smaller than or equal to the smallest structures found in the literature, while many other global optimization algorithms still work on general-sized neural networks. The proposed method is conceptually simple to implement in neural network applications.

Chapter 4

Function Extensions for Efficient Global Optimization

In this chapter, two methods exploiting function extension are proposed to improve the efficiency of global optimization algorithms. After applying the proposed function extension techniques, the objective function is transformed while preserving the characteristics of the original objective function relevant for optimization, such as the locations of all maxima and minima and their function values. By embedding the techniques into global optimization algorithms, such as ripple search strategy, it is more reliable than using the improved subenergy tunneling function for global algorithms to judge which trial point is promising to develop, hence saving the computational cost of unpromising trial points.

4.1 Function Extension Techniques

In Chapter 3, improved subenergy tunneling function is proposed to identify promising trial points. The disadvantage is that as more local minima are found, it is more difficult to locate a trial point in the promising region, in which there is definitely a better solution than the current best one. This is because that the promising region shrinks exponentially with the dimensions of a problem along with the optimization process. An 1D example is given in Figure 4.1. Suppose x_3^* is the global minimum. When the first local minimum x_1^* is found, the promising region for x_3^* is the interval $[a, b]$. After the second local

minimum x_2^* is found, the promising region for x_3^* is reduced to be the interval $[c, d]$ and the possibility of missing the global solution increases. This is also unavoidable to other methods trying to identify promising points in a similar way.

Figure 4.1: Desired properties of the transformation.

To overcome the limitation mentioned above, new transformation of the original objective function should be designed to identify promising trial points. Its desired properties are summarized as follows:

- The portion of the objective function in each attraction region should be forced to extend, as shown in Figure 4.1, along the arrowheads. In this way, the size of the promising region does not change much with the updating of the current best solution;
- The locations of all maxima and minima and their function values should keep unchanged after the transformation to preserve the characteristics of the original objective function relevant for optimization;
- The portion of the extended function in each attraction region should be *nearly flat and smooth*. That is, any trial point located in an attraction region, except for those near the edge of the region, should have similar function value as the local minimum

of the original objective function. Because any dramatical change of the extended function value may cause incorrect identification of promising trial points.

The desired function after transformation is like the dashed curve with '*' maker in Figure 4.1. Its promising region for x_3^* is the interval $[e, f]$ after the second minimum x_2^* is found and does not change much during the optimization process.

In the rest of this chapter, two transformations called function extension techniques designed to possess these properties are proposed. One of them is non-iterative function extension that can get the desired function in a single transformation. The other uses iterative computations to transform the objective function and therefore is called iterative function extension.

4.1.1 Non-iterative Function Extension

Let $f(x)$ be a function of the scalar variable x , and $\mathbf{A} = [a_{ij}]$ be a matrix or a vector. For notational simplicity, we use $f(\mathbf{A})$ to denote the matrix $[f(a_{ij})]$.

For the original objective function $f(\mathbf{X})$, function $\mathbf{J}_1$ is:

$$\mathbf{J}_1 = \tanh(10\mathbf{Y}_\mathbf{J}) \cdot * \mathbf{Y}_\mathbf{J}$$

where

$$\mathbf{Y}_\mathbf{J} = \frac{\partial Y_Y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial Y_Y}{\partial x_1} \\ \frac{\partial Y_Y}{\partial x_2} \\ \vdots \\ \frac{\partial Y_Y}{\partial x_n} \end{bmatrix}$$

with

$$Y_Y = (T_Y^2)^{\frac{1}{c_1}}$$

and

$$T_Y = \begin{cases} f(\mathbf{X}) & (f_{\min}(\mathbf{X}) > 0) \\ f(\mathbf{X}) + \delta & (f_{\min}(\mathbf{X}) = 0) \\ \frac{1}{15} \ln\left(\frac{1}{1 + \exp(-15(f(\mathbf{X}) - (-0.01)))}\right) + (-0.01) & (f_{\min}(\mathbf{X}) < 0) \end{cases}$$

where c_1 is a predefined fixed scalar and δ is a small positive scalar, such as $\delta = 0.01$.

Function $\mathbf{J}_2$ is:

$$\begin{aligned}\mathbf{J}_2 &= \frac{\tanh(10f(\mathbf{X})) + 1}{2} \left(\left(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} \right)^{\cdot \wedge 2} \right)^{\cdot \wedge \left(\frac{1}{c_2}\right)} \\ &= \frac{\tanh(10f(\mathbf{X})) + 1}{2} \left(\begin{bmatrix} \frac{\partial f(\mathbf{X})}{\partial x_1} \\ \frac{\partial f(\mathbf{X})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{X})}{\partial x_n} \end{bmatrix}^{\cdot \wedge 2} \right)^{\cdot \wedge \left(\frac{1}{c_2}\right)}\end{aligned}$$

where c_2 is a predefined fixed scalar. We use $\cdot \wedge (2)$ and $\cdot \wedge \left(\frac{1}{c_2}\right)$ to denote to raise the power operation of power 2 and power $\frac{1}{c_2}$ respectively to a matrix or a vector element-wise.

Refer to Equation (3.6), the improved subenergy tunneling transformation of $(\mathbf{J}_1 + \mathbf{J}_2)$ at the predefined fixed scalar b is:

$$\mathbf{J} = \frac{1}{15} \ln\{[1 + \exp(-15(\mathbf{J}_1 + \mathbf{J}_2 - b)]^{\cdot \wedge (-1)}\} + b$$

where we use $\cdot \wedge (-1)$ to denote to raise the power operation of power -1 to a matrix or a vector element-wise.

Let vector $\mathbf{X}_A$ be:

$$\mathbf{X}_A = \tanh\left(10\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}\right) \cdot * \mathbf{J} = \tanh\left(10\begin{bmatrix} \frac{\partial f(\mathbf{X})}{\partial x_1} \\ \frac{\partial f(\mathbf{X})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{X})}{\partial x_n} \end{bmatrix}\right) \cdot * \mathbf{J}$$

with $*$ denoting to multiply element-by-element two matrixes or vectors. The $\tanh(ax)$ is hyperbolic tangent function with the following properties:

$$\tanh(ax) = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}} \implies \begin{cases} \tanh(ax) \approx 1 & (x > \frac{3}{a}) \\ \tanh(ax) = 0 & (x = 0) \\ \tanh(ax) \approx -1 & (x < -\frac{3}{a}) \end{cases}$$

The non-iterative function extension $f_{non_i_exe}(\mathbf{X})$ of original objective function $f(\mathbf{X})$ is defined by:

$$f_{non_i_exe}(\mathbf{X}) = f(\mathbf{X} - \mathbf{X}_A), \quad (4.1)$$

The derivative of $f_{non_i_exe}(\mathbf{X})$ with respect to $\mathbf{x}$ is:

$$\frac{\partial f_{non_i_exe}(\mathbf{X})}{\partial \mathbf{X}} = \frac{\partial f(\mathbf{X} - \mathbf{X}_A)}{\partial \mathbf{X}} \left(1 - \frac{\partial \mathbf{X}_A}{\partial \mathbf{X}}\right)$$

When $\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} = 0$, we have $\mathbf{X}_A = 0$ and $\frac{\partial \mathbf{X}_A}{\partial \mathbf{X}} = 0$, therefore,

$$\frac{\partial f_{non_i_exe}(\mathbf{X})}{\partial \mathbf{X}} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} = 0 \quad (4.2)$$

and the non-iterative extension function $f_{non_i_exe}(\mathbf{X})$ has the following properties:

$$\begin{cases} f_{non_i_exe}(X_{min}) = f(X_{min}) \\ f_{non_i_exe}(X_{max}) = f(X_{max}) \end{cases} \quad (4.3)$$

Equations (4.2) and (4.3) show that, after transformation, the locations of all maxima and minima and their function values of original objective function keep unchanged. That is, $f_{non_i_exe}(\mathbf{X})$ is a transformation of $f(\mathbf{X})$ preserving the characteristics of the original objective function relevant for optimization. The non-iterative function extension defined by Equation (4.1) can get desired properties in a single transformation.

Figures 4.2 - 4.4 show examples of an 1D function

$$f(x) = x \sin x$$

to which the transformation in Equation (4.1) has been applied with $b = 1.5$, $c_1 = 7$ and $c_2 = 8$. In Figures 4.2 and 4.3,

$$\mathbf{J}'_1 = \frac{1}{15} \ln\{[1 + \exp(-15(\mathbf{J}_1 - b))]^{\wedge(-1)}\} + b$$

$$\mathbf{J}'_2 = \frac{1}{15} \ln\{[1 + \exp(-15(\mathbf{J}_2 - b))]^{\wedge(-1)}\} + b$$

that is, $\mathbf{J}'_1$ and $\mathbf{J}'_2$ is the improved subenergy tunneling transformation of $\mathbf{J}_1$ and $\mathbf{J}_2$ at b respectively.

Figure 4.4 illustrates the advantages of function extension technique and explains how it is superior to the improved subenergy tunneling function. For example, after the first local minimum x_1^* is found, the function extension technique treats trial points located in the interval $[c, d]$ as promising points, while the improved subenergy tunneling function discards those in the intervals $[c, a]$ and $[b, d]$, which may include good trial points leading to the global minimum and the possibility of missing the global solution increases. Thus function extension is more reliable than the improved subenergy tunneling function on recognizing promising points.

Figure 4.2: Illustration of non-iterative function extension: $\mathbf{J}_1'$.

Figures 4.5 - 4.9 show other examples of the transformation applied to the 2D function:

$$f_1(\mathbf{X}) = f_1(x_1, x_2) = \sin x_1 \cos x_2$$

with $b = 1.5$, $c_1 = 3$ and $c_2 = 8$.

Figures 4.10 - 4.14 show the transformation applied to Rastrigin function in the search space $-5.12 \leq x_1, x_2 \leq 5.12$:

$$f_2(\mathbf{X}) = f_2(x_1, x_2) = x_1^2 + x_2^2 - 10 \cos(2\pi x_1) - 10 \cos(2\pi x_2)$$

Figure 4.3: Illustration of non-iterative function extension: J_2' and J . Figure 4.4: Comparison of $f_{non-iter}(\mathbf{X})$ and $f(\mathbf{X})$ as an example of 1D function.

Figure 4.5: Function and contour plot of $f_1(\mathbf{X})$ as a 2D example Figure 4.6: Function and contour plot of $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$

with $b = 0.15$, $c_1 = 5$ and $c_2 = 8$.

The desired properties of identifying promising points are realized for all cases by the non-iterative function extension, as demonstrated in the figures.

Figure 4.7: Function and contour plot of $f_{non.i.exe}(\mathbf{X})$ for $f_1(\mathbf{X})$ Figure 4.8: Function and contour plot of $f_{non.i.exe}(\mathbf{X})$ for $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$

4.1.2 Iterative Function Extension

The iterative function extension $f_{i.exe}(\mathbf{X}) = f_{i.exe,k}(\mathbf{X})$ is defined by:

Figure 4.9: Projection on x_1 - x_2 plane: comparison of $f_{non_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_1(\mathbf{X})$ after improved subenergy tunneling transformation at $f_1(\mathbf{X}) = 0$

Figure 4.10: Function and contour plot of $f_2(\mathbf{X})$ as 2D example

$$\begin{aligned}
 f_{i_exe_0}(\mathbf{X}) &= f(\mathbf{X}) \\
 f_{i_exe_1}(\mathbf{X}) &= f_{i_exe_0}(\mathbf{X} - b \tanh(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}})) \\
 f_{i_exe_2}(\mathbf{X}) &= f_{i_exe_1}(\mathbf{X} - b \tanh(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}})) \\
 &\vdots \\
 f_{i_exe_k}(\mathbf{X}) &= f_{i_exe_(k-1)}(\mathbf{X} - b \tanh(\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}))
 \end{aligned} \tag{4.4}$$

Figure 4.11: Function and contour plot of $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$

Figure 4.12: Function and contour plot of $f_{non_i_exe}(\mathbf{X})$ for $f_2(\mathbf{X})$

where k and b are predefined fixed scalars called iteration number and extention stepsize respectively.

Definition 4.1.1.

Let $f_{i_exe_k}(\mathbf{X})$ be the k -th iteratively extended function of $f(\mathbf{X})$, $\mathbf{X}_1^*$ be an arbitrary local or global minimum and φ be a small positive scalar. For any point $\mathbf{X}$ in the attraction region

Figure 4.13: Function and contour plot of $f_{non-iexe}(\mathbf{X})$ for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$

Figure 4.14: Projection on x_1 - x_2 plane: comparison of $f_{non-iexe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$

of $\mathbf{X}_1^*$, except those near the maxima, the portion of the extended function $f_{iexe,k}(\mathbf{X})$ in the attraction region of $\mathbf{X}_1^*$ is nearly flat and smooth if

$$f(\mathbf{X}_1^*) \leq f_{iexe,k}(\mathbf{X}) < f(\mathbf{X}_1^*) + \varphi.$$

Let $\mathbf{X}_1^*(x_1^*, x_2^*, \dots, x_n^*)$ be an arbitrary minimum, where n is the number of variables in

the objective function. Denote the set of points in the attraction region of $\mathbf{X}_1^*$, except those close to the maxima, as $\mathbf{X}(\{x_1\}, \{x_2\}, \dots, \{x_n\})$. Let $K = \text{Int}(\frac{\max(|x_1^* - \{x_1\}|, |x_2^* - \{x_2\}|, \dots, |x_n^* - \{x_n\}|)}{b})$, where $\text{Int}(\cdot)$ means taking the integer part of a number and $|\cdot|$ means taking the absolute value. We have the following theorem:

Theorem 4.1.1. *In the iterative function extension described by Equation (4.4), when $k = K$ and b is small, the portion of the extended function $f_{i_exe_k}(\mathbf{X})$ in the attraction region of $\mathbf{X}_1^*$ is nearly flat and smooth.*

Proof:

Without loss of generality, one dimensional case is considered in the following:

For function $f(x)$, let $g(x) = b \tanh(\frac{df(x)}{dx})$, therefore

$$f_{i_exe_1}(x) = f(x - g(x)) = f(x - c_1)$$

where, $c_1 = g(x)$. Similarly we have,

$$f_{i_exe_2}(x) = f(x - g(x) - g(x - g(x))) = f(x - c_1 - c_2)$$

where, $c_2 = g(x - g(x)) = g(x - c_1)$;

$$f_{i_exe_3}(x) = f(x - g(x) - g(x - g(x)) - g(x - g(x) - g(x - g(x)))) = f(x - c_1 - c_2 - c_3)$$

where, $c_3 = g(x - g(x) - g(x - g(x))) = g(x - c_1 - c_2)$;

...

$$f_{i_exe_k}(x) = f(x - c_1 - c_2 - \dots - c_k)$$

with

$$c_k = g(x - c_1 - c_2 - \dots - c_{k-1}).$$

Suppose one of the local minima is x_1^* with function value of $f(x_1^*)$ and one of the neighborhood local maxima is x_2^* with function value of $f(x_2^*)$. Without loss of generality, let $x_1^* < x < x_2^*$, then we have $0 < g(x) \leq b$.

Define $a = x - x_1^*$ and the necessary number of iterations $K_a = \text{Int}(\frac{a}{b})$.

Then,

$$c_1 + c_2 + \dots + c_{K_a} \leq a,$$

Therefore for any k and m with $m < k < K_a$, we have

$$x_1^* < x - c_1 - c_2 - \dots - c_k < x$$

and

$$f(x_1^*) < f_{i_exe_k}(x) < f_{i_exe_m}(x) < f(x).$$

Thus ,

$$\lim_{k \rightarrow K_a} f_{i_exe_k}(x) \rightarrow f(x_1^*) \quad (4.5)$$

For different x between $[x_1^*, x_2^*]$, the necessary number of iterations K_a is different, but the setting for the iteration number k is the same. We investigate the case that the number of iterations $L > K_a$.

$$\text{If } x - c_1 - c_2 - \dots - c_{L-1} = x_1^* + \delta > x_1^*$$

$$\text{and } x - c_1 - c_2 - \dots - c_L = x_1^* - \varepsilon < x_1^*,$$

where δ and ε are positive scalars.

$$\text{Then } \varepsilon < b, \ c_{L+1} = g(x - c_1 - c_2 - \dots - c_L) \text{ and } -b \leq c_{L+1} < 0.$$

Thus,

$$f_{i_exe_L+1}(x) = f(x - c_1 - c_2 - \dots - c_{L+1}) = f(x_1^* - \varepsilon - c_{L+1}).$$

$$\text{Because } -b \leq c_{L+1} < 0,$$

$$\text{when } \varepsilon + c_{L+1} \geq 0, \ f(x_1^*) \leq f_{i_exe_L+1}(x) < f(x_1^* - \varepsilon);$$

$$\text{when } \varepsilon + c_{L+1} \leq 0, \text{ obviously there is } x_1^* - \varepsilon - c_{L+1} \leq x_1^* - \varepsilon + b,$$

$$\text{therefore } f(x_1^*) \leq f_{i_exe_L+1}(x) < f(x_1^* - \varepsilon + b).$$

Finally we have

$$f(x_1^*) \leq f_{i_exe_L+1}(x) < \max[f(x_1^* - \varepsilon), f(x_1^* - \varepsilon + b)]. \quad (4.6)$$

Equation (4.5) and Equation (4.6) present that when k is sufficiently large and b is small, the portion of the extended function $f_{i_exe_k}(x)$ in each attraction region is nearly flat and smooth. The desired properties for identifying promising points are realized.

□

Figure 4.15 shows an example of an 1D function,

$$f(x) = x \sin x$$

to which the transformation in Equation (4.4) has been applied with $b = 0.2$. In Figure 4.15, $f_8(x)$ is the function extended after 8 iterations. Compared with the result in Figure

4.4 by non-iterative function extension, the portion of the extended function $f_8(x)$ in each attraction region is flatter and smoother.

Figures 4.16 - 4.19 show other examples of the transformation applied to the 2D function:

$$f_3(\mathbf{X}) = f_3(x_1, x_2) = \sin x_1 \sin x_2$$

with $b = 0.3$ and $k = 8$.

Figures 4.20 - 4.22 show the transformation applied to Rastrigin function f_2 in the search space $-6 \leq x_1, x_2 \leq 6$ with $b = 0.1$ and $k = 3$.

Figures 4.23 - 4.25 show the transformation applied to the following 2D function in the search space $-5.12 \leq x_i \leq 5.12$:

$$f_4(\mathbf{X}) = f_4 = \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i)$$

with $b = 0.3$ and $k = 8$.

The desired properties of identifying promising points are realized for all cases by the iterative function extension, as demonstrated in the figures.

Figure 4.15: Comparison of $f_{i_exe}(\mathbf{X})$ and $f(\mathbf{X})$ for an example of 1D function.

Figure 4.16: Function and contour plot of $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$

Figure 4.17: Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_3(\mathbf{X})$

4.2 Function Extensions for Efficient Global Optimization

Function extension techniques can be used to construct efficient global optimization process. As can be seen from Theorem 2.2.1, Equation (2.14) and the analysis in Re-

Figure 4.18: Function and contour plot of $f_{i_exe}(\mathbf{X})$ for $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$

Figure 4.19: Projection on x_1 - x_2 plane: comparison of $f_{i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_3(\mathbf{X})$ after improved subenergy tunneling transformation at $f_3(\mathbf{X}) = 0$

mark 2.2.1, techniques for identifying promising points worth developing and excluding unpromising ones from consideration can definitely help to speed up the convergence and increase the probability of convergence to the global minimum in finite time. Actually the two function extension techniques proposed in this chapter can be used to identify

Figure 4.20: Function and contour plot of $f_{i_{exe}}(\mathbf{X})$ for $f_2(\mathbf{X})$ Figure 4.21: Function and contour plot of $f_{i_{exe}}(\mathbf{X})$ for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$

promising regions, not just promising points, and only develop a single point in the most promising region. Besides, the promising regions do not suffer from the dramatic decrease in size along with the optimization process, which is unavoidable to other methods trying to identify promising points, and thus it is not easy to miss good solutions. Therefore the proposed techniques are faster and more reliable to speed up the convergence. The idea of

Figure 4.22: Projection on x_1 - x_2 plane: comparison of $f_{i_exe}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_2(\mathbf{X})$ after improved subenergy tunneling transformation at $f_2(\mathbf{X}) = 0$

Figure 4.23: Function and contour plot of $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$

function extension is innovative and there is no other techniques possessing similar properties as the proposed methods, to the best of our knowledge. The framework of function extensions for global optimization includes exploring the feasible region by global search such as the ripple type search method, identifying promising regions by function extension

Figure 4.24: Function and contour plot of $f_{i_{exe}}(\mathbf{X})$ for $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$

Figure 4.25: Projection on x_1 - x_2 plane: comparison of $f_{i_{exe}}(\mathbf{X})$ (right) and $f(\mathbf{X})$ (left) for $f_4(\mathbf{X})$ after improved subenergy tunneling transformation at $f_4(\mathbf{X}) = 0$

techniques and developing the most promising point by local optimization methods.

The characteristics of the two function extension techniques are different, hence there are different ways to apply them to the global optimization problems.

The non-iterative function extension cannot guarantee that the portion of the extended

function in each attraction region is nearly flat and smooth because this highly depends on the setting of control parameters b , c_1 and c_2 . Therefore in practice, for reliability, it is better to be used to identify all promising points in turn during global search. All these points need to be developed respectively by a local optimization algorithm. For iterative function extension, as the portion of the extended function in each attraction region is nearly flat and smooth under mild conditions stated in Theorem 4.1.1, it can be used to identify the most promising region. Therefore there is only one point inside this region that needed to be developed by a local optimization algorithm. Consequently the convergence speed can be dramatically increased.

For clarity, the pseudo-algorithms of the two proposed methods embedded into ripple search strategy are given in Section 4.2.1 and Section 4.2.2 respectively.

4.2.1 Implementation of Non-iterative Function Extension with Global Optimization

—Initialization phase

Step 1.

Generate a random starting point $\mathbf{X}_0$.

Step 2.

Set the control parameters m_0 , p_a and Δr for ripple search strategy; Set the termination criteria; Set the control parameters b , c_1 and c_2 for non-iterative function extension.

Step 3.

Start from the point $\mathbf{X}_0$, adopt a proper local search algorithm to minimize the objective function. Get the solution $\mathbf{X}_0^*$ and the objective function value $f(\mathbf{X}_0^*)$.

Step 4.

Set $\mathbf{X}_{min} = \mathbf{X}_0^*$, $f_{min} = f(\mathbf{X}_0^*)$.

Step 5.

Set ripple $r = 1$.

Step 6.

Set $m = 1$.

— **Ripple search phase**

Step 7.

Generate $n_d(n_d - 1)/2$ random values of θ_{pq} for different p and q defined in Equation (2.3).

Construct and calculate T_{rm} by Equations (2.3)-(2.7);

Step 8.

Generate a new trial point $\mathbf{S}_{rm}$ by Equation (2.2).

— **Non-iterative function extension**

Step 9.

Transform $f(\mathbf{X})$ into the $f_{non_i_exe}(\mathbf{X})$ by Equation (4.1) and calculate the function value of $f_{non_i_exe}(\mathbf{S}_{rm})$.

Step 10.

If $f_{non_i_exe}(\mathbf{S}_{rm}) < f_{min}$, go to step 11; otherwise go to step 13.

— **Local minimization phase**

Step 11.

Let $\mathbf{S}_{rm}$ be the starting point. Apply the same local search method as in step 3 to $f(\mathbf{X})$ and get the corresponding solution $\mathbf{X}_{rm}$ and the value of the objective function $f(\mathbf{X}_{rm})$.

Step 12.

Set $\mathbf{X}_{min} = \mathbf{X}_{rm}$ and $f_{min} = f(\mathbf{X}_{rm})$.

Step 13.

Increase m by 1 and repeat steps 7 to 13, till $m > m_0 + p_a(r - 1)n_d$ or one of the termination criteria is satisfied.

Step 14.

Increase r by 1 and repeat steps 6 to 14, till the termination criteria is satisfied.

Step 15.

Let $\mathbf{X}^* = \mathbf{X}_{min}$. Therefore, $\mathbf{X}^*$ is the final solution of the optimization problem and $f(\mathbf{X}^*)$ is the corresponding objective function value.

4.2.2 Implementation of Iterative Function Extension in Global Optimization

— Initialization phase

Step 1.

Generate a random starting point $\mathbf{X}_0$.

Step 2.

Set the control parameters m_0 , p_a and Δr for ripple search strategy. Set the termination criteria.

Step 3.

Set the control parameters b and k for iterative function extension.

Step 4.

Set $\mathbf{X}_{min} = \mathbf{X}_0^*$, $f_{min} = f(\mathbf{X}_0)$.

Step 5.

Set ripple $r = 1$.

Step 6.

Set $m = 1$.

— Ripple search phase

Step 7.

Generate $n_d(n_d - 1)/2$ random values of θ_{pq} for different p and q defined in Equation (2.3).

Construct and calculate T_{rm} by Equations (2.3)-(2.7);

Step 8.

Generate a new trial point $\mathbf{S}_{rm}$ by Equation (2.2).

— Iterative function extension

Step 9.

Transform $f(\mathbf{X})$ into the $f_{i_exe_k}(\mathbf{X})$ by Equation (4.4) and calculate the function value of $f_{i_exe_k}(\mathbf{S}_{rm})$.

Step 10.

If $f_{i_exe_k}(\mathbf{S}_{rm}) < f_{min}$, update $\mathbf{X}_{min} = \mathbf{S}_{rm}$ and $f_{min} = f(\mathbf{S}_{rm})$.

Step 11.

Increase m by 1 and repeat steps 7 to 11, till $m > m_0 + p_a(r - 1)n_d$ or one of the termination criteria is satisfied.

Step 12.

Increase r by 1 and repeat steps 6 to 12, till the termination criteria is satisfied.

— **Local minimization phase**

Step 13.

Let $\mathbf{X}_0^* = \mathbf{X}_{min}$. Start from the point $\mathbf{X}_0^*$, adopt a proper local search algorithm to minimize the objective function $f(\mathbf{X})$. Get the solution $\mathbf{X}^*$ and the objective function value $f(\mathbf{X}^*)$. Therefore, $\mathbf{X}^*$ is the final solution of the optimization problem and $f(\mathbf{X}^*)$ is the corresponding objective function value.

4.3 Test and Simulation Results

Numerical simulations on the cost function

$$f(\mathbf{X}) = \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i)$$

in the search space $-7 \leq x_i \leq 7$ are presented to demonstrate the performance of the proposed methods with $n = 2$ and $n = 3$ respectively. This cost function has at least 2^n minima, with the global minimum at $(-2.9035, -2.9035, \dots, -2.9035)$ [82].

For the test on the 2D cost function, the initial point is generated randomly on the interval $[2, 7]^2$ which is a region relatively far from the global minimum to make the optimization difficult. To compare the difference in efficiency of algorithms for the cost function with different number of dimensions, the values of the first two variables of the initial point for the 3D cost function are set to be the same as those of the initial point for the 2D cost function, while the value of the third variable is set to be the same as the value of the second variable.

As discussed previously, in practice the non-iterative function extension is used to identify the promising points among all trial points generated during the global search. Therefore the control parameters m_0 , p_a and Δr of the non-iterative function extension with ripple search strategy are set to be the same as those of the ripple search strategy, as shown in

Table 4.1. For the iterative function extension, as the portion of the extended function in each attraction region is nearly flat and smooth, this technique can be used to identify promising regions by evaluating a single point in each region. Therefore the number of trial points required for convergence to a good solution can be decreased dramatically. The control parameters m_0 , p_a and Δr of the iterative function extension with ripple search strategy are set to generate sparser trial points than those of the ripple search strategy. All the experimental results are carried out without fine-tuning the control parameters of the proposed methods.

Table 4.1 shows the results of the ripple search strategy, non-iterative function extension with ripple search strategy, iterative function extension with ripple search strategy, ASA and DEA on the cost function. By comparing the number of function evaluations, it is obvious that iterative function extension with ripple search strategy is the most efficient one among all global optimization methods under comparisons, followed by the non-iterative function extension with ripple search strategy. Techniques of function extension can significantly improve the efficiency of the global optimization algorithm without loss on accuracy. Ripple search strategy is more efficient than ASA and DEA. The results also demonstrate that the difference on efficiency of algorithms enlarges as the number of dimensions of the cost function increases.

4.4 Conclusions

In this chapter, two novel techniques on function extension are proposed to improve the efficiency of global optimization algorithms. First, non-iterative and iterative function extensions are introduced, analyzed and illustrated with samples in detail. They are then embedded into global optimization algorithms. Test results demonstrate the good performance of the proposed approaches and show that they are very promising in this area. Our work presented here is a successful attempt to speed up ripple search strategy by function extension techniques. They can be applied to other global optimization algorithms as well. More simulation tests are expected in the future.

Table 4.1: Comparison Results on Cost Function

2D Cost Function					
	Ripple Search Strategy	Non-iterative Function Extension with Ripple Search	Iterative Function Extension with Ripple Search	ASA	DEA
Control Parameters	$m_0=20, p_a=2, \Delta r=0.25$	$m_0=20, p_a=2, \Delta r=0.25, b=1, c_1=5, c_2=8;$	$m_0=5, p_a=1, \Delta r=0.5, b=0.4, k=6$	TRS= $1e-5$ TAS=50, IT=1	NP=6,F=0.8 CR=0.5
Starting Points	(4,6.4)	(4,6.4)	(4,6.4)	(4,4)	
Final Function Value	-156.6647	-156.6647	-156.6647	-156.6647	-156.6602
No. of Function Evaluations	271	115	64	1707	288
3D Cost Function					
Starting Points	(4,6.4,6.4)	(4,6.4,6.4)	(4,6.4,6.4)		
Final Function Value	-234.9970	-234.9970	-234.9970		
No. of Function Evaluations	1030	232	78		

Chapter 5

Parallel Coordinates for Moderate- and High-dimensional Global Optimization

Global optimization algorithms, such as simulated annealing, are generally computationally expensive, especially when the number of dimensions of a problem is high. Inspired by multidimensional visualization techniques, the notion of Parallel Coordinates System, or Parallel Coordinates in short, is developed and introduced into the optimization research area to speed up the convergence rate of global optimization algorithms by transforming the original search domain into another search domain with lower dimensions. The proposed method is meant to address optimization problems of moderate and high dimensions. The idea of combining a global optimization algorithm with the novel Parallel Coordinates is explained and discussed in detail in this chapter. The superior performance of the proposed method is demonstrated through extensive numerical studies.

Our discussion begins with the introduction on multidimensional visualization techniques and our developed Parallel Coordinates before the main idea of the proposed method is presented.

5.1 Multidimensional Visualization Techniques and Parallel Coordinates

Numerical data are a low-level representation of natural phenomena, and are often hard to interpret directly. Visual representations are much easier to be understood. Multidimensional visualization is the process that transforms numerical data into visual representations [74].

A general tool for visualizing multidimensional data is Parallel Coordinates [74][75]. Parallel Coordinates is a technique pioneered in the 1980's [76] which has been applied to a diverse set of multidimensional problems [77]-[79]. In this technique, each data dimension in an d -dimensional space is represented as a horizontal or vertical axis, and the d separate, parallel axes are normally organized as uniformly spaced lines. A data element in the d -dimensional space is mapped to a polyline that traverses across all of the horizontal or vertical axes. With Parallel Coordinates, multi-dimensional data can be visualized in two dimensions with low complexity. Thus relations within multi-dimensional data can be discovered.

5.2 Developed Parallel Coordinates

In traditional Cartesian Coordinates, all axes are mutually perpendicular. The classic method of Parallel Coordinates represents the d -dimensional data as values on d coordinates parallel to the y -axis equally spaced along the x -axis. A d -dimensional data element $\mathbf{X}(x_1, \dots, x_i, \dots, x_d)$ is drawn by locating the x_i along the i th coordinate with a marker '*'. A polygonal line then connects all the markers, as shown in Figure 5.1(a). Each data element in the d -dimensional space corresponds to a polygonal line in Parallel Coordinates. The original purpose of using Parallel Coordinates is to visualize multidimensional data in two dimensions to find certain features among them. With slight modification, Parallel Coordinates can be introduced into optimization research area to speed up the convergence rate of global optimization algorithms by transforming the original search domain into another search domain.

Figure 5.1: Parallel Coordinates display of a 9-dimensional data element by: (a) a polygonal line; (b) a smooth curve.

In the thesis, instead of being connected by a polygonal line, the corresponding values on d coordinates of each d -dimensional data element are linked by a smooth curve, as the solid one shown in Figure 5.1(b). Using Fourier series, a general function $Y(\eta)$ is constructed to express the curve as:

$$Y(\eta) = a_0 + \sum_{j=1}^m (a_j \sin(j\eta) + b_j \cos(j\eta)) \quad (5.1)$$

where, a_0 , a_j and b_j are variables describing different curves; $\eta \in [0, d-1]$; m is a predefined integer. Thus, each smooth curve in Parallel Coordinates corresponds to a point in d -dimensional space and can be represented by the function $Y(\eta)$. We define

$$x_i = Y(i-1) = a_0 + \sum_{j=1}^m \{a_j \sin[(i-1)j] + b_j \cos[(i-1)j]\} \quad (5.2)$$

where, $i = 1, \dots, d$.

For an arbitrary given set of values for the variables $x_1, \dots, x_i, \dots, x_d$, it is clear that the above system of linear equations (5.2) has at least one solution set $a_0, a_1, b_1, \dots, a_m, b_m$ when $(2m+1) = d$. If $(2m+1) > d$, the system generally has infinite solution sets for $a_0, a_1, b_1, \dots, a_m, b_m$. Therefore to let a curve pass exactly through arbitrary '*' markers in Figure 5.1(b), the condition can be set as $(2m+1) = d$. The number of Fourier coefficients $a_0, a_1, b_1, \dots, a_m, b_m$ is not necessary to exceed the number of the original variables

$x_1, \dots, x_i, \dots, x_d$, say $(2m + 1) > d$. When $(2m + 1) < d$, the system has no solution in general. However, we can always find a solution set $a_0, a_1, b_1, \dots, a_m, b_m$ so that $Y(i - 1)$ can approximate x_i , for $i = 1, \dots, d$. In such a case, no curve can pass exactly through all the '*' markers in Figure 5.1(b).

5.3 Optimization with Parallel Coordinates

Let $f(\mathbf{X}) = f(x_1, \dots, x_i, \dots, x_d)$ be an objective function with d variables in the domain of interest $D \in R^d$. The optimization problem is to find

$$f(\mathbf{X}^*) = \min_{\mathbf{X} \in D} f(\mathbf{X}). \quad (5.3)$$

This section shows how to speed up the convergence rate of global optimization algorithm by introducing the notion of Parallel Coordinates. The framework of optimization with Parallel Coordinates includes the developed Parallel Coordinates for transforming the whole search space to a smaller search domain with lower dimensions and a global search algorithm, such as the simulated annealing or ripple search strategy, to explore the transformed search domain. The detailed idea of the proposed algorithm is described below.

5.3.1 Optimization with General Parallel Coordinates

The algorithm:

As explained previously, the set of points $\{\mathbf{X}\}$ in the d -dimensional space becomes a group of smooth curves in Parallel Coordinates and can be described by a set of functions $\{Y(\eta)\}$. Since each function $Y(\eta)$ corresponds to a point $\mathbf{X}$ in d -dimensional space with the value of $f(\mathbf{X})$, $\{\mathbf{X}\}$ is a vector space of functions and $f(\mathbf{X})$ is a functional depending on function $Y(\eta)$, denoted by $V[Y(\eta)]$. Consequently, the optimization problem becomes functional optimization aimed at finding a function $Y(\eta)$ with the minimal value of objective functional $V[Y(\eta)]$.

There are many standard methods for the functional optimization problem. A common one is the Ritz method [80]. Implementing the Ritz method is made possible by using

appropriate Ritz functions. In the proposed method, Equation (5.1) is taken as the Ritz function with $(2m+1) < d$. The mathematical formulation for the functional optimization problem is

$$\begin{aligned} \min_{Y(\eta) \in \{\mathbf{X}\}} V[Y(\eta)] &\Rightarrow \min_{Y(i-1) \in \{x_i\}} f(Y(0), \dots, Y(i-1), \dots, Y(d-1)) \\ &= f_Y(\mathbf{A}^*) = \min_{\mathbf{A} \in D_A} f_Y(\mathbf{A}) \end{aligned} \quad (5.4)$$

That is, the problem reduces to finding the best solution $\mathbf{A}^*$ of the $(2m+1)$ -dimensional function $f_Y(\mathbf{A})$, where $f_Y(\mathbf{A}) = f_Y(a_0, a_1, b_1, \dots, a_j, b_j, \dots, a_m, b_m)$, $D_A \in R^{2m+1}$. The components of $\mathbf{A}$ are Ritz coefficients. Minimizing the objective function $f_Y(\mathbf{A})$ can be done by any global optimization algorithm, such as simulated annealing or ripple search strategy. After the global solution $\mathbf{A}^*$ in domain D_A is obtained, function $Y(\eta)$ at $\mathbf{A}^*$ with minimal value of $V[Y(\eta)]$ can be presented by Equation (5.1), as the dashed curve shown in Figure 5.1(b). As $(2m+1) < d$, generally the curve described by $Y(\eta)$ at $\mathbf{A}^*$ will not pass exactly through $\mathbf{X}^*$ in Parallel Coordinates, but give an approximate solution $\hat{\mathbf{X}}$, which can be calculated by substituting the value of $\mathbf{A}^*$ into the system of linear equations (5.2).

To improve the accuracy of the solution obtained by the above stage, the next stage of the global optimization with Parallel Coordinates is to set $\hat{\mathbf{X}}$ as the starting point and further minimize the objective function $f(\mathbf{X})$ by a local optimization algorithm. If $\hat{\mathbf{X}}$ is a good approximation to the global solution $\mathbf{X}^*$, that is, it is in the attraction region of $\mathbf{X}^*$, the local optimization algorithm will be able to converge at the global solution starting from it. The definition of the attraction region has been previously introduced in Chapter 2. As discussed previously, in the case of $(2m+1) < d$, generally the corresponding point $\hat{\mathbf{X}}$ in domain D of the point $\mathbf{A}^*$ in domain D_A is only an approximation to $\mathbf{X}^*$. To let $\hat{\mathbf{X}}$ be a good approximation to $\mathbf{X}^*$, the value of m should not be too small. If the effects of m are excluded from considerations, $\hat{\mathbf{X}}$ is more likely to be a good approximation of $\mathbf{X}^*$ when the global solution $\mathbf{X}^*$ corresponds to a flat curve in Parallel Coordinates. In practice, this condition can not always be guaranteed. However this limitation can be overcome by locating the value of x_i/R_0 along the i th coordinate in Parallel Coordinates instead of

the value of x_i , where R_0 is a predefined positive constant. The larger the value of R_0 , the flatter the curve appears. Based on this reason, the following sections in this chapter assume that the corresponding curve of the global solution $\mathbf{X}^*$ in Parallel Coordinates is flat and only discuss the effects of m on the performance of the proposed method.

Discussions:

In our proposed method, Parallel Coordinates is employed to transform the search domain D of original objective function $f(\mathbf{X})$ into another search domain D_A of objective function $f_Y(\mathbf{A})$. Consequently, in stage one, the search by global optimization is made in domain D_A rather than in domain D . Though the search domain and the objective function have been changed, the implementations of global optimization algorithms is the same. The successive points taken in domain D_A may appear to be completely random after they are mapped to domain D . Therefore the landscapes of functions $f_Y(\mathbf{A})$ and $f(\mathbf{X})$ are generally different. From previous descriptions, it is clear that when $(2m + 1) = d$, the curve described by $Y(\eta)$ at $\mathbf{A}^*$ can pass exactly through $\mathbf{X}^*$ in Parallel Coordinates. Therefore we have $f_Y(\mathbf{A}^*) = f(\mathbf{X}^*)$. In the proposed method, Equation (5.1) is taken as the Ritz function with $(2m + 1) < d$. In this case, generally the corresponding point $\hat{\mathbf{X}}$ in domain D of the point $\mathbf{A}^*$ in domain D_A is only an approximation to $\mathbf{X}^*$. If $\hat{\mathbf{X}}$ is a good approximation, we have $f_Y(\mathbf{A}^*) = f(\hat{\mathbf{X}}) \approx f(\mathbf{X}^*)$ in general. The accuracy of the solution can be further improved in stage two of the proposed method. Compared with the original objective function $f(\mathbf{X})$, function $f_Y(\mathbf{A})$ has fewer unknown variables and a smaller search region. As the computational cost by global optimization algorithms may grow exponentially with the size of the problem, reduction on the dimension of the search domain by Parallel Coordinates provides a means to speed up the convergence rate of global optimization algorithms.

Note that the integer m plays a key role in the trade-off between convergence rate and quality of solution. A smaller value of m leads to smaller search domain and more improvement on the convergence rate. However, if m is too small and the dimension of the given problem is high, $\hat{\mathbf{X}}$ may not be a good approximation to the global solution $\mathbf{X}^*$. That is, $\hat{\mathbf{X}}$ may be outside the attraction region of $\mathbf{X}^*$. Starting from it, the local

optimization algorithm will fail to converge at $\mathbf{X}^*$. To maintain the quality of $\hat{\mathbf{X}}$ while keeping m small, the method of Section Parallel Coordinates is proposed below.

5.3.2 Optimization with Section Parallel Coordinates

The algorithm:

Consider an objective function with d variables, where d is not a very small number. Divide the d variables into k sections, each of which has u variables, and leave the remaining variables in the $(k+1)$ th section, as shown in Figure 5.2. The number of remaining variables is $\text{rem}(d/u)$, where $\text{rem}(\cdot)$ means taking the remainder after division. For variables in the τ th section ($\tau = 1, \dots, k$), construct the Ritz function $Y_\tau(\eta)$ based on Equation (5.1) and form the Ritz coefficient set $\mathbf{A}_\tau(a_{\tau,0}, a_{\tau,1}, b_{\tau,1}, \dots, a_{\tau,v}, b_{\tau,v})$, where $2v+1 < u$. The variables in the $(k+1)$ th section remain unchanged. That is, for the Ritz function $Y_{k+1}(\eta)$ with Ritz coefficient set $\mathbf{A}_{k+1}(x_{ku+1}, \dots, x_d)$, let $Y_{k+1}(i-1) = x_{ku+i}$ for $i = 1, \dots, \text{rem}(d/u)$. Hence $f(\mathbf{X})$ becomes a functional relying on a group of the Ritz functions, noted as $V[Y_1(\eta), \dots, Y_\tau(\eta), \dots, Y_k(\eta), Y_{k+1}(\eta)]$. For each $Y_\tau(\eta)$, $\eta \in [(\tau-1)u, \tau u - 1]$. In succession, minimize the objective functional with unknown Ritz coefficient sets $\mathbf{A}(\mathbf{A}_1, \dots, \mathbf{A}_\tau, \dots, \mathbf{A}_k, \mathbf{A}_{k+1})$ by a global optimization algorithm. The approximate solution $\hat{\mathbf{X}}(\hat{\mathbf{X}}_1, \dots, \hat{\mathbf{X}}_\tau, \dots, \hat{\mathbf{X}}_k, \hat{\mathbf{X}}_{k+1})$ of $\mathbf{X}^*$ can be obtained through substituting the value of $\mathbf{A}_\tau^*$ into corresponding Ritz function $Y_\tau(\eta)$. It then acts as the starting point for stage two of the proposed method.

Figure 5.2: Illustration of Section Parallel Coordinates.

The ratio ξ of saving in dimensions by Section Parallel Coordinates is:

$$\xi = \frac{[u - (2v + 1)]k}{d} \times 100\%, \quad (5.5)$$

which is k times over the ratio of General Parallel Coordinates. Note that the General Parallel Coordinates is a special case of Section Parallel Coordinates with $u = d$, $v = m$ and $k = 1$. The saving in computational cost is proportional to the ratio ξ .

For example, if $d = 27$, $u = 9$, $v = 3$ and $k = 3$, from Equation (5.5), we have $\xi = 22.22\%$. The number of reduced dimensions is 6.

Pseudo-code of the algorithm:

For clarity of exposition, the pseudo-code of optimization with Section Parallel Coordinates is given as follows:

—Initialization

Step 1. Set the values of k , u and v .

Step 2. Generate a random starting point $\mathbf{A}_0$ in domain D_A .

Step 3. Construct the objective function $f_Y(\mathbf{A})$ by Section Parallel Coordinates. The original objective function $f(\mathbf{X})$ with d dimensions in domain D becomes the objective function $f_Y(\mathbf{A})$ with $[(2v + 1)k + \text{rem}(d/u)]$ dimensions in domain D_A .

— Stage one: global optimization on $f_Y(\mathbf{A})$

Step 4. Set control parameters and terminating criterion for the global optimization algorithm used.

Step 5. Start from the point $\mathbf{A}_0$, minimize the objective function $f_Y(\mathbf{A})$ by global optimization until the terminating criterion is met.

Step 6. Record the best solution $\mathbf{A}^*$ found in domain D_A .

Step 7. Calculate the corresponding solution $\hat{\mathbf{X}}$ in domain D of $\mathbf{A}^*$.

— Stage two: local minimization on $f(\mathbf{X})$

Step 8. Let $\hat{\mathbf{X}}$ be the starting point. Apply a local optimization method to minimize the objective function $f(\mathbf{X})$.

Step 9. Record the solution as the final solution $\mathbf{X}^*$ and its objective function value $f(\mathbf{X}^*)$.

Remark 5.3.1:

Optimization with Parallel Coordinates is proposed in Chapter 5 to transform the whole search space to a smaller search domain with lower dimensions. The technique of Parallel Coordinates was never applied before to global optimization problems. Note that the size of the search domain decreases exponentially with $[u - (2v + 1)]k$. Therefore the number of trial points required for convergence to a good solution can be decreased dramatically. Besides, the cost of function evaluations in each iteration is lower as it is proportional to the number of unknown variables in the objective function, which is now smaller. The convergence speed can thus be improved. To verify this analysis, the proposed algorithm will be tested in the following section by numerical studies.

5.4 Test Functions and Results

5.4.1 Test Results of Simulated Annealing with Parallel Coordinates

Simulated annealing is a well-known stochastic global optimization technique for obtaining high quality solutions to difficult optimization problems [26]. Its main drawback is the considerable computational effort required. Several variations of the classic simulated annealing have been proposed in the literature to improve the convergence rate, see e.g. [34][81]. However, high computational cost is still an obstacle to applications of simulated annealing based algorithms, especially for moderate and high dimensional problems. In this section, simulated annealing on behalf of global optimization algorithms is combined with Parallel Coordinates and performance of the hybrid method is tested.

Test functions:

- Quasi-Rosenbrock function.

The Rosenbrock test function [26] with fixed global solution at 1 on each dimension is notorious in optimization benchmarks, because a small change in variables can give unpredictably large changes in value of the objective function. To assess the performance of the proposed method, a quasi-Rosenbrock function with d dimensions is designed as

follows:

$$\begin{aligned}
 f(\mathbf{X}) &= f(x_1, \dots, x_i, \dots, x_d) \\
 &= \sum_{k=1}^{d-1} (100((x_{k+1} + 1 - p_{k+1}) - (x_k + 1 - p_k))^2 + (1 - (x_k + 1 - p_k))^2)
 \end{aligned} \tag{5.6}$$

where $\mathbf{P}(p_1, \dots, p_i, \dots, p_d)$ is the global solution which is known *a priori* and can be set arbitrarily. The unconstrained optimization problem is defined by minimizing the quasi-Rosenbrock function. The optimal function value is zero.

- Quasi-COST function.

The COST test function [82] with fixed global solution at -2.9035 on each dimension has at least 2^d local minima. To assess the performance of the proposed method, a quasi-COST function with d dimensions is defined by:

$$\begin{aligned}
 f(\mathbf{X}) &= f(x_1, \dots, x_i, \dots, x_d) \\
 &= \sum_{k=1}^d ((x_k - 2.9035 - p_k)^4 - 16(x_k - 2.9035 - p_k)^2 + 5(x_k - 2.9035 - p_k))
 \end{aligned} \tag{5.7}$$

The unconstrained optimization problem is to minimize the quasi-COST function with $\mathbf{P}(p_1, \dots, p_i, \dots, p_d)$ as the global solution.

Test conditions:

The version of simulated annealing used in this study comes from [26]. As suggested by Cornan et al. [26], the values of some control parameters for simulated annealing are set as:

- step variation $N_s = 20$, where N_s is the number of cycles of random moves along each coordinate direction, after which the step length is adjusted to maintain the average percentage of accepted moves at about one-half of the total number of moves;
- temperature reduction $N_T = \max(100, 5d)$, where N_T is the number of the step adjustments after which the temperature is reduced and d is the number of variables in the objective function;

- varying criterion $c_i = 2$ ($i = 1, \dots, d$), where c_i is the coefficient used to adjust the step length along each direction;
- number of successive temperature reductions to test for termination $N_\epsilon = 4$, where N_ϵ is the number of times the tolerance for termination on function value at each temperature reduction must be achieved before end.

The remaining control parameters to be adjusted and listed with test results for different cases are:

- starting temperature T_0 ;
- temperature reduction coefficient t_r , used to control the cooling rate of simulated annealing. The temperature reduction equation is $T_{n+1} = t_r \times T_n$, where n is the index denoting successive temperature reductions;
- terminating criterion ϵ . Simulated annealing is terminated when $|f_n - f_{n-t}| \leq \epsilon$ and $(f_n - f_{opt}) \leq \epsilon$, where $t = 1, \dots, N_\epsilon$, f_n is the function value at n th temperature reduction and f_{opt} is the current smallest function value.

The three control parameters are vital to the performance of simulated annealing. Among them, the starting temperature T_0 is more critical than the other two because if it is not high enough the algorithm would be easily blocked in local minima, regardless of the settings for t_r and ϵ . Experience has shown that the value for t_r should be between 0.8 and 0.99 [83]. As observed in our empirical studies, when the starting temperature T_0 is not high enough, the global minimum is hard to reach even when $t_r = 0.995$. Note that much greater computational effort is required for higher value of t_r . Terminating criterion ϵ is the tolerance on function value for termination of the algorithm when it ceases to make progress.

The tests for simulated annealing with Parallel Coordinates are made on the quasi-Rosenbrock functions with 9, 18, 27, 56 and 82 variables and the quasi-COST functions with 30 and 50 variables respectively. The results are compared with those of classic simulated annealing as well as the Quasi-Newton method [84], a local optimization algorithm.

All the test results are carried out under the MATLAB environment on IBM-PC/Pentium compatible machines with 2.4GHz clock speed.

For the comparison of the proposed method and classic simulated annealing, starting points are generated as follows:

Firstly, generate a uniformly distributed random point $\mathbf{A}_0$ in domain D_A . Secondly, map point $\mathbf{A}_0$ to point $\mathbf{X}_0$ in domain D based on Equation (5.2). $\mathbf{A}_0$ and $\mathbf{X}_0$ are the starting points for the proposed method to search in domain D_A and for the classic simulated annealing to search in domain D respectively.

Because the landscapes and the number of variables of functions $f_Y(\mathbf{A})$ and $f(\mathbf{X})$ are different, there is no absolute equivalence of the starting points in domains D_A and D . However, by Parallel Coordinates, the starting point $\mathbf{A}_0$ in domain D_A corresponds to a unique starting point $\mathbf{X}_0$ in domain D and hence $f_Y(\mathbf{A}_0) = f(\mathbf{X}_0)$. Let $\hat{\mathbf{X}}$ denote the point in domain D that corresponds to the global solution $\mathbf{A}^*$ obtained in domain D_A . It follows that $f_Y(\mathbf{A}^*) = f(\hat{\mathbf{X}}) \approx f(\mathbf{X}^*)$ where $\mathbf{X}^*$ is the global solution in domain D . Hence, it makes sense to use starting points in both domains D_A and D in the above manner.

The global minima $\mathbf{P}$ and the starting points of the quasi-Rosenbrock functions with 9, 18, 27, 56 and 82 variables respectively are generated randomly with uniform distribution on the intervals listed in Table 5.1. The global minima $\mathbf{P}$ of the quasi-COST functions, the optimal function values f^* and the starting points are listed in Table 5.2 for the 30 and 50 dimensions respectively.

Table 5.1: The Global Minima $\mathbf{P}$ and the Starting Points of the Quasi-Rosenbrock Functions

No. of Variables: 9	No. of Variables: 18	No. of Variables: 27
$P \in [-2, 2]^9$	$P \in [-2, 2]^{18}$	$P \in [-2, 2]^{27}$
$A_0 \in [0, 1000]^7$	$A_0 \in [0, 1000]^{14}$	$A_0 \in [0, 1000]^{21}$
No. of Variables: 56	No. of Variables: 82	
$P \in [-2, 2]^{56}$	$P \in [-2, 2]^{82}$	
$A_0 \in [0, 200]^{44}$	$A_0 \in [0, 100]^{64}$	
X_0 : generated from A_0		

Table 5.2: The Global Minima $\mathbf{P}$, the Optimal Function Values f^* and the Starting Points of the Quasi-COST Functions

No. of Variables: 30	No. of Variables: 50
$\mathbf{P} \in [-2, 2]^{30}$	$\mathbf{P} \in [-2, 2]^{50}$
$\mathbf{A}_0 \in [0, 5]^{24}$	$\mathbf{A}_0 \in [0, 5]^{40}$
$f^* = -2.35\text{E}3$	$f^* = -3.9166\text{E}3$
$\mathbf{X}_0$: generated from $\mathbf{A}_0$	

Test results:

Test results on quasi-Rosenbrock function

- Quasi-Rosenbrock function with 9 variables.

Apply the method of General Parallel Coordinates. Set $m = 3$ therefore $\mathbf{A}$ is a 7-dimensional variable vector. Detailed test results of a single run are reported in Table 5.3.

For both simulated annealing with and without Parallel Coordinates, set the starting temperature $T_0 = 1\text{E}6$ and reduction coefficient $t_r = 0.9$. Due to the approximation of solution caused by the condition $(2m + 1) < d$ to the Ritz function $Y(\eta)$, after stage one, the function value of 8.7223 is the best which the simulated annealing with Parallel Coordinates can achieve and is the global minimum in domain D_A . Thus the terminating criterion $\epsilon = 1\text{E}-1$ is sufficient. Because the corresponding point in domain D of the solution obtained during stage one is a near-global solution, local optimization by the Quasi-Newton method [84] in stage two can help to improve the result and find the global minimum at $8.0999\text{E}-8$. For classic simulated annealing, set the same terminating criterion $\epsilon = 1\text{E}-1$ followed by the Quasi-Newton method for local search, it fails to converge at the global minimum. This implies that the search is prematurely terminated and the solution at the terminating criterion is still too far from the global minimum. Therefore it is necessary to set $\epsilon = 1\text{E}-6$ to achieve a comparable accuracy. As shown in Table 5.3, the savings in the number of function evaluations and CPU time by the proposed method compared with the classic simulated annealing are 40.24% and 39.2% respectively. Note that the Quasi-Newton method by itself fails to reach the global minimum beginning

at the same starting point in domain D . Table 5.11 shows the test results averaged over 10 runs beginning with random starting points.

- Quasi-Rosenbrock function with 18 variables.

Apply the method of Section Parallel Coordinates. Set $k = 2$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 14-dimensional variable vector. The test results for one run are reported in Table 5.4.

For both simulated annealing with and without Parallel Coordinates, firstly set starting temperature $T_0 = 1E6$. The simulated annealing with Parallel Coordinates can converge to the optimal solution while the classic simulated annealing cannot converge, even with other settings for t_r and ϵ . After improving the starting temperature to $T_0 = 1E7$ for the latter, the classic simulated annealing converges to $1.8418E-7$. The savings in the number of function evaluations and CPU time are 42.12% and 11.8% respectively. Test results averaged over 10 runs beginning with random starting points are shown in Table 5.11.

- Quasi-Rosenbrock function with 27 variables.

Apply the method of Section Parallel Coordinates. Set $k = 3$, $u = 9$ and $v = 3$. Therefore $\mathbf{A}$ is a 21-dimensional variable vector. Detailed test results of a single run are reported in Table 5.5, while Table 5.11 shows the results averaged over 10 runs beginning with random starting points.

- Quasi-Rosenbrock function with 56 variables.

Apply the method of Section Parallel Coordinates. Set $k = 6$, $u = 9$ and $v = 3$. Therefore $\mathbf{A}$ is a 44-dimensional variable vector. Detailed test results for one run are reported in Table 5.6.

Due to the approximation of solution caused by the condition $(2v + 1) < u$ to the Ritz function, after stage one, the function value of $1.5563E+3$ is the best achieved by the proposed method. There is no more improvement on the result even with higher starting temperature T_0 . Thus the terminating criterion $\epsilon = 1E-1$ is sufficient. Because the distance to global solution is only 6.8433, it is a near-global solution. Therefore after applying the Quasi-Newton method in stage two, the final result converges at $4.8523E-$

8. For classic simulated annealing, it is necessary to set $\epsilon = 1E-6$ to achieve the global solution. As shown in the table, the classic simulated annealing fails to converge at $T_0 = 1E15$. Until the starting temperature is increased to $T_0 = 1E35$, it converges at the function value of $3.1118E-7$. Note that using the same terminating criterion $\epsilon = 1E-1$ as the proposed method followed by Quasi-Newton method for local search, classic simulated annealing can not converge to the global minimum. The savings in the number of function evaluations and CPU time are 67.59% and 30.2% respectively. Table 5.11 shows the test results averaged over 10 runs beginning with different starting points.

- Quasi-Rosenbrock function with 82 variables.

Apply the method of Section Parallel Coordinates. Set $k = 9$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 64-dimensional variable vector. Detailed test results for a single run are reported in Table 5.7. Table 5.11 shows the test results averaged over 10 runs beginning with different starting points.

Numerical studies on quasi-Rosenbrock function with different dimensions indicate that, with Parallel Coordinates, the requirement on the starting temperature of simulated annealing is lower than that without Parallel Coordinates and its influence on solution accuracy is less. Together with the saving in the size of search domain, they make noteworthy contribution to the improvement on the convergence rate of simulated annealing.

Test results on quasi-COST function

- Quasi-COST function with 30 variables.

Apply the method of Section Parallel Coordinates. Set $k = 3$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 24-dimensional variable vector. Detailed test results for one run are reported in Table 5.8, while Table 5.11 shows the test results averaged over 10 runs beginning with random starting points.

Table 5.9 shows the impact of choosing different sets of values for k , u and v . Among the three, u and v are more critical and k can be calculated by $k = \text{Int}(\frac{d}{u})$, where $\text{Int}(x)$ means taking the integer part of x . In the first case of $k = 3$, $u = 9$ and $v = 1$, the dimension of the optimization problem is only 12 and the computational costs of number

of function evaluations and CPU time are low. However the solution obtained during stage one is too far from the global solution as a starting point for a local optimization algorithm to globally converge in stage two. In the second case of $k = 3$, $u = 9$ and $v = 2$, the dimension of the optimization problem is 18. The solution obtained during stage one is close enough to the global solution for a local optimization algorithm to globally converge. However the computational costs increase. In the third case of $k = 3$, $u = 9$ and $v = 3$, the dimension of the optimization problem is 24 and consequently the computational costs are the highest among the three cases under comparisons. Note that the quality of the solution obtained during stage one is high therefore the computational costs in stage two are very low compared with the first two cases. These results demonstrate the trade-off between the convergence rate and the quality of solution. Reasonable values for u and v , as observed from our numerous empirical studies, are $u = 9$ and $v = 3$.

- Quasi-COST function with 50 variables.

Apply the method of Section Parallel Coordinates. Set $k = 5$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 40-dimensional variable vector. Detailed test results for a single run are reported in Table 5.10. Table 5.11 shows the test results averaged over 10 runs beginning with different starting points.

The difference on the requirement of the starting temperature for the proposed method and classic simulated annealing is not observed in the numerical studies on quasi-COST function. A possible reason is that the starting points are generated from much smaller intervals compared with those for quasi-Rosenbrock function.

5.4.2 Test Results of Ripple Search Strategy with Parallel Coordinates

Ripple search strategy on behalf of global optimization algorithms is combined with Parallel Coordinates and the performance of the hybrid method is tested.

Test conditions:

The tests are first made for a single run on the quasi-Rosenbrock functions in 9, 18 and 27 dimensions respectively. The results are compared with those of ripple search strategy without Parallel Coordinates and those of the Quasi-Newton method, a local optimization algorithm.

Because the landscapes and the number of variables of functions $f_Y(\mathbf{A})$ and $f(\mathbf{X})$ are different, as explained previously in Section 5.3.1, there is no absolute equivalence of the starting points in domains D_A and D . Starting points are generated by the same way as those for the tests of simulated annealing. More detailed information is described in Section 5.4.1.

The global value $\mathbf{P}$ of the quasi-Rosenbrock function is set as below for the 9, 18 and 27 dimensions respectively:

$\mathbf{P}=(0.5, 1, -0.2, 1, 0.4, 0.2, 0.5, 1.2, -0.2)$
$\mathbf{P}=(0.5, 1, -0.2, 1, 0.4, 0.2, 0.5, 1.2, -0.2, 0.8, 0.7, 0.9, 0.5, 0.6, 1, 0.5, 1, -0.2)$
$\mathbf{P}=(0.5, 1, -0.2, 1, 0.4, 0.2, 0.5, 1.2, -0.2, 0.8, 0.7, 0.9, 0.5, 0.6, 1, 0.5, 1, -0.2, 0.2, 0.5, 1.2, 1, 0.8, 0.7, 0.9, -0.2, 1.2)$

The tests are then made for 10 Monte-Carlo runs on each problem. The global minima $\mathbf{P}$ and the starting points of the quasi-Rosenbrock functions with 9, 18 and 27 variables respectively are generated randomly with uniform distribution on the intervals listed in Table 5.1.

Test results:

- Quasi-Rosenbrock Function with 9 Variables.

Select $m = 3$ and therefore $\mathbf{A}$ is a 7-dimensional variable vector. The test results for a single run are reported in Table 5.12.

Due to the approximation on Ritz function $Y_r(i-1)$ of Equation (5.13) caused by limited value of m , after the first stage, the function value of 8.7189 is the best to achieve. This solution is consistent with the value obtained after the first stage by simulated annealing

with Parallel Coordinates. After the second stage with a traditional local optimization, the final result is $1.6187E - 7$. As shown in Table 5.3, to the starting point in this case, the saving in number of function evaluations is 84.39% though the accuracy decreased slightly. The number of function evaluations is in inverse proportion to the efficiency. Note that the Quasi-Newton method fails to reach the global minimum.

- Quasi-Rosenbrock Function with 18 Variables.

Apply Section Parallel Coordinates to this case. Set $k = 2$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 14-dimensional variable vector. The test results for a single run are reported in Table 5.13. In this case, the saving in number of function evaluations is 51.94%.

- Quasi-Rosenbrock Function with 27 Variables.

Apply the method of Section Parallel Coordinates. Set $k = 3$, $u = 9$, $v = 3$ and therefore $\mathbf{A}$ is a 21-dimensional variable vector. The test results for a single run are reported in Table 5.14.

Table 5.15 shows the test results averaged over 10 runs beginning with random starting points for each problem discussed above. The good performance of the proposed method is verified.

5.5 Discussion: From Parallel Coordinates to Radiation Coordinates

Parallel Coordinates handles multiple dimensional data and visualize higher-order geometries in an easily recognizable 2D representation, which in turn provide a means to resolve higher dimensional optimization problems. However, it is limited to data sets that have only a few dimensions more than 3D, as the space needed for Parallel Coordinates increases linearly with the number of data dimensions and visualization becomes difficult. Radiation coordinates is proposed to eliminate the limitation. The basic idea of using radiation coordinates is to arrange the coordinate axes on a circle on a polar coordinate plane with equal angles between the axes with an origin at the center of the circle, as

shown in Figure 5.3. Any d -dimensional data element $\mathbf{X}(x_1, x_2, \dots, x_i, \dots, x_d)$ is drawn by locating the value of $R_0 + x_i$ along the i th coordinate, where R_0 is a positive constant and satisfy $R_0 > \max(|x_i|)$, for example, $R_0 = 3\max(|x_i|)$. Locating the value of $R_0 + x_i$ instead of x_i makes sure that the value is larger than zero to satisfy the requirement by polar coordinates. Moreover, it guarantees that the mapping between the value of any d -dimensional data element on each coordinate and the point in radiation coordinates for that coordinate is unique. All the points with '*' marker in the Figure 5.3 are then connected by a curve. The larger the value of R_0 , the flatter the curve appears.

Figure 5.3: Radiation coordinates display of a multi-dimensional data element by curve.

The radiation coordinates is similar to a technique called star coordinates [85]. In star coordinates, the mapping of a data element in a d -dimensional space to a point in a two-dimensional Cartesian Coordinates is determined by the sum of all unit vectors on each coordinate multiplied by the value of the data element for that coordinate. Therefore, a single data point in star coordinates may correspond to a number of data values [85]. In radiation coordinates, the mapping of any d -dimensional data element and the curve on polar coordinate plane is unique. Additionally, each point in d -dimensional space is represented by a curve in radiation coordinates, instead of a point in star coordinates. Similar to formula (5.1), the curve (r, θ) on polar coordinates plane can be described as:

$$r = R_0 + Y(\eta) \approx R_0 + a_0 + \sum_{j=1}^m (a_j \sin(j\eta) + b_j \cos(j\eta))$$

$$0 \leq \theta \leq 2\pi \quad (5.8)$$

By the concept of radiation coordinates, the X_i on the polar coordinate plane is presented by

$$\begin{aligned} r_{x_i} &= R_0 + x_i \\ \theta_i &= \frac{2\pi}{d}(i-1) = \frac{2\pi}{d}\eta_i \end{aligned} \quad (5.9)$$

Thus,

$$\eta_i = \frac{d\theta_i}{2\pi} \Rightarrow \eta = \frac{d}{2\pi}\theta \quad (5.10)$$

Substitute Equation (5.10) into Equation (5.8), the curve function on the polar coordinate plane is

$$r \approx R_0 + a_0 + \sum_{j=1}^m (a_j \sin(j \frac{d\theta}{2\pi}) + b_j \cos(j \frac{d\theta}{2\pi})) \quad (5.11)$$

Therefore,

$$\begin{aligned} r_{x_i} &\approx R_0 + a_0 + \sum_{j=1}^m (a_j \sin(j \frac{d\theta_i}{2\pi}) + b_j \cos(j \frac{d\theta_i}{2\pi})) \\ \theta_i &= \frac{2\pi}{d}(i-1) \end{aligned} \quad (5.12)$$

viz.

$$x_i = Y_r(i-1) \approx a_0 + \sum_{j=1}^m (a_j \sin(j(i-1)) + b_j \cos(j(i-1))) \quad (5.13)$$

where, $i = 1, 2, \dots, d$.

Optimization with radiation coordinates is the same as with Parallel Coordinates.

5.6 Conclusions

In this chapter, the notion of Parallel Coordinates originating from multidimensional visualization is introduced into the optimization research to transform the whole search domain into a smaller one with lower dimensions. It provides a means to speed up the convergence rate of global optimization algorithms for moderate and high dimensional optimization problems. Numerical studies have demonstrated the good performance of the proposed method. Our work presented here is a successful attempt to speed up simulated annealing and ripple search strategy by Parallel Coordinates. It can be applied to other global optimization algorithms as well.

Table 5.3: Tests on Quasi-Rosenbrock Function with 9 Variables

Method: Simulated Annealing (SA) No. of Variables: 9		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E6$, $\epsilon=1E-1$, $t_r=0.9$				
	Stage one: SA	2.8857	8.7223	334000	167
	Stage two: Local optimization	2.8782E-4	8.0999E-8	648	47
	Final result	2.8782E-4	8.0999E-8	334648	214
	CPU (in seconds)	197.443+0.339=197.782			
Classic SA	Control parameters: $T_0=1E6$, $\epsilon=1E-1$, $t_r=0.9$				
	SA	4.3978E3	4.3217E3	530000	265
	Local optimization	2.0572E3	2.0249E3	908	69
	Final result	2.0572E3	2.0249E3	530908	334
	CPU (in seconds)	306.3152+0.1812= 306.4964			
	Control parameters: $T_0=1E6$, $\epsilon=1E-6$, $t_r=0.9$				
	Final result	1.3012E-4	3.9211E-8	560000	280
CPU (in seconds)		325.767			
Ratio of Saving on Dimension: $\xi=22.22\%$					
Ratio of Saving on No. of Function Evaluations: 40.24 %					
Ratio of Saving on CPU time: 39.2 %					
Quasi-Newton Method		4.1545E4	6.1919E4	5323	409

Table 5.4: Tests on Quasi-Rosenbrock Function with 18 Variables

Method: Simulated Annealing (SA) No. of Variables: 18		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control Parameters: $T_0=1E6$, $\epsilon =1E-1$, $t_r=0.9$				
	Stage one: SA	4.1178	42.7385	358000	179
	Stage Two: Local optimization	4.9044E-4	7.5907E-8	1922	87
	Final Result	4.9044E-4	7.5907E-8	359922	266
	CPU (in seconds)	694.98+0.187=695.167			
Classic SA	Control parameter: $T_0=1E6$, $\epsilon =1E-6$, $t_r=0.9$				
	Result	1.9933	3.9866	580000	290
	CPU (in seconds)	668.064			
	Control parameters: $T_0=1E7$, $\epsilon =1E-6$, $t_r=0.9$				
	Final Result	2.7058E-4	1.8418E-7	608000	304
	CPU (in seconds)	788.853			
Ratio of Saving on Dimension: $\xi= 22.22 \%$					
Ratio of Saving on No. of Function Evaluations: 42.12 %					
Ratio of Saving on CPU time: 11.8 %					
Quasi-Newton Method		4.8201	85.5313	4697	247

Table 5.5: Tests on Quasi-Rosenbrock Function with 27 Variables

Method: Simulated Annealing (SA) No. of Variables: 27		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E6$, $\epsilon=1E-1$, $t_r=0.9$				
	Stage one: SA	5.0491	54.5202	371700	177
	Stage two: Local optimization	2.1841E-4	1.7295E-7	3804	122
	Final Result	2.1841E-4	1.7295E-7	375504	299
	CPU (in seconds)	1.1378E3+0.327=1.1382E3			
Classic SA	Control parameters: $T_0=1E6$, $\epsilon=1E-6$, $t_r=0.9$				
	Result	3.8312E3	3.7599E3	1104300	409
	CPU (in seconds)	1.7268E3			
	Control parameters: $T_0=1E9$, $\epsilon=1E-6$, $t_r=0.9$				
	Final Result	2.8318E-4	2.6429E-7	955800	354
	CPU (in seconds)	1.5364E3			
Ratio of Saving on Dimension: $\xi=22.22\%$					
Ratio of Saving on No. of Function Evaluations: 60.27 %					
Ratio of Saving on CPU time: 25.9 %					
Quasi-Newton Method		2.0924E4	2.1167E6	12535	401

Table 5.6: Tests on Quasi-Rosenbrock Function with 56 Variables

Method: Simulated Annealing (SA) No. of Variables: 56		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E15$, $\epsilon=1E-1$, $t_r=0.9$				
	Stage one: SA	6.8433	1.5563E3	1680800	382
	Stage two: Local optimization	2.0807E-4	4.8523E-8	9077	151
	Final Result	2.0807E-4	4.8523E-8	1689877	533
	CPU (in seconds)	1.6505E4+1.074=1.6507E4			
Classic SA	Control parameters: $T_0=1E15$, $\epsilon=1E-6$, $t_r=0.9$				
	Result		5.136E8		
	Control parameters: $T_0=1E35$, $\epsilon=1E-1$, $t_r=0.9$				
	SA	1.0181E6	1.0283E8	5084800	908
	Local optimization	1.9933	3.9866	54071	898
	Final Result	1.9933	3.9866	5138871	1806
	CPU (in seconds)	2.2889E4+4.862=2.2894E4			
	Control parameters: $T_0=1E35$, $\epsilon=1E-6$, $t_r=0.9$				
	Final Result	9.6226E-5	3.1118E-7	5213600	931
CPU (in seconds)	2.3655E4				
Ratio of Saving on Dimension: $\xi=21.43\%$					
Ratio of Saving on No. of Function Evaluations: 67.59 %					
Ratio of Saving on CPU time: 30.2 %					
Quasi-Newton Method		7.3419	53.4892	16937	281

Table 5.7: Tests on Quasi-Rosenbrock Function with 82 Variables

Method: Simulated Annealing (SA) No. of Variables: 82		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E10$, $\epsilon=1E-1$, $t_r=0.9$				
	Stage one: SA	8.9122	4.3251E3	1747200	273
	Stage two: Local optimization	3.0077E-4	1.1847E-7	22744	264
	Final Result	3.0077E-4	1.1847E-7	1769944	537
	CPU (in seconds)	3.9792E4+2.3769=3.9794E4			
Classic SA	Control parameters: $T_0=1E10$, $\epsilon=1E-6$, $t_r=0.9$				
	Result		4.1903E4		
	Control parameters: $T_0=1E80$, $\epsilon=1E-6$, $t_r=0.9$				
	Final Result	1.5418E-4	4.0158E-7	1.5719E7	1917
	CPU (in seconds)	9.4134E4			
Ratio of Saving on Dimension: $\xi=21.9\%$					
Ratio of Saving on No. of Function Evaluations: 88.74 %					
Ratio of Saving on CPU time: 57.73 %					
Quasi-Newton Method		8.9823	80.5882	30455	353

Table 5.8: Tests on Quasi-COST Function with 30 Variables

Method: Simulated Annealing (SA) No. of Variables: 30		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E3$, $\epsilon=1E-1$, $t_r=0.95$				
	Stage one: SA	2.1119	-2.2325E3	561600	234
	Stage two: Local optimization	1.8174E-4	-2.35E3	271	8
	Final result	1.8174E-4	-2.35E3	561871	242
	CPU (in seconds)	2.6771E3+0.031=2.6772E3			
Classic SA	Control parameters: $T_0=1E3$, $\epsilon=1E-6$, $t_r=0.95$				
	Final result	2.0007E-4	-2.35E3	1380000	460
	CPU (in seconds)	3.5588E3			
Ratio of Saving on Dimension: $\xi=20\%$					
Ratio of Saving on No. of Function Evaluations: 59.2 %					
Ratio of Saving on CPU time: 24.7 %					
Quasi-Newton Method		25.6284	-1.841E3	2134	61

Table 5.9: Comparisons of Different Sets of Values for k , u and v on Quasi-COST Function with 30 Variables

Method: Simulated Annealing (SA) No. of Variables: 30		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
Control parameters of SA: $T_0=1E3$, $\epsilon=1E-1$, $t_r=0.95$					
SA with Parallel Coordinates	No. of Variables: 30D $\rightarrow$ 12D ($k=3$, $u=9$, $v=1$)				
	Stage one: SA	10.4595	-1.7536E3	446000	223
	Stage two: Local optimization	9.7866	-2.2651E3	1516	45
	Final result	9.7866	-2.2651E3	447516	268
	CPU (in seconds)	845.6971+ 0.1812 = 845.8783			
	No. of Variables: 30D $\rightarrow$ 18D ($k=3$, $u=9$, $v=2$)				
	Stage one: SA	2.4484	-2.1707E3	448000	224
	Stage two: Local optimization	1.8589E-4	-2.35E3	150004	4837
	Final result	1.8589E-4	-2.35E3	598004	5061
	CPU (in seconds)	1.3115E3+9.5453=1.3210E3			
	No. of Variables: 30 D $\rightarrow$ 24D ($k=3$, $u=9$, $v=3$)				
	Stage one: SA	2.1119	-2.2325E3	561600	234
	Stage two: Local optimization	1.8174E-4	-2.35E3	271	8
	Final result	1.8174E-4	-2.35E3	561871	242
	CPU (in seconds)	2.6771E3+0.031=2.6772E3			

Table 5.10: Tests on Quasi-COST Function with 50 Variables

Method: Simulated Annealing (SA) No. of Variables: 50		Distance to Global Solution	Final Function Value	No. of Function Evaluations	No. of Iterations
SA with Parallel Coordinates	Control parameters: $T_0=1E5$, $\epsilon=1E-1$, $t_r=0.95$				
	Stage one: SA	2.1147	-3.7913E3	1312000	328
	Stage two: Local optimization	2.3524E-4	-3.9166E3	1176	22
	Final Result	2.3524E-4	-3.9166E3	1313176	350
	CPU (in seconds)	9.5220E3+0.4641=9.5225E3			
Classic SA	Control parameters: $T_0=1E5$, $\epsilon=1E-6$, $t_r=0.95$				
	Final Result	2.3256E-4	-3.9166E3	2775000	555
	CPU (in seconds)	1.0893E4			
Ratio of Saving on Dimension: $\xi=20\%$					
Ratio of Saving on No. of Function Evaluations: 52.68 %					
Ratio of Saving on CPU time: 12.58 %					
Quasi-Newton Method		37.4798	-3.8476E3	3314	48

Table 5.11: Test Results Averaged over 10 Runs for Each Problem

Methods	No. of Variables	Mean of Distance to Global Solution	Mean of Final Function Value	Mean of No. of Function Evaluations	Mean of No. of Iterations	Mean of CPU (in seconds)
Qusi-Rosenbrock Function						
SA with Parallel Coordinates	9	7.1952E-5	1.5413E-8	337795	213.8	285.9208
	18	3.2034E-4	1.2127E-7	352110	255.3333	618.2301
	27	2.3940E-4	6.8241E-8	377860	275.25	1.1051E3
	56	3.4296E-4	3.0015E-7	1.8568E6	638	1.8217E4
	82	6.4279E-4	4.3364E-7	1.7724E6	455.25	3.7282E4
Classic SA	9	1.0159E-4	3.7331E-8	563600	281.8	501.7438
	18	1.4846E-4	1.5139E-7	616500	308.25	992.2029
	27	1.1736E-4	8.8214E-8	1.2177E6	451.26	2.2197E3
	56	5.2331E-5	4.9234E-7	5.1912E6	927.13	2.1718E4
	82	4.3014E-4	1.7446E-7	1.5801E7	1926.15	9.5653E4
Qusi-COST Function						
SA with Parallel Coordinates	30	2.3160E-4	-2.35E3	549719	255.6667	2.3540E3
	50	2.8212E-4	-3.9166E3	1304000	346.21	9.3852E3
Classic SA	30	2.1484E-4	-2.35E3	1379625	459.875	3.3921E3
	50	2.7293E-4	-3.9166E3	2773125	554.625	1.1298E4

Table 5.12: Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 9 Variables

Method: Ripple Search Strategy (RSS) No. of Variables: 9		Distance to Global Solution	Final Function Value	No. of Function Evaluations
RSS with Parallel Coordinates	Starting point A_0 : (727.6849, 560.6643, 479.6526, 431.6208, 439.6605, 409.4268, 374.8936)			
	Stage one: RSS	2.8820	8.7189	2927
	Stage two: Local optimization	4.2215E-4	1.6187E-7	630
	Final result	4.2215E-4	1.6187E-7	3557
Basic RSS	Starting point X_0 : 1000 $\times$ (2.0219, 1.3548, 0.6694, 0.4607, 0.4496, -0.2962, 1.1111, 2.0825, 0.4555)			
		1.3697e-4	1.1924e-8	22780
Ratio of Saving on Dimension: $\xi = 22.22\%$				
Ratio of Saving on No. of Function Evaluations: 84.39%				
Quasi-Newton Method		4.1545E+4	6.1919E+4	5323

Table 5.13: Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 18 Variables

Method: Ripple Search Strategy (RSS) No. of Variables: 18		Distance to Global Solution	Final Function Value	No. of Function Evaluations
RSS with Parallel Coordinates	Starting point A_0 : (727.6849, 560.6643, 479.6526, 431.6208, 439.6605, 409.4268, 374.8936, 579.9253, 735.2417, 567.2480, 487.2399, 595.1934, 421.8317, 451.5192)			
	Stage one: RSS	4.1161	42.7333	10296
	Stage two: Local optimization	3.9189E-4	1.3591E-7	1906
	Final result	3.9189E-4	1.3591E-7	12202
Basic RSS	Starting point X_0 : 1000 $\times$ (2.0219, 1.3548, 0.6694, 0.4607, 0.4496, -0.2962, 1.1111, 2.0825, 0.4555, 0.6647, 0.0446, -0.3365, -0.0008, 2.6871, 0.3777, 0.8336, -0.0229, 0.0875)			
		1.2704E-5	3.2896E-9	25390
Ratio of Saving on Dimension: $\xi = 22.22\%$				
Ratio of Saving on No. of Function Evaluations: 51.94%				
Quasi-Newton Method		4.8201	85.5313	4697

Table 5.14: Ripple Search Strategy with Parallel Coordinates on Quasi-Rosenbrock Function with 27 Variables

Method: Ripple Search Strategy (RSS) No. of Variables: 27		Distance to Global Solution	Final Function Value	No. of Function Evaluations
RSS with Parallel Coordinates	Starting point A_0 : (727.6849, 560.6643, 479.6526, 431.6208, 439.6605, 409.4268, 374.8936, 579.9253, 735.2417, 567.2480, 487.2399, 595.1934, 421.8317, 451.5192, 432.5398, 701.6719, 389.8711, 513.2670, 389.6074, 505.6570, 456.5348)			
	Stage One: RSS	5.0518	54.5046	30372
	Stage two: Local optimization	2.7807E-4	3.7842E-8	3832
	Final result	2.7807E-4	3.7842E-8	34204
Classic RSS	Starting point X_0 : 1000 $\times$ (2.0219, 1.3548, 0.6694, 0.4607, 0.4496, -0.2962, 1.1111, 2.0825, 0.4555, 0.6647, 0.0446, -0.3365, -0.0008, 2.6871, 0.3777, 0.8336, -0.0229, 0.0875, -1.0570, 2.0788, 0.7659, 0.7145, -0.0347, 0.3210, -0.9945, 1.1921, 1.5310)			
		3.0398E-5	5.3776E-9	56942
Ratio of Saving on Dimension: $\xi = 22.22\%$				
Ratio of Saving on No. of Function Evaluations: 39.93%				
Quasi-Newton Method		2.0924E+4	2.1167E+6	12535

Table 5.15: Test Results Averaged over 10 Runs for Each Problem

Methods	No. of Variables	Mean of Distance to Global Solution	Mean of Final Function Value	Mean of No. of Function Evaluations	Mean of CPU (in seconds)
Quasi-Rosenbrock Function					
RSS with Parallel Coordinates	9	2.7654E-4	7.8535E-8	3819	6.8742
	18	3.9771E-4	1.2639E-7	12834	25.3675
	27	1.8607E-4	4.5765E-7	21592	45.3099
Classic RSS	9	8.0395E-5	7.0648E-9	17124	18.6992
	18	1.6010E-4	1.2032E-8	36305	44.7681
	27	1.5444E-4	4.7173E-8	54671	83.2580

Chapter 6

Achievable Accuracy of Parameter Estimation for Two-Dimensional NMR Experiments

6.1 Fundamentals

Many, if not most, NMR spectroscopy experiments involve parameter estimation problems, for example, for the determination of Larmor frequencies, J coupling constants, relaxation parameters, etc. The accuracy of the estimates of the spectral parameters which are obtained by NMR spectroscopy is becoming an increasingly important issue, in particular in the determination of high resolution protein structures. In many recently developed techniques the success of the experiment depends on being able to obtain highly accurate parameter estimates (see e.g. [86]-[88]).

A fundamental result in statistics and signal processing, typically known as the *Cramer Rao Lower Bound (CRLB)*, gives an explicit bound on the achievable accuracy of parameter estimation. The important aspect is that this result is independent of the particular estimation method that is being used, provided that this method is *unbiased*, i.e. that on average it will produce the correct value. This means for example that the bound is independent of whether time domain or frequency domain methods are applied. What does influence this bound is the underlying quality of the data, such as the type of noise,

the noise level, the number of data points, the sampling method etc.

The availability of such a bound has a number of important applications. For example, parameter estimation algorithms can be evaluated and compared based on how closely they approach this bound. Moreover, experimental designs can be compared to determine which design produces a data set with lower Cramer Rao Lower Bound and will therefore allow the parameters to be estimated more accurately. These considerations have of course been central to endeavors of all experimentalists. The Cramer Rao Lower Bound adds an analytical tool that can be used to evaluate the various approaches.

The work presented in this chapter is not the first to discuss the use of the Cramer Rao Lower Bound in the context of NMR spectroscopy. In [89] and [90] it has been used for the investigation of one-dimensional experiments. However, to the best of our knowledge, this is the first work that derives the analytical expression of the Fisher information matrix and calculates Cramer Rao Lower Bound for two-dimensional (2D) experiments, which can be generalized straightforwardly to multidimensional NMR experiments. Since multi-dimensional NMR experiments are notoriously time consuming it is especially important that those experiments are planned in such a way as to produce data that allows the parameters to be estimated most accurately.

In the main result of the chapter (Section 6.2 and Section 6.7, Appendix A) the Fisher information matrix, i.e. the inverse of the Cramer Rao Lower Bound, is calculated for a general data model of a 2D NMR data set that allows for an arbitrary number of resonances. The data is assumed to be corrupted by additive Gaussian noise. Aside from the presentation of this main result our objective is to illustrate how this result can be used in the context of 2D NMR spectroscopy. To this end we carefully investigate the Cramer Rao Lower Bound for simple example systems. Considering the significant role that the Fisher information matrix and Cramer Rao Lower Bound has played in other areas we are confident that the potential uses of the Cramer Rao Lower Bound go well beyond the few applications discussed here.

In Section 6.3 we discuss the relation between a nonlinear least-squares estimation problem and the Cramer Rao Lower Bound. In particular we show that in the case of an example data set a nonlinear least-squares estimation procedure can produce or come very close

to producing parameter estimates whose covariance matrix equals the Cramer Rao Lower Bound. This implies that the performance of a parameter estimation algorithm can be evaluated using the Cramer Rao Lower Bound. First, the quality of an alternative algorithm can be compared to the existing one by investigating how close the alternative algorithm comes to achieving the same (low) covariance matrix of the parameter estimates, knowing for theoretical reasons that none can be found that improves on the Cramer Rao Lower Bound. Second, it can serve to detect convergence problems of an algorithm. A notorious problem in the use of nonlinear estimation routines for parameter estimation in NMR applications is that the estimates may not converge fully to the ‘correct’ estimates or that the estimates may converge to an ‘incorrect’ value due to local minima of the criterion function. If the estimation procedure is initialized with different initial conditions the estimate of the covariance matrix of the parameter estimates can be determined. If this covariance estimate is significantly different from the Cramer Rao Lower Bound, it is an indication that either the algorithm is far from optimal or that serious convergence problems exist, e.g. due to local minima.

A second and possibly even more important use of the Cramer Rao Lower Bound in NMR spectroscopy lies in its use as an analytical tool for experiment design (Section 6.4). The Cramer Rao Lower Bound promises to be a particularly powerful tool for the analysis of experimental designs for multidimensional NMR experiments. The extensive experimental time that is required makes careful design of the experiment an important consideration. Since in many cases the purpose of an NMR experiment is to obtain estimates of certain parameters, it is the accuracy of these parameters that is a prime concern in the design of an experiment. The Cramer Rao Lower Bound and therefore the best achievable covariance properties of the estimates depend crucially on the experimental situation. The parameters that are given by the sample and the available equipment, e.g. frequencies of the resonances, interconnectivity patterns, typically have to be assumed to be fixed. On the other hand, many other experimental parameters such as the number of data points that are acquired, the sampling scheme, can be adjusted. Even the signal to noise ratio is adjustable by either changing the concentration of the sample or by averaging a number of identical scans. Limitations on material and experimental time put serious constraints

on the experimental design. The problem that arises, therefore, is to find and evaluate various experimental designs. The Cramer Rao Lower Bound provides a powerful tool for this purpose.

We should point out that it is not our intention here to advocate one experimental design over another. The Cramer Rao Lower Bound allows the spectroscopist to quantitatively evaluate the tradeoffs of the various designs from the point of view of the expected variance of the parameter estimates. In a concrete situation it is easily conceivable that the spectroscopist will take other criteria into consideration that are not reflected in the Cramer Rao Lower Bound. An experienced spectroscopist may have a good understanding of the qualitative benefits of one experimental design over another. We believe, however, that the availability of a powerful quantitative tool will be of value in many practical circumstances. As part of the discussion on experimental design we also investigate what we believe to be the novel notion of non-uniform averaging.

When the initial parameter values are unknown, then local search methods, such as the nonlinear least-square estimation algorithm, would not produce the desirable results on parameter estimation. In this case, global optimization is required. We process the simulated 2D NMR data using ripple search strategy. The expressions for CRLB are also computed numerically and shown by computer simulations to match with the variance of NMR spectroscopy parameters estimated by our proposed global optimization algorithm.

6.2 Analytical Expression for the Fisher Information Matrix

In this section the Cramer Rao Lower Bound is derived for 2D NMR data sets. *First, a very general data model is introduced for 2D NMR experiments. For this data model the Fisher information matrix, i.e. the inverse of the Cramer Rao Lower Bound, is derived (see Appendix A.1 for the derivation and Appendix A.2 for a complete listing of the result). As an illustration, in a subsequent example these results are specialized for a small example data set.* We now introduce the data model for 2D NMR experiments.

In an NMR experiment, the nuclear spin has an interaction with an applied magnetic

field, behaving like a magnetic moment. At equilibrium the magnetic moments are not aligned randomly but in such a way that when their contributions are all added up there is a net magnetic field along the direction of the applied field B_0 . This is called the bulk magnetization of the sample. The magnetization can be represented by a vector, called the magnetization vector, pointing along the direction of the applied field, z . Suppose that the equilibrium magnetization vector is of size M_0 . Once perturbed from the equilibrium state, the magnetization vector rotates about the direction of the magnetic field sweeping out a cone with a constant angle β . The x - and y -components of the magnetization, M_x and M_y respectively, vary with time t and can be worked out using simple geometry:

$$M_x = M_0 \sin \beta \cos(\omega_1 t)$$

$$M_y = M_0 \sin \beta \sin(\omega_1 t),$$

where ω_1 is the angular Larmor frequency. The signals detected, $S_x(t)$ and $S_y(t)$, are proportional to these magnetizations:

$$S_x(t) = c_1 \cos(\omega_1 t)$$

$$S_y(t) = c_1 \sin(\omega_1 t),$$

where c_1 is a constant.

It is convenient to regard $S_x(t)$ and $S_y(t)$ as the real and imaginary parts of a complex signal $S(t)$ [17]:

$$S(t) = S_x(t) + iS_y(t) = c_1 \cos(\omega_1 t) + ic_1 \sin(\omega_1 t) = c_1 e^{(i\omega_1 t)}.$$

Since the magnetization vector tends to return to its equilibrium state, the signal will decay over time and this is presented by an exponential decay with a constant r_1 . The signal then becomes:

$$S(t) = c_1 e^{(r_1 t)} e^{i\omega_1 t}. \quad (6.1)$$

Due to instrumental factors, generally the real and imaginary parts of $S(t)$ cannot correspond exactly to the x - and y -components of the magnetization. Mathematically, this is expressed by multiplying the ideal function by a phase ϕ_1 :

$$S(t) = c_1 e^{(r_1 t)} e^{i\omega_1 t} e^{i\phi_1} = c_1 e^{(r_1 t)} e^{i(\omega_1 t + \phi_1)}.$$

The electrons in a molecule surround the nuclei and create a small magnetic field which shields the nuclei slightly from the external field. Therefore, the angular Larmor frequencies of the same kind of nuclei at different locations vary due to their different chemical environment. An FID can then be generally described as a sum of exponentially decaying sinusoids plus noise $\epsilon(t)$ [91]:

$$S(t) = \sum_{k=1}^K c_{1k} e^{(r_{1k}t)} e^{i(\omega_{1k}t + \phi_{1k})} + \epsilon(t),$$

where K is called the number of resonances. The noise is caused by various factors, such as the imperfections in spectrometer and the condition of samples during the NMR data collection.

Consider now the 2D NMR Signal. Each 2D NMR experiment consists of four sections: preparation section, evolution section with time variable s_t , mixing section and detection section with time variable t . In the first section, the nuclei are excited by one or more pulses. The resulting magnetization is allowed to evolve to its equilibrium state for a time period s_t in the evolution section. It is the evolution period that is responsible for providing the information encoded into the second frequency domain. Then the mixing section follows, which consists of a further pulse or pulses. After the mixing section the signal is recorded as a function of the second time variable t . This sequence of events is called a pulse sequence. The value of s_t , starting with $s_t = 0$, increases by a constant amount. For each value of s_t , the same pulse sequence is executed and an FID is recorded until sufficient data is acquired. Thus recording the two-dimensional signal involves repeating a pulse sequence for increasing values of s_t and recording the FID as a function of t for each value of s_t . Note that the signal is not detected during the time s_t . The FID is recorded only in the detection section as a function of t . The corresponding dimension with time variable t for detecting the FID is called the acquisition dimension, or the direct dimension. By contrast, another dimension with discrete time variable s_t , for each value of which a pulse sequence is repeated, is called the indirect dimension [17]. Most importantly, the signal eventually recorded during t is modulated by events occurring during the evolution time s_t . Thus the amplitude of the complex signal acquired during the detection section is proportional to $c_2 e^{(r_2 s_t)} e^{i\omega_2 s_t}$, instead of the constant c_1 as in

Equation (6.1), and depends parametrically on the value of s_t . Therefore the 2D NMR signal is a function of two time variables t and s_t and is represented as:

$$S(t, s_t) = \sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{r_{1k}t + r_{2l}s_t} e^{i(\omega_{1k}t + \omega_{2l}s_t + \phi_{kl})} + \epsilon(t, s_t),$$

where separate decay constants, angular frequencies and phases are required for each dimension, reflecting the differing behavior of the nuclear magnetization during the different evolution sections [17][91].

A device known as an analogue to digital converter or ADC is used to convert the NMR signal, after it was amplified, from the voltage to binary numbers which can be stored as a matrix in the computer. The ADC samples the signal resulting in a representation of the NMR signal as data points.

The data of a 2D NMR experiment is therefore described as ([91]-[93])

$$\begin{aligned} s(n, m) &= \sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{i\phi_{kl}} e^{(i\omega_{1k} + r_{1k})t_n + (i\omega_{2l} + r_{2l})s_m} + \epsilon(n, m) \\ &= \sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{r_{1k}t_n + r_{2l}s_m} e^{i(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl})} + \epsilon(n, m) \\ &= \sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{r_{1k}t_n + r_{2l}s_m} \cos(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) + \text{Re}(\epsilon(n, m)) \\ &\quad + i\left(\sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{r_{1k}t_n + r_{2l}s_m} \sin(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) + \text{Im}(\epsilon(n, m))\right), \end{aligned}$$

with $0 \leq t_1 < t_2 < \dots < t_n < \dots < t_N$, $1 \leq n \leq N$, and $0 \leq s_1 < s_2 < \dots < s_m < \dots < s_M$, $1 \leq m \leq M$. Here c_{kl} , ω_{1k} , ω_{2l} , r_{1k} , r_{2l} and ϕ_{kl} are real constants. They are respectively the amplitudes, the angular frequencies, the damping factors and the phases. Each column of the data matrix represents data collected for an FID at a fixed s value. We now specify the assumptions on the data sets that we are considering. The noise component $\epsilon(n, m)$ is assumed to be complex Gaussian, with zero mean. The real and imaginary parts are assumed to have variance σ_m^2 , and to be independent/uncorrelated, i.e. $\text{var}(\text{Re}(\epsilon(n, m))) = \text{var}(\text{Im}(\epsilon(n, m))) = \sigma_m^2$ and $E(\text{Re}(\epsilon(n, m))\text{Im}(\epsilon(n, m))) = 0$.

The noise free signal is denoted by

$$\begin{aligned} g(n, m) &:= \sum_{k=1}^K \sum_{l=1}^L c_{kl} e^{r_{1k} t_n + r_{2l} s_m} e^{i(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl})} \\ &= s(n, m) - \epsilon(n, m). \end{aligned}$$

Note that we have not assumed that the variance of the noise in the indirect dimension is identical for each increment. This will be analyzed further in Section 6.4. We also allow for non-uniform sampling in our setup both in the direct and indirect dimensions.

Using the well-known description of the probability distribution function of Gaussian noise we can write down the probability distribution function $p(x, \Theta)$ (Θ is the parameter vector) of an acquired sample $(x(n, m))_{1 \leq n \leq N; 1 \leq m \leq M}$ of the data as

$$\begin{aligned} p(x, \Theta) &= \prod_{n=1}^N \prod_{m=1}^M \frac{1}{\sqrt{2\pi\sigma_m^2}} e^{-\frac{1}{2\sigma_m^2} (Re(x(n, m)) - Re(g(n, m)))^2} \\ &\quad \cdot \frac{1}{\sqrt{2\pi\sigma_m^2}} e^{-\frac{1}{2\sigma_m^2} (Im(x(n, m)) - Im(g(n, m)))^2}. \end{aligned}$$

The log-likelihood function is therefore

$$\begin{aligned} \ln(p(x, \Theta)) &= \sum_{n=1}^N \sum_{m=1}^M [-\ln(2\pi) - 2\ln(\sigma_m) \\ &\quad - \frac{1}{2\sigma_m^2} (Re(x(n, m)) - Re(g(n, m)))^2 \\ &\quad - \frac{1}{2\sigma_m^2} (Im(x(n, m)) - Im(g(n, m)))^2]. \end{aligned}$$

The *Fisher information matrix* is defined as

$$F(\Theta) = \left[-E \frac{\partial^2 \ln p(x, \Theta)}{\partial \theta_i \partial \theta_j} \right]_{1 \leq i, j \leq P},$$

where E stands for taking the expectation and the parameter vector Θ is given by

$$\Theta = \begin{pmatrix} \theta_1 & \theta_2 & \cdots & \theta_P \end{pmatrix}.$$

For any unbiased estimator $\hat{\Theta} = (\hat{\theta}_1 \ \hat{\theta}_2 \ \cdots \ \hat{\theta}_P)$ of the parameter vector Θ we then have that the *Cramer Rao Lower Bound* is given by (see e.g. [94])

$$\text{var}(\hat{\theta}_i) \geq [F^{-1}(\Theta)]_{ii}.$$

An important result of this chapter is that the entries of the Fisher information matrix can be computed analytically for the 2D NMR data given above. This result is presented in Section 6.7, Appendix A.

In Appendix A.1, we also show how to derive several of the entries of the Fisher information matrix. The results presented in the Appendix address the relatively general case for 2D NMR experiments that was introduced above. Specific practical situations may require a somewhat different setting. Expressions for the higher dimensional NMR experiments can be derived analogously. Similarly, for example, if certain parameters are known, this information can be included in a relatively straightforward fashion. Another situation that may present itself is when the parameters that are discussed here are not the ones that are being sought but a transformation of these parameters is the actual interest of the experiment. For example, it may be a frequency shift that is to be estimated rather than the actual frequency. In this case a standard transformation result is of use (see e.g. [94]).

An Example

The above derived expressions of the Fisher information matrix for the general data model are rather lengthy. To illustrate the results we therefore specialize the expressions for a relatively simple situation. For the special case of one (not necessarily identical) resonance in the direct and indirect dimensions, i.e. the case $K = L = 1$, the data is given by

$$\begin{aligned}
 s(n, m) &= c_{11} e^{i\phi_{11}} e^{(i\omega_{11} + r_{11})t_n + (i\omega_{21} + r_{21})s_m} + \epsilon(n, m) \\
 &= c_{11} e^{r_{11}t_n + r_{21}s_m} e^{i(\omega_{11}t_n + \omega_{21}s_m + \phi_{11})} + \epsilon(n, m) \\
 &= c_{11} e^{r_{11}t_n + r_{21}s_m} \cos(\omega_{11}t_n + \omega_{21}s_m + \phi_{11}) \\
 &\quad + \operatorname{Re}(\epsilon(n, m)) \\
 &\quad + i(c_{11} e^{r_{11}t_n + r_{21}s_m} \sin(\omega_{11}t_n + \omega_{21}s_m + \phi_{11}) \\
 &\quad + \operatorname{Im}(\epsilon(n, m))).
 \end{aligned}$$

We assume that all six parameters c_{11} , r_{11} , r_{21} , ω_{11} , ω_{21} and ϕ_{11} are unknown.

Specializing the formulae in Section 6.7.2 we can obtain the entries of the 6×6 Fisher information matrix $F(\Theta)$. If the parameter vector Θ is given by

$$\Theta = \left(c_{11}, r_{11}, r_{21}, \omega_{11}, \omega_{21}, \phi_{11} \right)$$

then the entries of the symmetric Fisher information matrix are

$$\begin{aligned}
F(1,1) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{11}^2} = \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} e^{2(r_{11}t_n + r_{21}s_m)}, \\
F(2,2) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{11}^2} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} t_n^2, \\
F(3,3) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{21}^2} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} s_m^2, \\
F(4,4) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{11}^2} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} t_n^2, \\
F(5,5) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{21}^2} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} s_m^2, \\
F(6,6) &= -E \frac{\partial \ln^2(p(x, \Theta))}{\partial \phi_{11}^2} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)}, \\
F(1,2) &= F(2,1) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial c_{11} \partial r_{11}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11} e^{2(r_{11}t_n + r_{21}s_m)} t_n, \\
F(1,3) &= F(3,1) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial c_{11} \partial r_{21}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11} e^{2(r_{11}t_n + r_{21}s_m)} s_m, \\
F(1,4) &= F(4,1) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial c_{11} \partial \omega_{11}} = 0, \\
F(1,5) &= F(5,1) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial c_{11} \partial \omega_{21}} = 0,
\end{aligned}$$

$$\begin{aligned}
F(1, 6) &= F(6, 1) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial c_{11} \partial \phi_{11}} = 0, \\
F(2, 3) &= F(3, 2) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{11} \partial r_{21}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} t_n s_m, \\
F(2, 4) &= F(4, 2) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{11} \partial \omega_{11}} = 0, \\
F(2, 5) &= F(5, 2) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{11} \partial \omega_{21}} = 0, \\
F(2, 6) &= F(6, 2) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{11} \partial \phi_{11}} = 0, \\
F(3, 4) &= F(4, 3) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{21} \partial \omega_{11}} = 0, \\
F(3, 5) &= F(5, 3) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{21} \partial \omega_{21}} = 0, \\
F(3, 6) &= F(6, 3) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{21} \partial \phi_{11}} = 0, \\
F(4, 5) &= F(5, 4) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial \omega_{11} \partial \omega_{21}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} t_n s_m, \\
F(4, 6) &= F(6, 4) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial \omega_{11} \partial \phi_{11}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} t_n, \\
F(5, 6) &= F(6, 5) = -E \frac{\partial \ln^2 p(x, \Theta)}{\partial \omega_{21} \partial \phi_{11}} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{11}^2 e^{2(r_{11}t_n + r_{21}s_m)} s_m.
\end{aligned}$$

Note that in fact the Fisher information matrix has a number of repeated entries. If we set

$$\begin{aligned}
a &:= F(1, 1); \quad b := F(2, 1), \quad c := F(1, 3), \\
d &:= F(2, 2), \quad e := F(2, 3), \quad f := F(3, 3), \\
g &:= F(4, 6), \quad h := F(5, 6), \quad z := F(6, 6),
\end{aligned}$$

we have that the Fisher information matrix $F(\Theta)$ is given by

$$F(\Theta) = \begin{bmatrix} a & b & c & 0 & 0 & 0 \\ b & d & e & 0 & 0 & 0 \\ c & e & f & 0 & 0 & 0 \\ 0 & 0 & 0 & d & e & g \\ 0 & 0 & 0 & e & f & h \\ 0 & 0 & 0 & g & h & z \end{bmatrix}.$$

Note that this matrix is block diagonal, which has interesting consequences for the Cramer Rao Lower Bound. The Cramer Rao Lower Bound is given by the diagonal elements of the inverse of $F(\Theta)$,

$$CRLB_i = [F(\Theta)^{-1}]_{ii} = \begin{bmatrix} R & 0 \\ 0 & Q \end{bmatrix}_{ii},$$

where

$$R = \frac{1}{-ae^2 + 2bce - b^2f - c^2d + afd} \cdot \begin{bmatrix} -e^2 + fd & ce - bf & -cd + eb \\ ce - bf & -c^2 + af & bc - ae \\ -cd + eb & bc - ae & -b^2 + ad \end{bmatrix}$$

$$Q = \frac{1}{dfz - dh^2 - e^2z + 2geh - g^2h} \cdot \begin{bmatrix} fz - h^2 & -ez + gh & eh - gf \\ -ez + gh & dz - g^2 & ge - dh \\ eh - gf & ge - dh & df - e^2 \end{bmatrix}.$$

6.3 Cramer Rao Lower Bound and Parameter Estimation

As was discussed in Section 6.2 the Cramer Rao Lower Bound provides a lower bound for the variance of any parameter estimate given by an unbiased estimation procedure. One of the ways in which this bound can be used is to evaluate the performance of an estimation algorithm. Clearly, by the fundamental result the variance of an unbiased estimator cannot be lower than the Cramer Rao Lower Bound. However, it is not a priori obvious that there are algorithms that attain this bound. We will illustrate this with

an example that a least-square estimation procedure attains, or at least comes close to attaining, the bound.

Consider the case of two (not necessarily identical) resonances in the direct and indirect dimensions respectively, i.e. the case $K = L = 2$, for which the data is given by

$$s(n, m) = \sum_{k=1}^2 \sum_{l=1}^2 c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \cos(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}) + \text{Re}(\epsilon(n, m)) \\ + i \left(\sum_{k=1}^2 \sum_{l=1}^2 c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \sin(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}) + \text{Im}(\epsilon(n, m)) \right). \quad (6.2)$$

For this simulation example, we assume that all the sixteen parameters are ordered into the following parameter vector

$$\Theta = \left(c_{11}, c_{12}, c_{21}, c_{22}, r_{11}, r_{12}, r_{21}, r_{22}, \omega_{11}, \omega_{12}, \right. \\ \left. \omega_{21}, \omega_{22}, \phi_{11}, \phi_{12}, \phi_{21}, \phi_{22} \right).$$

For the purpose of illustration, we fix the values of the parameter vector as

$$\Theta^* = \left(.15, .22, .12, .13, -.1, -.35, -.15, -.45, \right. \\ \left. 1.445, 2.136, 2.702, .88, .683, 1.366, 2.4167, .982 \right).$$

Gaussian noise is then added to the signal. The noisy signal is uniformly sampled with 1024 samples in the direct and 32 samples in the indirect dimension (see Figure 6.1 for a plot of a trace of the noisy 2D data set with a -20 dB noise level). To demonstrate the validity of the Cramer Rao Lower Bound, we choose five different noise levels $(-10, -20, -30, -40, -50)$ dB respectively. The noise level is defined as follows. If the simulated Gaussian white noise has zero mean and variance σ^2 then the noise level in dB is defined by $10 \log_{10}(\sigma^2) = 20 \log_{10}(\sigma)$. When the noise level is at 0 dB, $\sigma^2 = 1$. Hence, the reference noise is the Gaussian white noise with zero mean and variance 1 and the reference noise level is 0 dB.

For each noise level, we estimate the parameters from the simulated data using a nonlinear least-squares estimation procedure, and the number of Monte-Carlo runs is set as fifty. When running the nonlinear least-squares optimization algorithm, we start with a random

Figure 6.1: The real part of the first trace of a 1024×32 2D data set with noise level at -20 dB.

initial estimate vector in the neighborhood of the real parameter vector. The least-squares optimization routine was coded in Matlab [84] using the optimization toolbox in a straightforward way.

In Figure 6.2 we show a comparison of the bound for the variance obtained from the Cramer Rao Lower Bound for the parameter ω_{11} and the estimated parameter based on the least-squares estimates. The comparison is shown for the five noise levels $-10, -20, -30, -40, -50$ dB respectively. The plot shows a high level of agreement. Full agreement cannot be expected since finite sample effects are unavoidable for the estimated variances. These finite sample effects could also result in the estimated variance being even lower than the CRLB. The plots also show that the variance of the estimates decreases with decreasing noise level of the data.

In [95], the full 16×16 matrix for the CRLB for a noise level of -20 dB for the above example was listed, together with the estimated variance matrix based on the simulations and estimation procedure discussed above. The agreement between the two matrices is good, also taking into consideration the finite sample effects that were discussed above.

Figure 6.2: Comparison of lower bound obtained for CRLB for the variance estimate of ω_{11} (solid line 'o-') with estimated variance based on nonlinear least-squares parameter estimate ('x') for five noise levels ($-10, -20, -30, -40, -50$ dB).

6.4 Experiment Design Using the Cramer Rao Lower Bound

In the previous section we have seen that the Cramer Rao Lower Bound is essentially attained by a least-squares parameter estimation method. In this section we investigate how the Cramer Rao Lower Bound can be used to give insights into how the experimental design influences the accuracy with which the parameters can be estimated. We follow the approach discussed earlier that a reduced variance of an unbiased parameter estimate is interpreted as a more accurate estimate. An experimental design can then be analyzed by computing the Cramer Rao Lower Bound for the data set that will result from the implementation of the experimental design. A decreased Cramer Rao Lower Bound would then lead to a parameter estimate that has increased accuracy assuming that a parameter estimation method is available that attains the bound, as was shown to exist for the example discussed in Section 6.3.

In the two subsections below, we will look at two examples with such considerations. The first shows how the CRLB could be used to evaluate whether or not non-uniform

sampling might improve the accuracy of the estimates. The second shows how to evaluate averaging schemes of traces in 2D experiments. It should be pointed out that our main concern is not to advocate certain experimental designs, such as non-uniform sampling over uniform sampling. Our interest is to provide a tool to the experimentalist that allows the experimentalist to decide in quantitative terms which experimental design is superior from the point of view of minimizing the Cramer Rao Lower Bound.

These two applications are only a small number of the potential uses of the Cramer Rao Lower Bound to evaluate experimental designs. They should, however, give a good indication for the type of analysis that is possible. For ease of presentation we again use a relatively simple data model. The same approach could be used to investigate more general data models.

6.4.1 Non-uniform Sampling

The question as to whether or not non-uniform sampling is to be preferred over uniform sampling has been addressed by various authors (see e.g. [91] for a textbook treatment). Here we do not wish to advocate a new sampling scheme. However, we would like to show how our results on the CRLB can be used to evaluate various schemes.

We again consider the 2D example of Section 6.3. For this example we are going to consider uniform and non-uniform sampling in the direct and indirect dimensions. We first consider the sampling process in the indirect dimension. For the uniform sampling scheme the following 16 sampling points are chosen with sampling interval $\Delta t = 1.54$ as $\left(\begin{array}{cccccccccccccccc} 0.00 & 1.54 & 3.07 & 4.61 & 6.14 & 7.68 & 9.21 & 10.75 & 12.28 & 13.82 & 15.35 & 16.89 & 18.42 & 19.96 & 21.49 \\ 23.03 \end{array} \right)$.

The non-uniform sampling points have been determined following an exponential scheme discussed in [91] with

$$t_{j+1} = -\frac{e^{-Lt_j} - \frac{1-e^{-LT}}{M-1}}{L}, \quad j = 1, 2, \dots, M-1$$

where $t_1 = 0$, $L = 0.1$, $T = 23.03$ and $M = 16$ for getting the non-uniform sampling points in the indirect dimension in this example. These samples are given by

$$\begin{pmatrix} 0.00 & 0.62 & 1.28 & 1.98 & 2.74 & 3.57 & 4.46 & 5.45 & 6.54 & 7.77 & 9.16 & 10.79 & 12.73 & 15.14 & 18.33 \\ 23.03 \end{pmatrix}.$$

In the direct dimension the 1024 sampling points were sampled in the uniform case at a sampling interval of $\Delta s = 0.015$. For the non-uniform sampling scheme the data points were generated using an exponential scheme given by a formula similar to the above one for t_{j+1} except that now $L = 1.15$, $T = 15.35$ and $M = 1024$.

In Table 6.1 the results of the corresponding Cramer Rao Lower Bound calculations are summarized. It follows clearly that by introducing the above discussed non-uniform sampling the CRLB is generally decreased. Ignoring other possible considerations such as ease of parameter estimation, this shows that the experimental design based on using the given non-uniform sampling schemes for both dimensions will produce the best data amongst the three experimental designs that were analyzed. The term 'best data set' is used here to indicate the data set that allows for the most precise estimation of the underlying parameters as assessed by the CRLB, amongst the data sets that are being considered. It might be intuitively clear that the presented non-uniform sampling scheme is superior to the uniform for the given data set.

The value of using the Cramer Rao Lower Bound to evaluate the different sampling strategies lies in the fact that a precise quantitative evaluation for the various strategies is possible. In practice the spectroscopist would have to weigh the established benefits of the non-uniform sampling scheme against other considerations that are not addressed by the Cramer Rao Lower Bound such as the fact that non-uniformly sampled data cannot be processed by the standard Fourier transform based methods to obtain a spectrum.

6.4.2 Averaging to Improve Parameter Estimation Accuracy

Experimental time is at a premium in NMR spectroscopy, in particular for multidimensional experiments. The experimental time is directly proportional to the number of scans that are acquired. The signal to noise ratio is often improved by averaging scans. This amounts to changing the variance σ_m^2 in the noise description. Note that if σ_m^2 is constant for all m , i.e. $\sigma^2 := \sigma_m^2$, for $m = 1, 2, \dots$, it follows from the expressions for the Cramer Rao Lower Bound in the Appendix that the Cramer Rao Lower Bound is proportional to σ^2 . This shows that uniformly lowering the noise level will lead to an improvement in the bound on the variance of the parameter estimates by an equal amount.

The question that will be addressed now *to illustrate the potential use of the Cramer Rao Lower Bound* is whether it can, for example, be beneficial to average more scans for low increment numbers rather than averaging the same number of scans for each increment. In Table 6.2 the results of CRLB calculations are summarized for our standard 2D example system (see Section 6.3) with 16 being the size of the indirect dimension. Two averaging schemes were considered. The first is the ‘traditional’ scheme in which for each increment in the indirect dimension a fixed number of repeated experiments (in our case 16) are acquired and then averaged. Since in our example the indirect dimension has

	$u u$	$n u$	$n n$
c_{11}	5.923e-05	4.534e-05	2.655e-05
c_{12}	1.220e-04	8.709e-05	4.619e-05
c_{21}	8.925e-05	7.332e-05	4.244e-05
c_{22}	1.852e-04	1.417e-04	7.341e-05
r_{11}	1.842e-05	2.355e-05	1.147e-05
r_{12}	1.459e-03	1.167e-03	6.827e-04
r_{21}	7.548e-05	5.347e-05	6.197e-05
r_{22}	4.580e-04	2.715e-04	1.834e-04
ω_{11}	1.842e-05	2.355e-05	1.147e-05
ω_{12}	1.459e-03	1.167e-03	6.827e-04
ω_{21}	7.548e-05	5.347e-05	6.197e-05
ω_{22}	4.580e-04	2.715e-04	1.834e-04
θ_{11}	2.632e-03	2.015e-03	1.180e-03
θ_{12}	2.521e-03	1.799e-03	9.543e-04
θ_{21}	6.198e-03	5.092e-03	2.947e-03
θ_{22}	1.096e-02	8.385e-03	4.344e-03

Table 6.1: CRLB for Different Sampling Methods. $u|u$ ($n|n$) stands for (non-)uniform sampling in both dimensions; $n|u$ stands for non-uniform sampling in the indirect dimension and uniform sampling in the direct dimension. The precise sampling scheme is explained in the text.

16 increments this amounts to a total of 256 acquired traces. In the second scheme we propose to acquire an unequal number of traces per increment. In this ‘non-uniform’ averaging scheme we propose to again acquire 256 scans for the 2D experiment but we acquire and average 29 traces for the first increment, 27 traces for the second increment, 26 for the third, etc. The list of numbers of traces for all the increments is given by 29,27,26,24,22,20,19,17,15,13,12,10,8,6,5,3. In the noise description for a 2D data set we allowed each ‘column’ $s(n, m)_{n \geq 1}$ to have a different variance σ_m^2 , $m \geq 1$. This means that our proposed averaging scheme can be easily incorporated in the general data model. If an acquired trace has variance σ^2 then the average of k traces will have variance $\frac{\sigma^2}{k}$. This means for example, that in our case the variance σ_1^2 for the first increment will be $\sigma_1^2 = \frac{\sigma^2}{29}$.

Four different experimental designs have been considered, in which uniform averaging and non-uniform averaging have been combined with uniform and non-uniform sampling. The results in Table 6.2 show that a combination of the above discussed non-uniform averaging scheme with the non-uniform sampling schemes introduced in Section 6.4.1 produces a reduction in the CRLB for the various parameters by a factor of about 3.

An issue that needs to be addressed in the context of experimental design is that in practice the parameters of a system are often not known and the purpose of the experiment is to estimate these unknown parameters. We believe that the results presented can nevertheless be of importance in practice. First, in many situations the experimentalist has a relatively good knowledge of the ranges within which the parameters can be expected to lie. An experimental design can then be evaluated using ‘expected’ parameters. Such an approach would give the experimentalist a good indication about which parameters are difficult and which are less difficult to estimate. Moreover, a preliminary experiment could be carried out to obtain this initial information. Based on the preliminary results a careful experimental design could be developed for a full experiment that is aimed to very precisely estimate the required parameters.

	u u UA	n u UA	n u NA	n n NA
c_{11}	3.702e-06	2.834e-06	2.569e-06	1.471e-06
c_{12}	7.627e-06	5.443e-06	4.680e-06	2.483e-06
c_{21}	5.578e-06	4.583e-06	3.677e-06	2.096e-06
c_{22}	1.158e-05	8.856e-06	6.894e-06	3.597e-06
r_{11}	1.151e-06	1.472e-06	2.184e-06	1.062e-06
r_{12}	9.118e-05	7.296e-05	6.683e-05	3.900e-05
r_{21}	4.718e-06	3.342e-06	2.440e-06	2.829e-06
r_{22}	2.863e-05	1.697e-05	1.113e-05	7.524e-06
ω_{11}	1.151e-06	1.472e-06	2.184e-06	1.062e-06
ω_{12}	9.118e-05	7.296e-05	6.683e-05	3.900e-05
ω_{21}	4.718e-06	3.342e-06	2.440e-06	2.829e-06
ω_{22}	2.863e-05	1.697e-05	1.113e-05	7.524e-06
θ_{11}	1.645e-04	1.259e-04	1.142e-04	6.540e-05
θ_{12}	1.576e-04	1.125e-04	9.670e-05	5.130e-05
θ_{21}	3.874e-04	3.182e-04	2.553e-04	1.456e-04
θ_{22}	6.850e-04	5.240e-04	4.079e-04	2.129e-04

Table 6.2: CRLB for Different Sampling and Weighting Methods. $u|u$ ($n|n$) stands for (non-)uniform sampling in both dimensions; $n|u$ stands for non-uniform sampling in the indirect dimension and uniform sampling in the direct dimension. UA (NA) stands for (non-)uniform averaging in the first dimension. The precise sampling and weighting scheme is explained in the text.

6.5 Global Optimization in Parameter Estimation of 2D NMR Data

Parameter estimation is used to estimate the values of a group of parameters from a given set of observations. It is an interesting and challenging problem that is important in a variety of different fields and applications. One major difficulty of this problem is that the data are often corrupted by noise or contaminated with interfering signals.

The processing and analysis of experimental data that contain the information of the resonance spectrum is a crucial step in NMR spectroscopy. This is particularly so for multidimensional NMR experiments as they are used, for example, for the determination of biomolecular structure.

At early stages of NMR spectroscopy analysis, the application of discrete Fourier transform (DFT) has made great progress in this subject field. Consequently, many variations of this conventional Fourier approach have been proposed in order to deal with some of its shortcomings. Nevertheless the frequency resolution of the Fourier transform based algorithms is limited by the short acquisition time of the NMR data and measurement noise. Hence, to improve the resolution, many model-based methods have been proposed for the parameter estimation of NMR spectroscopy [99]-[103]. Among them, the Kumaresan-Tufts (KT) algorithm [99] is one of the most effective algorithms and the autoregressive modeling of NMR data is one of the most commonly used algorithms in the analysis of 1-D NMR data [100]. In particular, it has been shown that these methods based on the linear prediction technique can estimate parameters more accurately than the standard DFT methods. Unfortunately, a well-recognized disadvantage of the linear prediction method is that it can be very sensitive to noise in the data.

In recent years, the spectral estimation of multidimensional systems has attracted increasing attention and therefore many new algorithms have been proposed. Most of research literature only present estimators for part of the parameters, such as frequency [104, 105], or focus on low dimensional systems [104]-[106].

Global optimization is another technique which can be used to estimate the parameters in NMR Data. It predicts the optimum operating conditions of a system so that some performance criterion is satisfied. In mathematics the performance criterion could be, for example, to minimize the integral of the squared difference between a specified function and an approximation to it generated as a function of the controlled parameters. The importance of global optimization lies not in trying to find out all about a system but in finding out, with the least possible effort, the best way to adjust the system. If this is carried out well, systems can have a more economic and improved design, they can operate more accurately or at less cost. During the last three decades, many global

optimization methods have been used to explore the local region of operation and predict the way that the system parameters should be adjusted to bring the system to optimum. The global optimization algorithm uses the available information to evaluate the best estimated value of the parameters, which can in some way be defined as optimal, in the sense of minimizing the objective function. In a simple case this may be the minimization or maximization of some system characteristic that is dependent on parameters $\mathbf{x}$. In a more advanced formulation the objective function, $f(\mathbf{x})$, to be minimized or maximized, may be subject to constraints in the form of equality constraints, inequality constraints, and/or parameter bounds.

Multidimensional NMR data can be modeled as the sum of several multidimensional exponentially damped sinusoids [16, 96]. The main aim of this section is to estimate the parameters of 2D exponentially damped sinusoids in simulated NMR signal by ripple search strategy. The expressions for CRLB are also computed numerically and shown by computer simulations to match with the variance of NMR spectroscopy parameters estimated by our proposed global optimization algorithm.

I. Ripple Search Strategy in NMR Data Processing

The objective function is set up using least-squares error criterion defined by [107]

$$\text{minimize } E = \sum_{m=1}^M \sum_{n=1}^N \{w(m, n) f_{(m, n)}(\mathbf{X})^2\} \quad (6.3)$$

Here, $w(m, n) = \frac{1}{2}$, $f_{(m, n)}(\mathbf{X}) = s(n, m) - g(n, m)$, where $s(n, m)$ is the sample data buried in Gaussian noise, $g(n, m)$ is the pure signal with unknown parameter vector $\mathbf{X}$.

The test data for 2D NMR signal in the case of $K = L = 2$ are shown in Table 6.3. The number of parameters is sixteen. For this simulation, we assume that all the sixteen parameters are ordered into the following parameter vector:

$$\Theta = \left(c_{11}, c_{12}, c_{21}, c_{22}, r_{11}, r_{12}, r_{21}, r_{22}, \omega_{11}, \omega_{12}, \right. \\ \left. \omega_{21}, \omega_{22}, \phi_{11}, \phi_{12}, \phi_{21}, \phi_{22} \right).$$

The size of the sample data is 32×16 by uniform sampling. For the purpose of illustration,

we fix the values of the parameter vector as

$$\Theta^* = \left(.15, .22, .12, .13, -.1, -.35, -.15, -.45, \right. \\ \left. 1.445, 2.136, 2.702, .88, .683, 1.366, 2.4167, .982 \right).$$

For comparison purpose with ASA and DEA on efficiency and accuracy, we choose noiseless case.

Table 6.3: Simulation Results and Comparisons for Noiseless 2D NMR Data

		Ripple Search Strategy	ASA	DEA
True Parameters		Estimated Parameters	Estimated Parameters	Estimated Parameters
C_{11}	0.15	0.1499	0.1500724	0.1498
C_{12}	0.22	0.2200	0.2200385	0.2199
C_{21}	0.12	0.1198	0.120103	0.1199
C_{22}	0.13	0.1299	0.1300341	0.1300
R_{11}	-0.1	-0.1000	-0.1000251	-0.1001
R_{12}	-0.35	-0.3499	-0.3501441	-0.3501
R_{21}	-0.15	-0.1500	-0.1500537	-0.1497
R_{22}	-0.45	-0.4501	-0.4499869	-0.4496
W_{11}	1.445	1.4451	1.445131	1.4452
W_{12}	2.136	2.1360	2.136118	2.1389
W_{21}	2.701	2.7016	2.701756	2.7013
W_{22}	0.88	0.8801	0.8797056	0.8797
Φ_{11}	0.683	0.6830	0.6830617	0.6850
Φ_{12}	1.366	1.3660	1.365683	1.3660
Φ_{21}	2.4167	2.4157	2.417126	2.4113
Φ_{22}	0.982	0.9822	0.9823125	0.9787
Global function values		3.6976e-9	6.043e-9	9.8059e-7
No. of Function Evaluations		39705	60000	46599

The control parameters for ripple search strategy are set as $m_0 = 50$, $\Delta r = 0.8$, $p_a = 2$. For the results of ASA, the setting of three most important control parameters is as follows: $TRA = 1e-94$, $TAS = 1e10$ and $IT = 10$. For the results of DEA, the setting of three essential control parameters, the population size NP , amplifying differential factor F and crossover constant CR , are 30, 0.97 and 0.98 respectively. It is seen that global

minimum from a random starting point is found by ripple search strategy. The results obtained by our proposed method are better than those of ASA and DEA. Comparing the number of function evaluations, it is obvious that the ripple search strategy is more efficient than the ASA and DEA under the same test conditions for this problem.

II. Variances of Estimated Parameters Compared with CRLB

Detailed simulation conditions are:

1. The sample signal for simulation is 2D exponentially damped sinusoids in NMR spectroscopy with $K = L = 2$, where K and L are the numbers of exponentially damped sinusoids. From Equation (6.2), in the model of the sample signal in Gaussian noise, there are sixteen parameters to be estimated. For this simulation, we assume that all the sixteen parameters are ordered into the following parameter vector

$$\Theta = \left(c_{11}, c_{12}, c_{21}, c_{22}, r_{11}, r_{12}, r_{21}, r_{22}, \omega_{11}, \omega_{12}, \right. \\ \left. \omega_{21}, \omega_{22}, \phi_{11}, \phi_{12}, \phi_{21}, \phi_{22} \right).$$

For the purpose of illustration, we fix the values of the parameter vector as

$$\Theta^* = \left(.12, .13, .15, .22, -.23, -.20, -.15, -.10, \right. \\ \left. .8796, 1.4451, 2.1363, 2.7018, .683, .9817, 1.3659, 2.4166 \right).$$

Gaussian noise is then added to the signal. The noisy signal is uniformly sampled with 32 samples in the direct and 16 samples in the indirect dimension.

2. To model (6.2), by prior knowledge the bounds for each parameter are,

$$c_{kl} \geq 0$$

$$r_{1k} \leq 0$$

$$r_{2l} \leq 0$$

$$0 \leq \omega_{kl} \leq \pi$$

$$0 \leq \phi_{kl} \leq 2\pi$$

3. The number of Monte-Carlo simulation runs is set to be 10.

4. Random initial points generated inside parameter bounds are used for estimation;

The simulation result for the parameter ω_{22} is shown in Figure 6.3. Results for other parameters are similar, hence they are not included. The performance of our proposed method is illustrated by Monte-Carlo simulations and is compared with the CRLB. In the figure, the scale of x axes is the values of SNR series from lower to higher; the scale for y axes is the variances of the estimate. The solid line with marker 'o' describes the CRLB, while the dashed with marker '*' demonstrates the variances of the estimated parameter. Clearly, the accuracy of the simulation results by the proposed method increases with the SNR and is close to the CRLB. Additionally, slight changes in the signal, such as the change of SNR, have a small effect on the accuracy of the estimates, thus the ripple search method is a kind of robust methods [111]. Furthermore, our method is especially suitable to those cases with small SNR and less sample data, in other words, the NMR data.

Figure 6.3: CRLB (solid) and the estimated variance (dashed with '*' marker) for parameter ω_{22} at different SNR.

6.6 Conclusions

Classical statistical results show that the Cramer Rao Lower Bound provides a lower bound for the variance of a parameter estimation scheme, provided it is unbiased, i.e. provided that on average it produces the correct results. Here we have shown that the CRLB can be calculated explicitly for a parameter estimation problem for a general 2D NMR data set. Using Monte Carlo type simulations we have shown for a simulated example that the variance of estimates using a nonlinear least-squares parameter estimation algorithm in fact attain this bound. It was also shown how the developed results can be applied to study questions of experimental design. Using example systems we used our CRLB results to show that a certain non-uniform sampling scheme in fact produces superior results over the uniform sampling scheme that was considered. We also introduced (for the first time to the best of our knowledge) a non-uniform averaging scheme for traces. This was also investigated in conjunction with non-uniform sampling to show that an improved CRLB can be obtained for the examined system.

Additionally in this chapter, ripple search strategy is applied to estimate all the parameters of NMR data starting from a random initial point. It can estimate the signal parameters under low signal-to-noise ratio (SNR) by using few data points which is of particular significance for multidimensional NMR experimentation, because any savings in data in the indirect dimensions can lead to major savings of expensive experimental time. The effectiveness and performance of the method are confirmed by computer simulations and are compared with CRLB and other global optimization methods. The proposed method can be used straightforwardly to the parameter estimation of multidimensional NMR data.

6.7 Appendix A. The Entries of the Fisher Information Matrix

6.7.1 A.1. Derivation of Entries of Fisher Information Matrix

We are here going to show how to derive the entries for a few cases. The other entries are derived similarly and the results are listed below.

To compute the entries of the Fisher information matrix the second order partial derivatives of $\ln p(x, \Theta)$ have to be calculated. The first order partial derivatives are given by,

$$\begin{aligned}
 \frac{\partial \ln(p(x, \Theta))}{\partial c_{kl}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) e^{r_{1k}t_n + r_{2l}s_m} \cos(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) \\
 &\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) e^{r_{1k}t_n + r_{2l}s_m} \sin(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}), \\
 \\
 \frac{\partial \ln(p(x, \Theta))}{\partial r_{1k}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) \sum_{v=1}^L c_{kv} e^{r_{1k}t_n + r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) t_n \\
 &\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) \sum_{v=1}^L c_{kv} e^{r_{1k}t_n + r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) t_n \\
 &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k}t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
 &\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k}t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}), \\
 \\
 \frac{\partial \ln(p(x, \Theta))}{\partial r_{2l}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) s_m e^{r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
 &\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) s_m e^{r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}), \\
 \\
 \frac{\partial \ln(p(x, \Theta))}{\partial \omega_{1k}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k}t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
 &\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k}t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}),
 \end{aligned}$$

$$\begin{aligned}
& \frac{\partial \ln(p(x, \Theta))}{\partial \omega_{2l}} \\
&= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) s_m e^{r_{2l} s_m} \sum_{u=1}^K c_{ul} e^{r_{1u} t_n} \sin(\omega_{1u} t_n + \omega_{2l} s_m + \phi_{ul}) \\
&+ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) s_m e^{r_{2l} s_m} \sum_{u=1}^K c_{ul} e^{r_{1u} t_n} \cos(\omega_{1u} t_n + \omega_{2l} s_m + \phi_{ul}),
\end{aligned}$$

$$\begin{aligned}
& \frac{\partial \ln(p(x, \Theta))}{\partial \phi_{kl}} \\
&= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \sin(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}) \\
&+ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \cos(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}),
\end{aligned}$$

Based on the above first order partial derivatives, we can continue to derive all the second order partial derivatives. However, since the derivation is long and tedious, we only give a few examples here. In particular, as to be illustrated in the following, we have to derive separately second order partial derivatives such as $\frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k}^2}$ and $\frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}}$ ($k \neq p$) since they are different to each other. However, after taking the expectation as required for the Fisher information matrix, it is often possible to use a general expression. For example, although we are unable to arrive at a unified expression $\frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}}$ that would cover both cases $k = p$ and $k \neq p$, we are able to derive a unified expression $E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}}$ that does cover both cases $k = p$ and $k \neq p$.

$$\begin{aligned}
& \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k}^2} \\
&= \frac{\partial}{\partial r_{1k}} \left\{ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right. \\
&\quad \left. + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right\} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n \frac{\partial}{\partial r_{1k}} \{ e^{r_{1k} t_n} \} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial r_{1k}} \{ (Re(x(n, m)) - Re(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n \frac{\partial}{\partial r_{1k}} \{ e^{r_{1k} t_n} \} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial r_{1k}} \{ (Im(x(n, m)) - Im(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n^2 e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{2r_{1k} t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right)^2 \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n^2 e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{2r_{1k} t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right)^2,
\end{aligned}$$

On the other hand for $k \neq p$,

$$\begin{aligned}
& \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}} \\
&= \frac{\partial}{\partial r_{1p}} \left\{ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right. \\
&\quad \left. + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right\} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial r_{1p}} \{ (Re(x(n, m)) - Re(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial r_{1p}} \{ (Im(x(n, m)) - Im(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p}) t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v} s_m} \cos(\omega_{1p} t_n + \omega_{2v} s_m + \phi_{pv}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p}) t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v} s_m} \sin(\omega_{1p} t_n + \omega_{2v} s_m + \phi_{pv}) \right).
\end{aligned}$$

It is not possible to consider $\frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}}$ in a unified fashion for both cases $k = p$ and $k \neq p$.

By taking the expectation, we have for the case $p = k$ that

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k}^2} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{2r_{1k} t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right)^2 \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{2r_{1k} t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right)^2,
\end{aligned}$$

and for $k \neq p$

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \cos(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \sin(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right).
\end{aligned}$$

The above two expressions can be unified using a single expression:

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \cos(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k} + r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \sin(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right),
\end{aligned}$$

where k can be equal to p and also be different to p .

Two more second order partial derivatives are derived in the following.

$$\begin{aligned}
\frac{\partial \ln^2(p(x, \theta))}{\partial c_{kl}^2} &= \frac{\partial}{\partial c_{kl}} \left\{ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) e^{r_{1k}t_n + r_{2l}s_m} \cos(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) \right. \\
&\quad \left. + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) e^{r_{1k}t_n + r_{2l}s_m} \sin(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) \right\} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial c_{kl}} \{ (Re(x(n, m)) - Re(g(n, m))) \} e^{r_{1k}t_n + r_{2l}s_m} \cos(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial c_{kl}} \{ (Im(x(n, m)) - Im(g(n, m))) \} e^{r_{1k}t_n + r_{2l}s_m} \sin(\omega_{1k}t_n + \omega_{2l}s_m + \phi_{kl}) \\
&= - \sum_n \sum_m \frac{1}{\sigma_m^2} e^{2(r_{1k}n + r_{2l}m)} \cos^2(\omega_{1k}n + \omega_{2l}m + \phi_{kl}) \\
&\quad - \sum_n \sum_m \frac{1}{\sigma_m^2} e^{2(r_{1k}n + r_{2l}m)} \sin^2(\omega_{1k}n + \omega_{2l}m + \phi_{kl}) \\
&= - \sum_n \sum_m \frac{1}{\sigma_m^2} e^{2(r_{1k}n + r_{2l}m)}.
\end{aligned}$$

$$\begin{aligned}
& \frac{\partial \ln^2 p(x, \Theta)}{\partial r_{1k} \partial \omega_{2l}} \\
&= \frac{\partial}{\partial \omega_{2l}} \left\{ \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right. \\
&\quad \left. + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right\} \\
&= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n e^{r_{1k} t_n} \frac{\partial}{\partial \omega_{2l}} \left\{ \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right\} \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial \omega_{2l}} \{ (Re(x(n, m)) - Re(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n e^{r_{1k} t_n} \frac{\partial}{\partial \omega_{2l}} \left\{ \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \right\} \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} \frac{\partial}{\partial \omega_{2l}} \{ (Im(x(n, m)) - Im(g(n, m))) \} t_n e^{r_{1k} t_n} \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Re(x(n, m)) - Re(g(n, m))) t_n s_m c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \sin(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k} t_n + r_{2l} s_m} \sum_{u=1}^K c_{ul} e^{r_{1u} t_n} \sin(\omega_{1u} t_n + \omega_{2l} s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \cos(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}) \\
&\quad + \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} (Im(x(n, m)) - Im(g(n, m))) t_n s_m c_{kl} e^{r_{1k} t_n + r_{2l} s_m} \cos(\omega_{1k} t_n + \omega_{2l} s_m + \phi_{kl}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k} t_n + r_{2l} s_m} \sum_{u=1}^K c_{ul} e^{r_{1u} t_n} \cos(\omega_{1u} t_n + \omega_{2l} s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v} s_m} \sin(\omega_{1k} t_n + \omega_{2v} s_m + \phi_{kv}).
\end{aligned}$$

Other second order partial derivatives can be similarly derived, but are omitted here to save space.

6.7.2 A.2. List of Entries of Fisher Information Matrix

Using the notation of Section 6.2, and taking the expectation of the second order partial derivatives derived in the previous section of the Appendix, we here list the entries of the Fisher information matrix for the data analysis of 2D NMR data sets are discussed in Section 6.2.

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial c_{pq}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2q})s_m} \cdot \cos((\omega_{1k} - \omega_{1p})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{kl} - \phi_{pq})), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \cos(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \sin(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{2l} \partial r_{2q}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \left(\sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \right) \\
&\quad \cdot \left(\sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \left(\sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \right) \\
&\quad \cdot \left(\sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}) \right), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{1k} \partial \omega_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \sin(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \left(\sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \right) \\
&\quad \cdot \left(\sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \cos(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \right), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{2l} \partial \omega_{2q}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \left(\sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \right) \\
&\quad \cdot \left(\sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}) \right) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \left(\sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \right) \\
&\quad \cdot \left(\sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}) \right), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \phi_{kl} \partial \phi_{pq}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{kl} c_{pq} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2q})s_m} \cdot \cos((\omega_{1k} - \omega_{1p})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{kl} - \phi_{pq})).
\end{aligned}$$

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial r_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{v=1}^L \frac{1}{\sigma_m^2} t_n c_{pv} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2v})s_m} \cdot \cos((\omega_{1k} - \omega_{1p})t_n + (\omega_{2l} - \omega_{2v})s_m \\
&\quad + (\phi_{kl} - \phi_{pv})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial r_{2q}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{u=1}^K \frac{1}{\sigma_m^2} s_m c_{uq} e^{(r_{1k}+r_{1u})t_n + (r_{2l}+r_{2q})s_m} \cdot \cos((\omega_{1k} - \omega_{1u})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{kl} - \phi_{uq})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial \omega_{1p}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{v=1}^L \frac{1}{\sigma_m^2} t_n c_{pv} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2v})s_m} \cdot \sin((\omega_{1k} - \omega_{1p})t_n + (\omega_{2l} - \omega_{2v})s_m \\
&\quad + (\phi_{kl} - \phi_{pv})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial \omega_{2q}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{u=1}^K \frac{1}{\sigma_m^2} s_m c_{uq} e^{(r_{1k}+r_{1u})t_n + (r_{2l}+r_{2q})s_m} \cdot \sin((\omega_{1k} - \omega_{1u})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{kl} - \phi_{uq})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial c_{kl} \partial \phi_{pq}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} c_{pq} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2q})s_m} \cdot \sin((\omega_{1k} - \omega_{1p})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{kl} - \phi_{pq})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial r_{2l}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial \omega_{1p}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \sin(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n^2 e^{(r_{1k}+r_{1p})t_n} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{v=1}^L c_{pv} e^{r_{2v}s_m} \cos(\omega_{1p}t_n + \omega_{2v}s_m + \phi_{pv}). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial \omega_{2l}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}).
\end{aligned}$$

$$\begin{aligned}
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{1k} \partial \phi_{pl}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{v=1}^L \frac{1}{\sigma_m^2} t_n c_{kv} c_{pl} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2v})s_m} \cdot \sin((\omega_{1k} - \omega_{1p})t_n + (\omega_{2v} - \omega_{2l})s_m \\
&\quad + (\phi_{kv} - \phi_{pl})). \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{2l} \partial \omega_{1k}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{2l} \partial \omega_{2q}} &= \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} s_m^2 e^{(r_{2l}+r_{2q})s_m} \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad \cdot \sum_{u=1}^K c_{uq} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2q}s_m + \phi_{uq}), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial r_{2l} \partial \phi_{kq}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{u=1}^K \frac{1}{\sigma_m^2} s_m c_{kq} c_{ul} e^{(r_{1k}+r_{1u})t_n + (r_{2l}+r_{2q})s_m} \cdot \sin((\omega_{1u} - \omega_{1k})t_n + (\omega_{2l} - \omega_{2q})s_m \\
&\quad + (\phi_{ul} - \phi_{kq})), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{1k} \partial \omega_{2l}} &= - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \sin(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \sin(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}) \\
&\quad - \sum_{n=1}^N \sum_{m=1}^M \frac{1}{\sigma_m^2} t_n s_m e^{r_{1k}t_n + r_{2l}s_m} \sum_{v=1}^L c_{kv} e^{r_{2v}s_m} \cos(\omega_{1k}t_n + \omega_{2v}s_m + \phi_{kv}) \\
&\quad \cdot \sum_{u=1}^K c_{ul} e^{r_{1u}t_n} \cos(\omega_{1u}t_n + \omega_{2l}s_m + \phi_{ul}), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{1k} \partial \phi_{pl}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{v=1}^L \frac{1}{\sigma_m^2} t_n c_{kv} c_{pl} e^{(r_{1k}+r_{1p})t_n + (r_{2l}+r_{2v})s_m} \cdot \cos((\omega_{1k} - \omega_{1p})t_n + (\omega_{2v} - \omega_{2l})s_m \\
&\quad + (\phi_{kv} - \phi_{pl})), \\
E \frac{\partial \ln^2(p(x, \Theta))}{\partial \omega_{2l} \partial \phi_{kq}} &= - \sum_{n=1}^N \sum_{m=1}^M \sum_{u=1}^K \frac{1}{\sigma_m^2} s_m c_{kq} c_{ul} e^{(r_{1k}+r_{1u})t_n + (r_{2l}+r_{2q})s_m} \cdot \cos((\omega_{1k} - \omega_{1u})t_n + (\omega_{2q} - \omega_{2l})s_m \\
&\quad + (\phi_{kq} - \phi_{ul})).
\end{aligned}$$

Chapter 7

Biomolecular Structure

Determination from NMR Data

Biomolecular structures provide important information for understanding biological functions and are the basis for many research areas such as structure-based drug discovery and design. Due to the complex nature of molecules, computing the three-dimensional molecular structures is very difficult and there is an obvious need for effective and accurate structure determination methods. In this chapter, the potential energy function is further developed and the detailed transformation formulae from Cartesian coordinates to internal coordinates are presented. The ripple search strategy is applied to the molecular structure determination with NMR constraints. The practical applicability of the developed potential energy function and the good performance of the ripple search strategy are demonstrated through structural investigations of several molecules.

7.1 Procedure of Structure Determination from NMR Data

In the recent 20 years, the use of NMR data has become a widely developed technique for determining biomolecular structures, such as protein, nucleic acids and polysaccharides as well as small molecules. The procedure can be summarized as three steps: structure infor-

mation presentation, energy function minimization with NMR constraints and structure visualization, as shown in Figure 7.1.

Figure 7.1: Procedure of structure determination from NMR data

Step one: Structure information presentation

The 3D Cartesian position of each atom a_m , $m = 1, \dots, n$, is represented by $(x_m, y_m, z_m) \in R^3$, where x_m, y_m, z_m are unknown variables, n is the number of atoms in the biomolecule. With the knowledge of the primary structure obtained from NMR sequential resonance assignments, biomolecular structure information can be transformed into internal coordinates which consist of bond lengths L , bond angles θ , and the dihedral angles ϕ . The transformations will be introduced and defined in Section 7.2.

The dihedral angle ϕ , also known as the torsion angle, is a very useful descriptive tool as it can describe all combinations of tetrahedral, trigonal and diagonal atoms. The dihedral angle is the angle between two planes defined as $A-X-Y$ and $X-Y-B$ when considering four atoms connected in the order $A-X-Y-B$, as shown in Figure 7.2. It can also be considered as the relative angle between the $A-X$ bonds and the $B-Y$ bonds. Rotations around a $X-Y$ single bond will therefore lead to different dihedral angles. By convention, a positive value of the dihedral angle $A-X-Y-B$ is assigned to the clockwise rotation of

up to 180° necessary to bring the front atom into an eclipsed position with the rear atom. For example, as shown in Figure 7.3, in the first case (left) which is positive, it would be necessary to rotate $A-X$ clockwise so that the front bond $A-X$ eclipsed the back bond $Y-B$. In the second case (right) it is necessary to rotate the front bond $A-X$ anti-clockwise less than 180° to eclipse the back bond $Y-B$, therefore the dihedral angle is negative.

Figure 7.2: Four atoms connected in the order $A-X-Y-B$

Figure 7.3: Definition of the positive or negative value of a dihedral angle

Step two: Energy function minimization with NMR constraints

An important class of difficult global minimization problems arises in molecular structure calculations with NMR constraints. The data obtained from NMR studies consist of distance and angle constraints. Once resonances have been assigned, NOE contacts are selected, and their intensities are used to calculate interproton distances. Information on torsional angles are based on the measurement of coupling constants and analysis of proton chemical shifts. The objective function to be minimized is an energy called potential energy function. Together, this information can be used to formulate a nonlinear optimization problem, whose solution should provide the correct biomolecular structure. In these calculations, the determination of the correct biomolecular structure, or native state, can be formulated by calculating the global (or approximate global) minimum of the potential energy function. The energy landscape of a biomolecule possesses an enormous

number of minima, or conformational substates, that may grow exponentially with the size of the biomolecule. Note that the energy at local minima may be much higher than the energy of the global minimum, and hence it is very important to find the global minimum.

The approach above is generally based on two assumptions. First, there exists a potential energy function for the biomolecule; Second, the folded (native) state corresponds to the structure with the lowest potential energy (global minimum of the potential energy function) and is thus in a state of thermodynamic equilibrium. This view is supported by in vitro observations that biomolecules can successfully refold from a variety of denatured states. Evolutionary theory also supports a folded state at a global energy minimum. Protein sequences have evolved under pressure to perform certain functions, which for most known occurrences requires a stable, unique, and compact structure. Unless specifically required for a certain function, there was no biochemical need for proteins to hide their global minimum behind a large kinetic energy barrier.

Among the most commonly used potential energy functions are the AMBER, CHARMM, GROMOS and OPLS/AMBER force fields [12]. All of them have similar presentations. The continuing development of force fields remains an intense area of research with implications for both fundamental research as well as for applied research in the pharmaceutical industry. The potential energy function used in our research will be introduced and developed in Section 7.2.

Calculating 3D structures using NMR data is, therefore, dependent on the development of efficient optimization methods. The development of parameter sets is a very laborious task, requiring extensive optimization. This is an area of continuing research and many groups have been working over the past two decades to derive functional forms and parameters for potential energy functions of general applicability to biological molecules. Currently, there are several widely available methods for solving these problems, which, in general, rely on the optimization of penalty-type target functions to satisfy the conformational constraints. Typically, these methods combine simulated annealing protocols with molecular dynamics and local minimization, either in distance or torsional angle space, for example, the algorithms used in software packages like X-PLOR, DYANA [11][12].

The disadvantage is that it requires long computational time. An overview of available methods for predicting 3D biomolecular structures can be found elsewhere [112].

The similarity of two NMR structures can be measured by the root-mean-square deviation (RMSD) [113]:

$$RMSD = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_i - r_i')^2} \quad (7.1)$$

which can be obtained by fitting a structure to the other by translation and rotation, where n is the number of atoms being compared, r_i and r_i' are the atomic coordinate sets of the computed and the known structures respectively.

Step three: Structure visualization

The global solution of the potential energy function minimization presents the locations of atoms in Cartesian Coordinate Space, and can be visualized by many softwares, such as RasMol [114][115].

RasMol is a molecular graphics program intended for the visualization of proteins, nucleic acids and small molecules. RasMol reads in molecular coordinate files in a number of formats including Brookhaven Protein Databank (PDB) format as shown in Figure 7.4 and interactively displays the molecule on the screen in a variety of color schemes and representations. The displayed molecule may be rotated, translated and zoomed using either the mouse, the scroll bars or the command line. A menu allows various types of structure, such as spacefilling, backbone, and coloring schemes. More complex coloring and highlighting can be accomplished through the Command Line window. Finally the rendered image may be written out in a variety of formats such as PostScript, GIF, BMP, etc.

7.2 Definition of Potential Energy Function

The basic potential energy function used in our study is a traditional one which has been introduced in [116]. It takes the following form:

```

COMPND      ETHANOL
AUTHOR      DAVE WOODCOCK  96 01 03
ATOM        1  C          1      -0.426  -0.115  -0.147  1.00  0.00
ATOM        2  O          1      -0.599   1.244  -0.481  1.00  0.00
ATOM        3  H          1      -0.750  -0.738  -0.981  1.00  0.00
ATOM        4  H          1      -1.022  -0.351   0.735  1.00  0.00
ATOM        5  H          1      -1.642   1.434  -0.689  1.00  0.00
ATOM        6  C          1       1.047  -0.383   0.147  1.00  0.00
ATOM        7  H          1       1.370   0.240   0.981  1.00  0.00
ATOM        8  H          1       1.642  -0.147  -0.735  1.00  0.00
ATOM        9  H          1       1.180  -1.434   0.405  1.00  0.00
TER         10
END

```

Figure 7.4: The PDB file of a simple compound ethanol including atom coordinates.

Figure 7.5: Display in RasMol and the command line window.

$$E = E_{bonds} + E_{angles} + E_{improper} + E_{distance} + E_{torsion} + E_{forcefield} \quad (7.2)$$

The first three terms (E_{bonds} , E_{angles} , $E_{improper}$) in the above equation are generally expressed as quadratic harmonic potentials for bond lengths, bond angles, and chirality or planarity. For example:

$$E_{bonds} = \sum_{bonds} k_L (L - L_0)^2 \quad (7.3)$$

$$E_{angles} = \sum_{angles} k_{\theta}(\theta - \theta_0)^2 \quad (7.4)$$

$$E_{improper} = \sum_{improvers} k_{\phi}(\phi - \phi_0)^2 \quad (7.5)$$

where L_0 , θ_0 , ϕ_0 represent reference values for the bond lengths L , bond angles θ and improper angles ϕ respectively, k_L , k_{θ} , and k_{ϕ} are the corresponding force constants, which determine the strength of the forces. Both reference values and force constants are specific for each pair of bond atoms, depending on chemical type of atoms-constituents. Reference values can be inferred from high resolution crystal structures or microwave spectroscopy data. Values of force constants are often evaluated from experimental data such as infrared stretching frequencies or from quantum mechanical calculations. These three terms describe the deviation from an ideal geometry. Effectively, they are penalty functions and in a perfectly optimized structure, the sum of them should be close to zero. Note that the definition of dihedral and improper angles is identical. Historically, improper angles are mostly used with the third term to maintain planarity or chirality, whereas dihedral angles are used with the sixth term to describe multi-minimum torsion potentials.

The fourth term $E_{distance}$ of Equation (7.2) represents the objective function contribution corresponding to NMR experimental distance constraints. The most typical form uses a simple square well potential including a summation over both upper and lower distance violations, for example, $E_{distance} = E_{distance}^{upper} + E_{distance}^{lower}$. For upper distance constraints, there is:

$$E_{distance}^{upper} = \sum_{upper} \begin{cases} A_{ij}(r_{ij} - r_{ij}^{upper})^2 & (\text{if } r_{ij} > r_{ij}^{upper}) \\ 0 & (\text{otherwise}) \end{cases} \quad (7.6)$$

The squared violation energy is considered only when the calculated distance r_{ij} between any atom pair (i, j) exceeds the upper reference distance r_{ij}^{upper} and is multiplied by a weighting factor A_{ij} . It is similar to calculate lower distance violations:

$$E_{distance}^{lower} = \sum_{lower} \begin{cases} B_{ij}(r_{ij} - r_{ij}^{lower})^2 & (\text{if } r_{ij} < r_{ij}^{lower}) \\ 0 & (\text{otherwise}) \end{cases} \quad (7.7)$$

where r_{ij}^{lower} is the lower reference distance and B_{ij} is the weighting factor.

When considering constraints of torsion angle φ represented by term $E_{torsion}$ of Equation (7.2), a form similar to Equations (7.6) and (7.7) is often used. The total violation, $E_{torsion}$, is a sum over upper and lower violations (i.e., $E_{torsion} = E_{torsion}^{upper} + E_{torsion}^{lower}$). A torsion angle φ_j can be constrained by employing a quadratic square well potential using upper (φ_j^{upper}) and lower (φ_j^{lower}) bounds on the variable values. This results in the following equations:

$$E_{torsion}^{upper} = \sum_{upper} \begin{cases} A_j \left(1 - 2 \left[\frac{\varphi_j - \varphi_j^{upper}}{2\pi - (\varphi_j^{upper} - \varphi_j^{lower})} \right]^2 \right) (\varphi_j - \varphi_j^{upper})^2 & (\text{if } \varphi_j > \varphi_j^{upper}) \\ 0 & (\text{otherwise}) \end{cases} \quad (7.8)$$

$$E_{torsion}^{lower} = \sum_{lower} \begin{cases} B_j \left(1 - 2 \left[\frac{\varphi_j - \varphi_j^{lower}}{2\pi - (\varphi_j^{upper} - \varphi_j^{lower})} \right]^2 \right) (\varphi_j - \varphi_j^{lower})^2 & (\text{if } \varphi_j < \varphi_j^{lower}) \\ 0 & (\text{otherwise}) \end{cases} \quad (7.9)$$

where A_j and B_j are weighting factors.

Finally, the last term $E_{forcefield}$ in Equation (7.2) accounts for the force field energy expressions used to model the nonbonded interactions of the biomolecule. These often correspond to simple Van der Waals repulsion terms. More detailed force fields employ 6-12 Lennard-Jones and modified 10-12 Lennard-Jones terms to model nonbonded and hydrogen bonded interactions, respectively. An additional Coulombic electrostatic term may also be included. The total force field energy, $E_{forcefield}$, is calculated as the sum of the electrostatic, nonbonded, hydrogen bonded, and torsion contributions.

The general potential energy terms of $E_{forcefield}$ are shown in Equation (7.10).

$$E_{forcefield} = \sum_{torsions} \frac{V_n}{2} (1 + \cos(n\varphi_j + \delta)) + \sum_{i,j} \epsilon_{ij} \left(\left(\frac{r_{ij}^0}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{ij}^0}{r_{ij}} \right)^6 \right) + \sum_{i,j} \frac{q_i q_j}{r_{ij}} \quad (7.10)$$

The first term in $E_{forcefield}$ represents the torsion angle potential function which models the presence of steric barriers between atoms separated by three covalent bonds. V_n is the parameter corresponding to torsional barrier energies for a given torsion angle, n refers to the symmetry type for the particular torsion angle and δ is the phase shift. This term provides for preferred torsion angles within the molecular structure. For example, if $n = 3$ and the phase shift δ is zero, preferred torsion angles (configurations resulting in a small penalty) can be found at 60° , 180° , and 300° .

The second term in $E_{forcefield}$ represents the Van der Waals interaction energy, where r_{ij}^0 is the equilibrium separation distance and ϵ_{ij} is the well depth. The van der Waals interaction between two atoms arises from a balance between repulsive and attractive forces. The repulsive force arises at short distances where the electron-electron interaction is strong. The attractive force, also referred to as the dispersion force, arises from fluctuations in the charge distribution in the electron clouds. The fluctuation in the electron distribution on one atom or molecule gives rise to an instantaneous dipole which, in turn, induces a dipole in the second atom or molecule giving rise to an attractive interaction. Each of these two effects is equal to zero at infinite atomic separation r and become significant as the distance decreases. The attractive interaction is longer range than the repulsion but as the distance becomes shorter, the repulsive interaction becomes dominant. This gives rise to a minimum in the energy. Positioning of the atoms at the optimal distances stabilizes the system. The van der Waals interaction is most often modeled using the Lennard-Jones 6-12 potential which expresses the interaction energy.

The third term in $E_{forcefield}$ represents the electrostatic interaction energy with q_i and q_j as the atoms' partial charges. Generally, atoms separated by three bonds may have electrostatic interactions reduced by a multiplicative scale factor.

7.3 Development of Potential Energy Function

7.3.1 Potential Energy Function Development

The forms in Equations (7.6) - (7.9) as a part of potential energy function are not practical in global optimization because they are equations of condition. Therefore in this thesis,

we define

$$E_{distance} = \sum_{constraints} \frac{A_{ij}}{2} \left(\tanh c \left(r_{ij} - r_{ij}^{upper} \right) - \tanh c \left(r_{ij} - r_{ij}^{lower} \right) + 2 \right) \left(r_{ij} - \frac{r_{ij}^{upper} + r_{ij}^{lower}}{2} \right)^2 \quad (7.11)$$

where c is a large constant (i.e., $c = 300$) and A_{ij} is weighting factor.

When considering constraints of torsion angle φ , similarly we have

$$E_{torsion} = \sum_{constraints} \frac{B_{ij}}{2} \left(\tanh c \left(\varphi_j - \varphi_j^{upper} \right) - \tanh c \left(\varphi_j - \varphi_j^{lower} \right) + 2 \right) \left(\varphi_j - \frac{\varphi_j^{upper} + \varphi_j^{lower}}{2} \right)^2 \quad (7.12)$$

with B_{ij} as the weighting factor.

In the above definitions, penalty terms of both upper and lower constraints are included in one equation, rather than two as in [116]. Such definitions are more convenient and practical for the optimization problem. The sketch of Equations (7.11) and (7.12) is shown in Figure 7.6.

Figure 7.6: The sketch of $y = \frac{1}{2} (\tanh 300 (x - b) - \tanh 300 (x - a) + 2)$.

7.3.2 Transformation from Cartesian Coordinates to Internal Coordinates

For clarity, the potential energy function is represented by bond lengths L , bond angles θ , torsion angles φ and atom distances r_{ij} in internal coordinates. Note that all methods calculate in the end the Cartesian coordinates, rather than internal coordinates [112]. Therefore it is necessary to substitute internal coordinates by Cartesian coordinates. However, the crucial step of transformation is usually neglected in the literature.

In this thesis, we give out the detailed transformation formulae from Cartesian coordinates of atoms to internal coordinates. These formulae will be used with our developed potential energy function for the structure determination problems.

Calculate atom distances:

Suppose that the three dimensional Cartesian position of atom a_m , $m = 1, \dots, n$, is represented by $(x_m, y_m, z_m) \in R^3$. The distance r_{ij} (including L_{ij}) between any atom pair (i, j) is calculated by

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \quad (7.13)$$

where $i = 1, \dots, n-1$ and $j = i+1, \dots, n$.

Calculate bond angles:

The bond angle $\theta_{i-2, i-1, i}$ of bonds $(i-2, i-1)$ and $(i-1, i)$ is calculated by

$$\theta_{i-2, i-1, i} = \arccos \frac{P_i P_{i-1} + Q_i Q_{i-1} + R_i R_{i-1}}{\sqrt{P_i^2 + Q_i^2 + R_i^2} \sqrt{P_{i-1}^2 + Q_{i-1}^2 + R_{i-1}^2}} \quad (7.14)$$

where,

$$\begin{cases} P_i = x_i - x_{i-1} \\ Q_i = y_i - y_{i-1} \\ R_i = z_i - z_{i-1} \end{cases} \quad (7.15)$$

for $i = 3, \dots, n$.

Calculate torsion angles:

Suppose that there are three atoms a_i , a_{i-1} and a_{i-2} , presented by (x_i, y_i, z_i) , $(x_{i-1}, y_{i-1}, z_{i-1})$ and $(x_{i-2}, y_{i-2}, z_{i-2})$ respectively, on a plane, where $i = 4, \dots, n$. The function of the plane can be expressed by:

$$\begin{vmatrix} x - x_{i-2} & y - y_{i-2} & z - z_{i-2} \\ x_{i-1} - x_{i-2} & y_{i-1} - y_{i-2} & z_{i-1} - z_{i-2} \\ x_i - x_{i-2} & y_i - y_{i-2} & z_i - z_{i-2} \end{vmatrix} = 0 \quad (7.16)$$

As the general form of the function for the plane is:

$$A_1x + B_1y + C_1z = D_1 \quad (7.17)$$

we have

$$A_1 = y_{i-2}(z_{i-1} - z_i) + y_{i-1}(z_i - z_{i-2}) + y_i(z_{i-2} - z_{i-1});$$

$$B_1 = z_{i-2}(x_{i-1} - x_i) + z_{i-1}(x_i - x_{i-2}) + z_i(x_{i-2} - x_{i-1});$$

$$C_1 = x_{i-2}(y_{i-1} - y_i) + x_{i-1}(y_i - y_{i-2}) + x_i(y_{i-2} - y_{i-1});$$

$$D_1 = x_{i-2}(y_{i-1}z_i - z_{i-1}y_i) + y_{i-2}(z_{i-1}x_i - z_ix_{i-1}) + z_{i-2}(x_{i-1}y_i - x_iy_{i-1}).$$

Similarly, the general form of the plane composed by atoms a_{i-1} , a_{i-2} and a_{i-3} is:

$$A_2x + B_2y + C_2z = D_2 \quad (7.18)$$

with

$$A_2 = y_{i-3}(z_{i-2} - z_{i-1}) + y_{i-2}(z_{i-1} - z_{i-3}) + y_{i-1}(z_{i-3} - z_{i-2});$$

$$B_2 = z_{i-3}(x_{i-2} - x_{i-1}) + z_{i-2}(x_{i-1} - x_{i-3}) + z_{i-1}(x_{i-3} - x_{i-2});$$

$$C_2 = x_{i-3}(y_{i-2} - y_{i-1}) + x_{i-2}(y_{i-1} - y_{i-3}) + x_{i-1}(y_{i-3} - y_{i-2});$$

$$D_2 = x_{i-3}(y_{i-2}z_{i-1} - z_{i-2}y_{i-1}) + y_{i-3}(z_{i-2}x_{i-1} - z_{i-1}x_{i-2}) + z_{i-3}(x_{i-2}y_{i-1} - x_{i-1}y_{i-2}).$$

Thus the torsion angle of atoms a_i , a_{i-1} , a_{i-2} and a_{i-3} is calculated by

$$\varphi_{i,i-1,i-2,i-3} = \arccos \frac{A_1A_2 + B_1B_2 + C_1C_2}{\sqrt{A_1^2 + B_1^2 + C_1^2} \sqrt{A_2^2 + B_2^2 + C_2^2}} \quad (7.19)$$

Illustrations of bond lengths, bond angles, atom distances and torsion angles are shown in Figure 7.7. Substituting Equations (7.13), (7.14) and (7.19) into (7.2), the actual variables of the potential energy function is the atomic Cartesian coordinates (x_m, y_m, z_m) , rather than variables in internal coordinates. The potential energy function is differentiable with respect to the atomic coordinates and is highly nonlinear and very complicated for minimization.

Figure 7.7: Illustrations of the bond length L_{12} , the bond angle θ_{234} , the atom distance r_{13} and the torsion angle φ_{5432} .

7.4 Applications to Molecular Structure Determination

Molecular structure determination can be formulated as an unconstrained nonlinear optimization problem of minimizing the potential energy function (7.2), that is:

$$\min_{\mathbf{X} \in R^{3n}} E(\mathbf{X}) \quad (7.20)$$

where, n is the number of molecular atoms, $\mathbf{X} = [x_1, y_2, z_3, \dots, x_n, y_n, z_n]$ is a vector of atomic Cartesian coordinates presenting molecular structures. The search region $D \in R^{3n}$ is the Cartesian coordinate space.

The global minimum of the potential energy function should provide the correct molecular structure and satisfy the NMR experimental data. The search space is the Cartesian

coordinate space of the atoms in a molecule. Due to the complex nature of molecules, computing the global minimum of the energy function is very difficult as there are typically a very large number of local minima which may grow exponentially with the size of the molecule. Therefore there is an obvious need for effective and accurate structure determination methods. Global optimization technique is a good choice and it also has many other applications in computational biology, such as similar ancestral sequence reconstructions and RNA discovery [117]-[119]. Currently some methods and software packages based on global optimization have been applied to molecular structure determination. However they are in general time-consuming and most of them obtain molecular structures by using drastically simplified potential energy functions to reduce the computational complexity [120]. Such simplifications include assuming that the bond lengths and bond angles are fixed at their equilibrium values, therefore to address the potential energy function with just the torsion angles as the variables [116][121]-[125]. In practice it is common that the terms E_{bonds} , E_{angles} and $E_{improper}$ in the potential energy function (7.2) are neglected [116][121]-[123]. Some methods model the molecular structures by the positions and restrictions between the non-Hydrogen atoms or atoms on the backbone only [121]. With these simplifications, the number of variables and the landscape complexity of the potential energy function are significantly reduced. The drawback is the low structural quality [121] and thus generally multiple runs of structural computation are required. Some methods use additional process to generate a sufficiently accurate starting structure for quickly producing an adequate solution by minimization [11][12][121].

In this section, ripple search strategy is applied to determine the 3D structures of two organic molecules, called 1-Propanol and 2-Methyl-1-Propanol, and one deoxyribonucleic acid with protein data bank entry 3REC [126] using NMR constraints. The suggested ranges of the control parameters for general molecular structure determination problems are $m_0 \subseteq [10, 30]$, $p_a \subseteq [1, 3]$ and $\Delta r \subseteq [0.1, 0.3]$. In Equation (7.2), the force constants and weighting factors are assumed to be identical for each atom pair (i, j) [127]. Similar to [127], we set $k_L = 310.0 \text{ kcal}/\text{Angstrom}^2$, $k_\theta = 40.0 \text{ kcal}/\text{rad}^2$, $k_\phi = 50.0 \text{ kcal}/\text{rad}^2$, $c = 300$, $A_{ij} = B_{ij} = 100 \text{ kcal}/\text{Angstrom}^2$, $V_n = 1.3 \text{ kcal}$ and $\epsilon_{ij} = 0.181 \text{ kcal}$. The reference values L_0 , θ_0 and ϕ_0 and equilibrium distance r_{ij}^0 are calculated from the standard

structure in the literature by Equations (7.13), (7.14) and (7.19), describing the ideal geometry. All the experimental results are generated under MATLAB 6.5 environment [84] on a PC with 2.4GHz clock speed and are carried out without fine-tuning the control parameters of the ripple search strategy. In all experiments, SQP method [84] is adopted as the traditional local search method. It is worth noting that different traditional methods with different settings may lead to different number of function evaluations and CPU time. All structures are visualized by software RasMol [114][115].

7.4.1 Organic Molecule 1-Propanol

The chemical formula of the molecule 1-Propanol is $CH_3(CH_2)_2OH$. The molecule has 12 atoms thus the potential energy function has 36 variables to be optimized. Its standard 3D structure is shown in Figure 7.8.

Figure 7.8: Ball and stick display in RasMol for the molecule 1-Propanol.

NOE cross peaks were classified as strong, medium and weak by visual inspection of the contour levels. The following distance limits were assigned to these NOE intensities:

- strong: (1.8 to 2.5 Angstrom)
- medium: (2.5 to 3.5 Angstrom)
- weak: (3.5 to 4.5 Angstrom)

With the knowledge of the primary structure obtained from NMR sequential resonance assignments, as shown in Figure 7.9, atomic Cartesian coordinates can be transformed into internal coordinates by Equations (7.13), (7.14) and (7.19). They are then substituted into Equation (7.2) to form the objective function.

The numbers of distance and torsion angle constraints are 27 and 10 respectively. The control parameters for ripple search strategy are $m_0 = 20$, $p_a = 2$ and $\Delta r = 0.25$. The starting point is generated randomly in the interval $[-1, 1]$. Because the unknown variables of the potential energy function are the atomic Cartesian coordinates of the molecule, the random starting point presents a starting structure generated randomly, or random starting structure in short. Figure 7.10 shows the standard structure in the literature in display of sticks, while Figure 7.11 presents the structure found by the ripple search strategy. The number of function evaluations is 753694. The CPU time taken is 903.353 seconds.

Figure 7.9: Primary structure of the molecule 1-Propanol.

Figure 7.10: Standard structure of the 1-Propanol in display of sticks.

Note that the global minimum of the potential energy function provides the relative atomic coordinates of the molecule instead of the absolute Cartesian coordinates. In order to compare the computed and the standard structures, according to the definition of the RMSD in Equation (7.1), structure translation and rotation are required. The calculation formulae are given below. More detailed information is described in [128][129].

Assume that the structure is defined by the Cartesian coordinates of the atoms and represented by an $n \times 3$ coordinate matrix, where n is the number of atoms in the molecule.

Figure 7.11: Structure of the 1-Propanol found by ripple search strategy in display of sticks.

Let $\mathbf{X}_m$ and $\mathbf{Y}_m$ be the coordinate matrices of the computed and the known structures to be compared, respectively.

Structure translation can be done by adding a translation vector to each row of the coordinate matrix so that after translation the geometric centers of the computed and the known structures are at the same location. First, we define:

$$\mathbf{x}_c(j) = \frac{1}{n} \sum_{i=1}^n \mathbf{X}_m(i, j), \quad j = 1, 2, 3$$

and

$$\mathbf{y}_c(j) = \frac{1}{n} \sum_{i=1}^n \mathbf{Y}_m(i, j), \quad j = 1, 2, 3$$

where the column vector $\mathbf{x}_c = (\mathbf{x}_c(1), \mathbf{x}_c(2), \mathbf{x}_c(3))^T$ is the mean vector of the matrix $\mathbf{X}_m$ and similarly for $\mathbf{y}_c$. Then, the difference matrix $\mathbf{X}_D$ between $\mathbf{X}_m$ and its mean vector $\mathbf{x}_c$ is:

$$\mathbf{X}_D = \mathbf{X}_m - e_n \mathbf{x}_c^T,$$

where the column vector $e_n = (1, \dots, 1)^T$ in R^n . Similarly,

$$\mathbf{Y}_D = \mathbf{Y}_m - e_n \mathbf{y}_c^T.$$

Therefore the geometric centers of $\mathbf{X}_D$ and $\mathbf{Y}_D$ are coincide.

Structure rotation can be done via multiplying the coordinate matrix by a rotation matrix.

Let $\mathbf{C} = \mathbf{Y}_D^T \mathbf{X}_D$ and compute its singular-value decomposition $\mathbf{C} = \mathbf{U} \Sigma \mathbf{V}^T$. It can be verified that the matrix $\mathbf{Q} = \mathbf{U} \mathbf{V}^T$ is the rotation matrix required in the calculation of RMSD and $\mathbf{Q} \mathbf{Q}^T = \mathbf{I}$ [128][129].

The RMSD of the two structures is then calculated by:

$$RMSD = ||\mathbf{X}_D - \mathbf{Y}_D \mathbf{Q}||_F / \sqrt{n}, \quad (7.21)$$

where $||\cdot||_F$ is the matrix Frobenius norm [128].

Therefore the RMSD value between the structure determined by ripple search strategy and the standard structure is

$$RMSD = ||\mathbf{X}_D - \mathbf{Y}_D \mathbf{Q}||_F / \sqrt{12} = 0.0036(\text{Angstrom}). \quad (7.22)$$

Note that structures with the RMSD value of 0 Angstrom are considered identical and those with the RMSD value smaller than 2 Angstrom are normally considered highly similar [130].

Unlike many other methods for structure determination using additional process to generate a sufficiently accurate starting structure for minimization [11][12][121], ripple search strategy begins with a random starting structure. It uses no simplifications of the potential energy function (7.2) or assumptions on molecular structures, as mentioned previously. Therefore it is hard to find proper comparisons in the literature. However by tests the basic idea of ripple search strategy has shown superiority over many global optimization methods including simulated annealing [60], which is adopted in software packages like X-PLOR and DYANA [11][12].

7.4.2 Organic Molecule 2-Methyl-1-Propanol

The chemical formula of the molecule 2-Methyl-1-Propanol is $(CH_3)_2CHCH_2OH$. The molecule has 15 atoms thus the potential energy function has 45 variables. Its standard 3D structure is shown in Figure 7.12.

With the knowledge of the primary structure shown in Figure 7.13, atomic Cartesian coordinates can be transformed into internal coordinates by Equations (7.13), (7.14) and (7.19). They are then substituted into Equation (7.2).

The numbers of distance and torsion angle constraints used are 44 and 8 respectively. The control parameters for ripple search strategy are $m_0 = 20$, $p_a = 2$ and $\Delta r = 0.25$. The starting structure is generated randomly in the interval $[0, 50]$. Figures 7.14 and 7.15

Figure 7.12: Ball and stick display in Rasmol for the molecule 2-Methyl-1-Propanol.

Figure 7.13: Primary structure of the molecule 2-Methyl-1-Propanol.

Figure 7.14: Standard structure of the 2-Methyl-1-Propanol in display of sticks.

show the standard structure and the one found by the ripple search strategy respectively. According to Equation (7.21), the RMSD value between the structure determined by ripple search strategy and the standard structure is

$$RMSD = \|\mathbf{X}_D - \mathbf{Y}_D \mathbf{Q}\|_F / \sqrt{15} = 0.0030(\text{Angstrom}). \quad (7.23)$$

The structure found by our method is nearly the same as the standard one in the literature. The number of function evaluations is 1056102. The CPU time taken is 2045 seconds.

Figure 7.15: Structure of the 2-Methyl-1-Propanol found by ripple search strategy in display of sticks.

7.4.3 Deoxyribonucleic Acid 3REC

The third biomolecule under investigation is 3REC, a deoxyribonucleic acid, deposited in the protein data bank [126][131] in 1997. Its structure well explains the axial extension of DNA in the RecA-filaments relative to B-form DNA and leads to a possible interpretation of the role of this extension in homologous pairing [131]. The biomolecule has 63 atoms thus the potential energy function has 189 variables. Its standard 3D structure is shown in Figure 7.16.

Figure 7.16: Ball and stick display in Rasmol for the biomolecule 3REC.

The numbers of distance and torsion angle constraints are 99 and 14 respectively. The control parameters for ripple search strategy are $m_0 = 10$, $p_a = 2$ and $\Delta r = 0.1$. The

starting structure is generated randomly in the interval $[-20, 0]$.

Figure 7.17 shows the standard structure in the protein data bank, while Figure 7.18 presents the structure found by the ripple search strategy.

Figure 7.17: Standard structure of the 3REC in display of sticks.

Figure 7.18: Structure of the 3REC found by ripple search strategy in display of sticks.

The RMSD value between the structure determined by ripple search strategy and the

standard structure is

$$RMSD = ||\mathbf{X}_D - \mathbf{Y}_D \mathbf{Q}||_F / \sqrt{63} = 0.2727(\text{Angstrom}). \quad (7.24)$$

The number of function evaluations is 29200094. The CPU time taken is $4.8845E5$ seconds. Note that the CPU time is basically spent on the local minimization by the traditional method and the computational costs of generating trial points by ripple type search can be ignored, especially for high-speed computers.

The experiment shows that ripple search strategy can be successfully applied to the 3REC structure determination with total 189 variables in the potential energy function. Whereas in [116][121][123][124], the proteins under investigations have 75 atoms with 24 variables, 13 residues with 73 variables, 72 residues with 104 variables (up to 140 variables) and 58 residues with 116 variables in the potential energy functions respectively. All of them use simplified potential energy functions to reduce the number of variables and the complexity of the function landscape. Therefore ripple search strategy is also applicable to larger size biomolecular structure determinations.

7.5 Conclusions

In this chapter, the procedure of biomolecular structure determination from NMR data is firstly introduced. Then the potential energy function is explained and further developed. The distance and torsion angle constraints are proposed in novel presentations in the energy function which are convenient and practical to the optimization problem. This chapter also offers detailed transformation formulae from Cartesian coordinates to internal coordinates, which is important but usually neglected in the literature. Finally, the ripple search strategy is applied to determine the 3D structures of two organic molecules and one deoxyribonucleic acid 3REC with 189 variables in the potential energy function. All applications begin with random starting structures without simplifications of the potential energy function or assumptions on molecular structures. Therefore the structures found by ripple search strategy are of high quality and are nearly the same as the standard ones in the literature. The practical applicability of the developed potential energy function and

the good performance of the ripple search strategy on molecular structure determination are verified.

Chapter 8

Conclusions and Recommendations for Further Research

8.1 Conclusions

In the thesis, we have studied some related topics in global optimization, NMR data processing and biomolecular structure determination. Both the contributions in theory and applications have been presented.

Firstly, we have proposed five new and promising approaches to solve nonlinear global optimization problems.

In Chapter 2, a novel ripple search strategy has been proposed to escape from local solutions of global optimization problems. The proposed algorithm, as a stochastic global search method, combined a novel ripple search and a proper local optimization algorithm to find the global solution of a given optimization problem starting from a random initial point. The ripple search differs from those random search schemes which explore the feasible region randomly. Hence the proposed method does not suffer from the disadvantages of random search methods, such as the easiness of leading to the same local minimum and the inefficiency of implementation. Unlike some stochastic global optimization algorithms having many control parameters, such as simulated annealing and genetic algorithm, the

new algorithm requires only three control parameters. Therefore it is fairly easy to apply to diverse industrial applications. The proposed method has the inherent potential for parallel processing while many existing methods do not have. Though for a global optimization problem without any prior knowledge on the objective function or the global minimum, the method normally can not guarantee that the solution obtained in a finite time is the true global minimum, its probabilistic convergence to the global solution has been validated theoretically. Test results on extensive benchmarks have demonstrated the good performance and robustness of the proposed approach and shown that the ripple search strategy is the most accurate and efficient method among those global optimization algorithms under comparisons.

Because ripple search strategy develops all trial points taken in the search space without considering their worth for further investigation, its efficiency can be further improved by combining other techniques for identifying promising trial points or decreasing the number of trial points required for convergence to a good solution. To this end, four algorithms have been proposed in Chapters 3, 4 and 5 respectively to achieve the goal of speeding up the convergence of the global optimization algorithms, which is a key issue in global optimization.

In Chapter 3, subenergy tunneling function has been introduced and an improved subenergy tunneling function has been proposed to combine with the ripple search strategy to construct an efficient global optimization algorithm. The subenergy tunneling function of the original TRUST method has the transition regions in which the original function is modified asymptotically. For the improved subenergy tunneling function, the transition regions can be ignored. Though the improved subenergy tunneling function is not so smooth as the original subenergy tunneling function in the transition region, no problem would be caused to the minimization process. In the proposed algorithm, the improved subenergy tunneling function has been used to identify promising trial points and the ripple search strategy has been applied to get away from local minima. The global convergence properties of the proposed algorithm have been demonstrated through some frequently used neural network learning applications using small-sized neural networks.

The performance of the new method is in general better than that of other training methods in the literature. The drawback is that as more local minima are found, it is more difficult to locate a trial point in the promising region as the region shrinks exponentially with the dimensions of a problem along with the optimization process. This is also unavoidable in other methods trying to identify promising points in a similar way. However the proposed algorithm is conceptually simple and straightforward to implement.

In Chapter 4, two novel techniques on function extension have been proposed to overcome the limitation mentioned above. The two methods, non-iterative and iterative function extension, have been introduced, analyzed and illustrated with examples in detail. They have been embedded into global optimization. The non-iterative function extension cannot guarantee that the portion of the extended function in each attraction region is nearly flat and smooth because this highly depends on the setting of control parameters. Therefore in practice, it has been used to identify the promising points among all trial points generated during the global search. For the iterative function extension, as the portion of the extended function in each attraction region is nearly flat and smooth, this technique has been used to identify promising regions by evaluating a single point in each region. Consequently the convergence speed has been dramatically increased. Function extension techniques are more reliable than improved subenergy tunneling function for identifying promising trial points. Test results have demonstrated the good performance of the proposed approaches and shown that they are very promising in this area. The idea of function extension is innovative and there is no other techniques possessing similar properties as the proposed methods, to the best of our knowledge.

Global optimization algorithms, such as simulated annealing, are generally computationally expensive, especially when the number of dimensions of a problem is high. In Chapter 5, the novel idea of combining a global optimization algorithm with Parallel Coordinates has been explained and discussed in detail. In our proposed method, Parallel Coordinates has been employed to transform the original search domain into another search domain with lower dimensions. As the computational cost by global optimization algorithms may grow exponentially with the size of the problem, reduction on the dimensions of

the search domain by Parallel Coordinates has provided a means to speed up the convergence rate of global optimization algorithms. The number of trial points required for convergence to a good solution has also been decreased. The proposed method is meant to address optimization problems of moderate and high dimensions. Numerical studies have demonstrated the superior performance of the proposed method and shown that adopting Parallel Coordinates can significantly reduce the computational cost of global optimization algorithms. The technique of Parallel Coordinates has never been applied before to global optimization problems.

Secondly, the developed techniques have been applied to NMR data processing applications.

Classical statistical results have shown that the Cramer Rao Lower Bound provides a lower bound for the variance of a parameter estimation scheme, provided it is unbiased. In Chapter 6, we have given the analytical expression for the Fisher information matrix and have shown that the CRLB can be calculated explicitly for a parameter estimation problem for a general 2D NMR data set. Using Monte Carlo type simulations we have shown for a simulated example that the variance of estimates using a nonlinear least-squares parameter estimation algorithm in fact attain this bound. It has also shown how the developed results can be applied to study questions of experimental design. Using example systems we have used our CRLB results to show that a certain non-uniform sampling scheme in fact produces superior results over the uniform sampling scheme that was considered. We have also introduced (for the first time to the best of our knowledge) a non-uniform averaging scheme for traces. This has also been investigated in conjunction with non-uniform sampling to show that an improved CRLB can be obtained for the examined system.

When the initial parameter values are unknown, local search methods, such as the nonlinear least-square estimation algorithm, would not produce the desirable results on parameter estimation. In this case, global optimization is required. We have applied ripple search strategy to the simulation for parameter estimation of the simulated 2D NMR data. The effectiveness of finding good minima has been demonstrated and the performance of the

ripple search strategy has been compared with the exact parameter value and the results of other global optimization methods. These results have also been compared with the CRLB. The proposed method can be used straightforwardly to the parameter estimation of multidimensional NMR data.

Thirdly, the developed techniques have been applied to biomolecular structure determination applications.

In Chapter 7, the procedure of biomolecular structure determination from NMR data has been introduced. Then the potential energy function has been explained in detail and further developed. The distance and torsion angle constraints have been proposed in novel presentations in the energy function, which are convenient and practical to the optimization problem. Detailed transformation formulae from Cartesian coordinates to internal coordinates has also been offered, which is important but usually neglected in the literature. Finally, ripple search strategy has been applied to determine the 3D structures of two organic molecules, 1-Propanol and 2-Methyl-1-Propanol, and one deoxyribonucleic acid 3REC with 189 variables in the potential energy function. All applications begin with random starting structures without simplifications of the potential energy function or assumptions on molecular structures. Therefore the structures found by ripple search strategy are of high quality and are nearly the same as the standard ones in the literature. The practical applicability of the developed potential energy function and the good performance of the ripple search strategy on molecular structure determination are verified.

8.2 Recommendations for Further Research

Although we have tried to establish a complete theoretical framework and conduct a thorough analysis of the underlying topics, there still remain some problems to be further considered. In fact, the present work seems to open up a great number of new opportunities for further study. Some recommendations for further research are listed below.

- There are many types of optimization problems. The highly efficient optimization algorithms proposed in this thesis are mainly applied to continuous problems. We plan to extend the idea of our ripple search strategy and other proposed methods to solve discrete problems.

Additionally, the ripple search strategy has the potential for parallel processing. We plan to investigate parallel optimization algorithms on the base of ripple search strategy.

- In Chapter 4, the techniques of function extension need to be further tested on more benchmarks. In NMR data processing and biomolecular structure determination applications, only ripple search strategy is applied. The proposed methods in Chapters 3 to 5 are expected to be applied to these applications with ripple search strategy and other global optimization algorithms to further improve their efficiency.
- In the thesis, the simulations of parameter estimation by ripple search strategy for 2-D NMR Data in the cases of $K = L = 2$ are completed satisfactorily. The estimated parameters include amplitude, damping factor, angular frequency and the initial phase of NMR signal in Gaussian noise. The performance of ripple search strategy on more complicated experimental NMR signal requires further investigation in the future.
- The proposed methods are expected to be applied to structure determination for more complicated biomolecules. The biomolecular structure determination has many applications, such as disease diagnosis and structural-based drug design, which could be another interesting and valuable research direction.

Author's Publications

1. H. Ye and Z. Lin, "Speed-up Simulated Annealing by Parallel Coordinates," *European Journal of Operational Research*, in press.
2. R. J. Ober, Z. Lin, H. Ye and E. S. Ward, "Achievable Accuracy of Parameter Estimation for Multidimensional NMR Experiments", *Journal of Magnetic Resonance*, Vol. 157, pp. 1-16, 2002.
3. H. Ye and Z. Lin, "Global Optimization of Neural Network Weights using Subenergy Tunneling Function and Ripple Search", in *Proceedings of the IEEE ISCAS*, pp. 725-728, 2003.
4. H. Ye and Z. Lin, "A Stochastic Method for Minimizing Functions with Many Minima", in *Proceedings of the IEEE Workshop NNSP*, pp. 289-296, 2002.

Bibliography

- [1] J. D. Pinter, *Global Optimization in Action*, London: Kluwer Academic Publishers, 1996.
- [2] J. E. Dennis, “Nonlinear Least Squares”, in *State of the Art in Numerical Analysis*, D. Jacobs, Ed., London: Academic Press, pp. 269-312, 1977.
- [3] J. J. More, “The Levenberg-Marquardt Algorithm: Implementation and Theory”, in *Numerical Analysis: Lecture Notes in Mathematics 630*, G. A. Watson, Ed., New York: Springer-Verlag, pp. 105-116, 1977.
- [4] C. G. Broyden, “The Convergence of a Class of Double-Rank Minimization Algorithms”, *Journal of Institute Mathematical Application*, Vol. 6, pp. 76-90, 1970.
- [5] R. Fletcher, “A New Approach to Variable Metric Algorithms”, *Computer Journal*, Vol. 13, pp. 317-322, 1970.
- [6] D. Goldfarb, “A Family of Variable Metric Updates Derived by Variational Means”, *Mathematics of Computing*, Vol. 24, pp. 23-26, 1970.
- [7] D. F. Shanno, “Conditioning of Quasi-Newton Methods for Function Minimization”, *Mathematics of Computing*, Vol. 24, pp. 647-656, 1970.
- [8] K. Schittowski, “Solving Constrained Nonlinear least Squares Problems by a General Purpose SQP-Method”, *International Series of Numerical Mathematics*, Vol. 84, pp. 295-309, 1988.
- [9] C. A. Floudas, *Deterministic Global Optimization*, London: Kluwer Academic Publishers, 2000.

-
- [10] T. P. Boris, *Introduction to Optimization*, New York: Optimization Software, Publication Division, 1987.
- [11] A. T. Brunger, “X-PLOR, Version 3.1, A system for X-ray Crystallography and NMR”, *Yale University Press*, New Haven, CT, 1992.
- [12] P. Guntert, W. Braun and K. Wuthrich, “Efficient Computation of Three-dimensional Protein Structures In Solution From Nuclear Magnetic Resonance Data Using the Program DIANA and The Supporting Programs CALIBA, HABAS and GLOMSA”, *Journal of Molecular Biology*, Vol. 217, pp. 517–530, 1991.
- [13] A. Törn and A. Zilinskas, *Global Optimization*, Berlin: Springer-Verlag, 1989.
- [14] R. Horst and P. M. Pardalos, *Handbook of Global Optimization*, Boston: Kluwer Academic Publishers, 1995.
- [15] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*. Berlin: Springer-Verlag, 1993.
- [16] Atta-ur-Rahman and M. I. Choudhary, *Solving Problems with NMR Spectroscopy*, California: Academic Press, 1995.
- [17] J. Cavanagh, W. J. Fairbrother, A. G. Palmer and N. J. Skelton, *Protein NMR Spectroscopy*, San Diego: Academic Press, 1996.
- [18] T. Lengauer and R. Zimmer, “Protein Structure Prediction Methods For Drug Design”, *Briefings in Bioinformatics*, Vol. 1, pp. 275–288, 2000.
- [19] J. C. Whisstock and A. M. Lesk, “Prediction Of Protein Function From Protein Sequence And Structure”, *Review of Biophysics*, Vol. 36, pp. 307–340, 2003.
- [20] J. N. S. Evans, *Biomolecular NMR Spectroscopy*, Oxford University Press, 1995.
- [21] S. Kirkpatrick, C. Gelatt and P. Vecchi, “Optimization by Simulated Annealing”, *Science*, Vol. 220, pp. 671–679, 1983.

- [22] V. Cerny, “Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulated Algorithm”, *Journal of Optimization Theory and Applications*, Vol. 45, pp. 41-51, 1985.
- [23] N. Metropolis, A. N. Rosenbluth, M. N. Rosenbluth, A. H. Teller and H. Teller, “Equation of State Calculation by Fast Computing Machines”, *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092, 1953.
- [24] M. Lundy and A. Mees, “Convergence of an Annealing Algorithm”, *Mathematical Programming*, Vol. 34, pp. 111-124, 1986.
- [25] C. A. Floudas, P. M. Pardalos, C. S. Adjiman and W. R. Esposito, *Handbook of Test Problems in Local and Global Optimization*, Dordrecht: Kluwer Academic Publishers, 1999.
- [26] A. Corana, M. Marchesi, C. Martini and S. Ridella, “Minimizing Multimodal Functions of Continuous Variables with the ‘Simulated Annealing’ Algorithm”, *ACM Transactions on Mathematical Software*, Vol. 13, No. 3, pp. 262-280, 1987.
- [27] M. Locatelli, “Simulated Annealing Algorithms for Continuous Global Optimization: Convergence Conditions”, *Journal of Optimization Theory and Applications*, Vol. 104, No. 1, pp. 121-133, 2000.
- [28] L. Franconi and C. Jennison, “Comparison of a Genetic Algorithm and Simulated Annealing in an Application to statistical Image Reconstruction”, *Statistics and Computing*, Vol. 7, No. 3, pp. 193-207, 1997.
- [29] S. G. Ponnambalam, N. Jawahar and P. Aravindan, “A Simulated Annealing Algorithm for Job Shop Scheduling”, *Production Planning and Control*, Vol. 10, No. 8, pp. 767-777, 1999.
- [30] M.C. Robini, T. Rastello and I.E Magnin, “Simulated Annealing, Acceleration Techniques, and Image Restoration ”, *IEEE Transactions on Image Processing*, Vol. 8, No. 10, pp. 1374-1387, Oct. 1999.

-
- [31] W. X. Zhang, T. E. Smithgall and W. H. Gmeiner, "Three-dimensional Structure of the Hck SH2 Domain in Solution", *Journal of Biomolecular NMR*, Vol. 10, No. 3, pp. 263-272, 1997.
- [32] K. M. Cheh, J. B. Goldberg and R. G. Askin, "A Note on The Effect of Neighborhood Structure in Simulated Annealing", *Computers and Operations Research*, Vol. 18, No. 6, pp. 537-547, 1991.
- [33] M. W. Park and Y. D. Kim, "A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithms", *Computers and Operations Research*, Vol. 25, No. 3, pp. 207-217, 1998.
- [34] L. Ingber, "Adaptive Simulated Annealing (ASA): lessons Learned", *Journal of Control and Cybernetics*, Vol. 25, No. 1, pp. 33-54, 1996.
- [35] R. Storn and K. Price, "Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces", *Journal of Global optimization*, Vol. 11, No. 4, pp. 341-359, December 1997.
- [36] The MathWorks Technical Staff, *MATLAB Reference Guide*, The MathWorks, Inc., 1994.
- [37] G. G. Judge, W. E. Griths, R. C. Hill, H. Lutkepohl and T. Lee, *The Theory and Practice of Econometrics*, Second Edition, New York: John Wiley and Sons, 1985.
- [38] W. L. Goffe, G. D. Ferrier and J. Rogers, "Global Optimization of Statistical Functions with Simulated Annealing", *Journal of Econometrics*, Vol. 60, pp. 65-99, 1994.
- [39] F. M. Rabinowitz, "Algorithm 744: A Stochastic Algorithm for Global Optimization with Constraints", *ACM Transactions on Mathematical Software*, Vol. 21, No. 2, pp. 194-213, 1995.
- [40] G. L. Bilbro, "Fast Stochastic Global Optimization", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, pp. 684-689, Apr. 1994.

- [41] P. R. Chowdhury, Y. P. Singh and R. A. Chansarkar, “Hybridization of Gradient Descent Algorithm with Dynamic Tunneling Methods for Global Optimization”, *IEEE Transactions on Systems, Man and Cybernetics—Part A*, Vol. 30, pp. 384–390, May 2000.
- [42] E. M. Oblow, “SPT: A Stochastic Tunneling Algorithm for Global Optimization”, *Journal of Global Optimization*, Vol. 20, pp. 195–212, May 2001.
- [43] J. Barhen, V. Protopopescu and D. Reister, “TRUST: A Deterministic Algorithm for Global Optimization”, *Science*, Vol. 276, pp. 1094–1097, 1997.
- [44] B. C. Cetin, J. Barhen and J. W. Burdick, “Terminal Repeller Unconstrained Subenergy Tunneling (TRUST) for Fast Global Optimization”, *Journal of Optimization Theory and Applications*, Vol. 77, No. 1, pp. 97–126, 1993.
- [45] J. Barhen and V. Protopopescu, “Generalized TRUST Algorithms for Global Optimization”, in *State of the Art in Global Optimization: Computational Methods and Applications*, C. A. Floudas and P. M. Pardalos, Eds., Netherlands: Kluwer Academic Press, pp. 163–180, 1996.
- [46] B. C. Cetin, J. W. Burdick and J. Barhen, “Global Descent Replace Gradient Descent to Avoid Local Minima Problem in Learning with Artificial Neural Networks”, *Proceedings of IEEE International Conference on Neural Networks*, Vol. 2, pp. 836–842, 1993.
- [47] K. W. Tang, S. Kak, “Fast Classification Networks For Signal Processing”, *Circuits, Systems, and Signal Processing*, Vol. 21, No. 2, pp. 207–224, 2002.
- [48] H. Xiong, M. N. S. Swamy, M. O. Ahmad and K. Irwin, “Branching Competitive Learning Network: A Novel Self-creating Model”, *IEEE Transactions on Neural Networks*, Vol. 15, No. 2, pp. 417–429, 2004.
- [49] D. He, C. He, L. Jiang, et al., “A New Phase Acquisition Method of Address Code Utilizing the Combination of Logistic Map and BP Network”, *Circuits, Systems, and Signal Processing*, Vol. 22, No. 1, pp. 1–17, 2003.

- [50] Y. Shang and B. W. Wah, "Global Optimization for Neural Network Training", *IEEE Computer*, Vol. 29, pp. 45–54, 1996.
- [51] S. Abid, F. Fnaiech and M. Najim, "A Fast Feedforward Training Algorithm Using a Modified Form of the Standard Backpropagation Algorithm", *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pp. 424–430, Mar. 2001.
- [52] Y. X. Chen and B. M. Wilamowski, "TREAT: A Trust-Region-Based Error-Aggregated Training Algorithm for Neural Networks", *Proceedings of the 2002 International Joint Conference on Neural Networks*, Vol. 2, pp. 1463–1468, 2002.
- [53] H. Demuth and M. Beale, *Neural Network Toolbox For Use With MATLAB*, Massachusetts: The MathWorks Inc., 2002.
- [54] N. Ampazis and S. J. Perantonis, "Two Highly Efficient Second-order Algorithms for Training Feedforward networks", *IEEE Transactions on Neural Networks*, Vol. 13, No. 5, pp. 1064–1074, Sept. 2002.
- [55] J. Bilski and L. Rutkowski, "A Fast Training Algorithm for Neural Networks", *IEEE Transactions on Circuits and Systems II : Analog and Digital Signal Processing*, Vol. 45, No. 6, pp. 749–753, Jun. 1998.
- [56] N. S. Rubanov, "The Layer-Wise Method and the Backpropagation Hybrid Approach to Learning a Feedforward Neural Network", *IEEE Transactions on Neural Networks*, Vol. 11, No. 2, pp. 295–305, Mar. 2000.
- [57] Z. Tang and G. J. koehler, "Deterministic Global optimal FNN Training Algorithms", *Neural Networks*, Vol. 7, No. 2, pp. 301–311, 1994.
- [58] N. K. Treadgold and T. D. Gedeon, "Simulated Annealing And Weight Decay In Adaptive Learning: The SARPROP Algorithm", *IEEE Transactions on Neural Networks*, Vol. 9, pp. 662 – 668, 1998.
- [59] K. A. Toh, "Deterministic Global Optimization For FNN Training", *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, Vol. 33, pp. 977–983, 2003.

- [60] H. Ye and Z. Lin, “A Stochastic Method for Minimizing Functions with Many Minima”, *Proceedings of IEEE International Workshop on Neural Networks for Signal Processing*, pp. 289–296, Sep. 2002.
- [61] K. A. Toh, “Fast Deterministic Global Optimization For FNN Training”, *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5, pp. 413–418, 1999.
- [62] T. Kozek, T. Roska and L. O. Chua, “Genetic Algorithm for CNN Template Learning”, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 40, No. 6, pp. 392–402, Jun. 1993.
- [63] N. E. Mastorakis, I. F. Gonos and M. N. S. Swamy, “Design of Two-dimensional Recursive Filters Using Genetic Algorithms”, *IEEE Transactions on Circuits and Systems I*, Vol. 50, No. 5, pp. 634–639, 2003.
- [64] D. Rumelhart, G. Hinton, and R. Williams, “Learning Internal Representations by Error Propagation, in Parallel Distributed Processing”, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1, MIT Press, Cambridge, pp. 318–362, 1986.
- [65] R. P. Gorman and T. J. Sejnowski, “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets” *Neural Networks*, Vol. 1, pp. 75–89, 1988.
- [66] C. C. Chuang, S. F. Su and C. C. Hsiao, “The Annealing Robust Backpropagation (ARBP) Learning Algorithm”, *IEEE Transactions on Neural Networks*, Vol. 11, No. 5, pp. 1067–1077, Sept. 2000.
- [67] D. S. Chen, and R. C. Jain, “A Robust Backpropagation Learning Algorithm for Function Approximation”, *IEEE Transactions on Neural Networks*, Vol. 5, No. 3, pp. 467–476, May 1994.
- [68] K. J. Lang and M. J. Witbrock, “Learning to Tell Two Spirals Apart”, *Proceeding of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, pp. 52–59, 1988.

- [69] B. Baum and K. J. Lang, “Constructing Hidden Units using Examples and Queries”, *Proceedings of Advances in Neural Information Processing Systems*, Vol. 3, pp. 904–910, 1991.
- [70] S. E. Fahlman and C. Lebiere, “The Cascade-Correlation Learning Architecture”, *Proceedings of Advances in Neural Information Processing Systems*, Vol. 2, pp. 524–532, 1990.
- [71] E. mizutani, “Powell’s Dogleg Trust-region Steps with the Quasi-Newton Augmented Hessian for Neural Nonlinear Least-Squares Learning”, *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 2, pp. 1239–1244, 1999.
- [72] X. Zhang and Y. Q. Chen, “Ray-guided Global Optimization Method for Training Neural Networks”, *Neurocomputing*, No. 30, pp. 333–337, 2000.
- [73] Y. Yao, “Dynamic Tunneling Algorithm for Global Optimization”, *IEEE Transactions on System, Man, and Cybernetics*, Vol. 19, No. 5, pp. 1222–1230, Spet. - Oct. 1989.
- [74] W. Schroeder, K. Martin and B. Lorensen, *The visualization toolkit*, New Jersey: Prentice Hall PTR, pp. 400–401, 1998.
- [75] A. Inselberg and B. Dimsdale, “Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry”, *Proceedings of the First IEEE Conference on Visualization*, pp. 361–378, 1990.
- [76] A. Inselberg, “N-dimensional Graphics, Part I—Lines and Hyperplanes”, IBM Los Angeles Scientific Center, IBM Scientific Center, 9045 Lincoln Boulevard, Los Angeles (CA), 900435, Technical Report G320-2711, 1981.
- [77] E. J. Wegman, “Hyperdimensional Data Analysis Using Parallel Coordinates”, *Journal of the American Statistical Association*, Vol. 85, pp. 664–675, 1990.
- [78] J. S. Eickemeyer, A. Inselberg and B. Dimsdale, “Visualizing P-flats in N-space using Parallel Coordinates”, IBM Palo Alto Scientific Center, Technical Report G320-3581, 1992.

- [79] E. J. Wegman and Q. Luo, “High Dimensional Clustering using Parallel Coordinates and the Grand Tour”, *Computing Science and Statistics*, Vol. 28, pp. 352–360, 1997.
- [80] K. D. Hjelmstad, “Chapter 6: The Ritz Method of Approximation”, in *Fundamentals of Structural Mechanics*, New Jersey: Prentice-Hall, pp. 157–186, 1997.
- [81] H. Szu and R. Hartley, “Fast Simulated Annealing”, *Physical Letters A*, Vol. 122, No. 3, pp. 157–162, 1987.
- [82] Y. Yao, “Dynamic Tunneling Algorithm for Global Optimization”, *IEEE Transactions on System, Man, and Cybernetics*, Vol. 19, No. 5, pp. 1222–1230, 1989.
- [83] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, Inc., 1989.
- [84] T. F. Coleman, M. A. Branch and A. Grace, *Optimization Toolbox For Use With MATLAB. User’s Guide.*, Version 2, The MathWorks Inc., Natick, MA, 1999.
- [85] E. Kandogan, “Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions”, IBM Almaden research Center, <http://people.cs.vt.edu/north/infoviz>.
- [86] N. Thandra and A. Bax, “Direct Measurement of Distances and Angles in Biomolecules by NMR in a Dilute Liquid Crystalline Medium”, *Science*, Vol. 278, pp. 1111–1114, 1997.
- [87] A. Bax and N. Tjandra, “Are Proteins Even Floppier than We Thought?”, *Nature Structural Biology*, Vol. 4, pp. 254–256, 1997.
- [88] D. Jin and M. Andrec G. T. Montelione and R. M. Levy, “How Sensitive are NMR Relaxation Measurements to Protein Dynamics?”, *Abstract to Plenary Lecture Delivered at the 1998 ENC*, 1998.
- [89] H. Barkhuijsen and R. de Beer and D. van Ormondt, “Error Theory for Time-domain Signal Analysis with Linear Prediction and Singular Value Decomposition”, *Journal of Magnetic Resonance*, Vol. 67, pp. 371–375, 1986.

- [90] S. Cavassila, S. Deval, C. Huegen, D. van Ormondt and D. Graveron-Demilly, “Cramér-Rao Lower Bound Expressions for Parametric Estimation of Overlapping Peaks: Influence of Prior Knowledge”, *Journal of Magnetic Resonance*, Vol. 143, pp. 311–320, 2000.
- [91] J. C. Hoch and A. S. Stern, *NMR Data Processing*, New York: John Wiley-Sons, 1996.
- [92] R. R. Ernst, G. Bodenhausen and A. Wokaun, *Principles of Nuclear Magnetic Resonance in One and Two Dimensions*, Oxford: Oxford University Press, 1994.
- [93] R. J. Ober and E. S. Ward, “System Theoretic Formulation of NMR Experiments”, *Journal of Mathematical Chemistry*, Vol. 20, pp. 47–65, 1996.
- [94] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, New Jersey: Prentice Hall, 1993.
- [95] R. J. Ober, Z. Lin, H. Ye and E. S. Ward, “Achievable Accuracy of Parameter Estimation for Multidimensional NMR Experiments”, *Journal of Magnetic Resonance*, Vol. 157, pp. 1–16, 2002.
- [96] J. Cavanagh et al., *Protein NMR Spectroscopy: Principles and Practice*, San Diego: Academic Press, 1996.
- [97] K. Steiglitz, “On the Simultaneous Estimation of Poles and Zeros in Speech Analysis”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 25, No. 3, pp. 229–234, Jun. 1977.
- [98] T. L. Henderson, “Geometric Methods for Determining System Poles from Transient Response”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 29, pp. 982–988, Oct. 1981.
- [99] R. Kumaresan and R. W. Tufts, “Estimating the Parameters of Exponentially Damped Sinusoids and Pole-zero Modeling in Noise”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 30, pp. 833–840, Dec. 1982.

-
- [100] D. S. Stephenson, "Linear Prediction and Maximum Entropy Methods in NMR Spectroscopy", *Progress in NMR spectroscopy*, Vol. 20, pp. 515–626, 1988.
- [101] M. M. Barbieri and P. Barone, "A Two-dimensional Prony's Method for Spectral Estimation", *IEEE Transactions on Signal Processing*, Vol. 40, pp. 2747–2756, Nov. 1992.
- [102] J. J. Sacchini, W. M. Steedly, and R. L. Moses, "Two-dimensional Prony Modeling and Parameter Estimation", *IEEE Transactions on Signal Processing*, Vol. 41, pp. 3127–3137, Nov. 1993.
- [103] H. Barkhuijsen, R. De Beer, and D. V. Ormondt, "Improved Algorithm for Non-iterative Time-domain Model Fitting to Exponentially Damped Magnetic Resonance Signals", *Journal of Magnetic Resonance*, Vol. 73, pp. 553–557, 1987.
- [104] S. Kay and R. Nekovei, "An Efficient Two-Dimensional Frequency Estimator", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 38, No. 10, pp. 1807–1809, Oct. 1990.
- [105] Y. Hua, "Estimating Two-Dimensional Frequency by Matrix Enhancement and Matrix Pencil", *IEEE Transactions on Signal Processing*, Vol. 40, No. 9, pp. 2267–2280, Sept. 1992.
- [106] J. Razavilar, Y. Li and K. J. R. Liu, "Parameter Estimation of NMR spectroscopy using Matrix Pencil", *Proceedings of IEEE-EMBC and CMBEC 17th Annual Conference*, Vol.1, No.8, pp. 493–494, 1997.
- [107] P. R. Adby and M. A. H. Dempster, *Introduction to Optimization Method*, London: Chapman and Hall, 1982.
- [108] L. He and E. Polak, "Multistart Method with Estimation Scheme for Global Satisfying Problems", *Journal of Global optimization*, No.3, pp. 139–156, 1993.
- [109] C. Mullis and R. Roberts, "Synthesis of Minimum Round-off Noise Fixed Point Digital Filters", *IEEE Transactions on Circuits and Systems*, Vol. CAS-23, No. 9, pp. 551–562, Sept. 1976.

-
- [110] L. Thiele, “On the Sensitivity of Linear State-space Systems”, *IEEE Transactions on Circuits and Systems*, Vol. CAS-33, pp. 502-510, May 1986.
- [111] R. R. Wilcox, *Introduction to Robust Estimation And Hypothesis Testing*, New York: Academic Press, 1997.
- [112] G. Wider, “Structure Determination of Biological Macromolecules in Solution using NMR Spectroscopy”, *BioTechniques*, Vol. 29, pp. 1278–1294, 2000.
- [113] I. Ono, H. Fujiki, M. Ootsuka, N. Nakashima, N. Ono and S. Tate, “Global Optimization of Protein 3-Dimensional Structures in NMR by A Genetic Algorithm”, *Proceedings of IEEE Congress on Evolutionary Computation*, pp.303–308, 2002.
- [114] Roger Sayle and E. James Milner-White, “RasMol: Biomolecular Graphics for All”, *Trends in Biochemical Sciences*, Vol. 20, No. 9, pp. 374–376, Sept. 1995.
- [115] Herbert J. Bernstein, “Recent Changes to RasMol, Recombining the Variants,” *Trends in Biochemical Sciences (TIBS)*, Vol. 25, No. 9, pp. 453-455, Sept. 2000.
- [116] J. L. Klepeis, C. A. Floudas, D. Morikis and J. D. Lambris, “Predicting Peptide Structures using NMR Data and Deterministic Global Optimization”, *Journal of Computational Chemistry*, Vol. 20, No. 13, pp. 1354–1370, 1999.
- [117] T. Pupko, I. Pe’er, M. Hasegawa, D. Graur and N. Friedman, “A Branch-and-bound Algorithm For The Inference Of Ancestral Amino-acid Sequences when The Replacement Rate Varies Among Sites ”, *Bioinformatics*, Vol. 18, pp. 1116–1123, 2002.
- [118] G. B. Fogel, D. G. Weekes, R. Sampath and D. J. Ecker, “Parameter Optimization of an Algorithm for RNA Structure Discovery”, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 607–613, 2004.
- [119] S. Salzberg, D. Searls and S. Kasif, *Computational Methods in Molecular Biology*, Elsevier Science, 1998.

-
- [120] W. H. Wong, “Computational Molecular Biology”, *Journal of the American Statistical Association*, Vol. 95, No. 449, pp. 322–326, 2000.
- [121] L. Krippahl and P. Barahona, “PSICO: Solving Protein Structures With Constraint Programming And Optimization”, *Constraints*, Vol. 7, pp. 317–331, 2002.
- [122] K. F. Lim, G. Beliakov and L. M. Batten, “Predicting Molecular Structures: An Application of The Cutting Angle Method”, *Physical Chemistry Chemical Physics*, Vol. 5, pp. 3884–3890, 2003.
- [123] J. L. Klepeis, M. G. Ierapetritou and C. A. Floudas, “Global Optimization Approaches in Protein Folding and Peptide Docking”, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 47, pp. 141–148, 1999.
- [124] R. H. Byrd, E. Eskow, A. van der Hoek, R. B. Schnabel, C. Shao and Z. Zou, “Global Optimization Methods for Protein Folding Problems”, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 23, pp. 29–39, 1995.
- [125] K. A. Dill, A. T. Phillips and J. B. Rosen, “Molecular Structure Prediction By Global Optimization”, *Developments in Global Optimization*, pp. 217–234, 1997.
- [126] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, Jr., M.D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, “The Protein Data Bank: A Computer-based Archival File for Macromolecular Structures”. *Journal of Molecular Biology*, Vol. 112, pp. 535–542, 1977.
- [127] A. T. Phillips, J. B. Rosen and V. H. Walke, “Molecular Structure Determination by Convex Global Underestimation of Local Energy Minima”. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 23, pp. 181–198, 1995.
- [128] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Third Edition, Baltimore: Johns Hopkins University Press, 1996.

-
- [129] Z. Wu, “Linear Algebra in Biomolecular Modeling”, to be published in *Handbook of Linear Algebra*, L. Hogben, Ed., Chapman/Hall CRC press, 2006, <http://mbi.osu.edu/publications/techreport%2034.doc>.
- [130] J. J. More and Z. Wu, “Distance Geometry Optimization for Protein Structures”. *Journal of Global Optimization*, Vol. 15, pp. 219–234, 1999.
- [131] T. Nishinaka, Y. Ito, S. Yokoyama and T. Shibata, “An Extended DNA Structure Through Deoxyribose-Base Stacking Induced by RecA Protein”. *Proceeding of National Academy of Sciences*, Vol. 94, pp. 6623–6628, 1997.