

APPROXIMATING THE OBJECTIVE GRADIENT USING PERCEPTRONS FOR
CONSTRAINED MINIMIZATION WITH APPLICATION IN DRAG REDUCTION

by

Burak Kocuk

B.S., Industrial Engineering, Boğaziçi University, 2011

B.S., Mathematics, Boğaziçi University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University

2012

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my thesis supervisor Prof. İ. Kuban Altinel for his encouragement and guidance throughout my undergraduate and graduate studies. I appreciate the opportunity he gave to me to work on a unique research area.

I would like to thank Prof. Necati Aras for his valuable comments throughout my thesis studies and Prof. Ethem Alpaydın for taking part in my defense jury.

I wish to thank Prof. Sedat Tardu for providing an interesting problem from fluid mechanics and helping us to understand the technical details. I am indebted to Frédéric Bauer and Olivier Doche for conducting the necessary simulation runs.

I owe thanks to all my friends both on and off campus for their support. I am also thankful to my fellow assistants for their help whenever I needed.

I would like to express my gratitude to my family, especially my mother and sister, for their continuous support. Without these two wonderful ladies, I would never become the person I am now.

This thesis is partially supported by TÜBİTAK under BİDEB 2210 program.

ABSTRACT

APPROXIMATING THE OBJECTIVE GRADIENT USING PERCEPTRONS FOR CONSTRAINED MINIMIZATION WITH APPLICATION IN DRAG REDUCTION

This thesis is concerned with the constrained minimization problems whose objective function does not have a closed form analytical expression. However, it can be evaluated numerically which enables the determination of an estimator using random samples. We assume that the set of constraints are well-defined and explicitly known and use perceptrons to estimate the objective function. The new method, which we call Neural Reduced Gradient Algorithm, approximates the reduced gradient by combining the gradient value of the estimated objective function with the exact gradients of the constraints. It is tested with some randomly generated and also, some available convex and nonconvex optimization problems. We also provide an interesting real application of the new method in shear stress minimization for drag reduction and compare its performance with the ones obtained using linear estimators, which are proposed as a part of this work.

ÖZET

KISITLI ENKÜÇÜKLEME İÇİN AMAÇ FONKSİYONU GRADYANININ ALGILAYICI KULLANILARAK YAKLAŞIKLANMASI VE SÜRÜKLEMENİN AZALTILMASINA UYGULANMASI

Bu tez, kapalı biçimde analitik ifadesi olmayan kısıtlı enküçükleme problemleri ile ilgilidir. Ancak bu tür amaç fonksiyonlarının sayısal olarak hesaplanabildikleri ve yaklaşıkleyen kestiricilerinin elde edilmesine olanak veren rassal örneklemeler oluşturulabildiği varsayılmaktadır. Çalışmada kestirici olarak kullanılan algılayıcılar bulunmakta ve bunların yardımıyla gradyan yaklaşımaktadır. Sinir Ağı İndirgenmiş Gradyan Algoritması adını verdiğimiz yeni yöntem, indirgenmiş gradyanı algılayıcılar kullanılarak elde edilen yaklaşık amaç fonksiyonunun gradyanı ile kısıtların gerçek gradyanlarının birleşimiyle kestirir. Yöntemin doğruluğu, var olan ve rassal olarak üretilen dışbükey ve dışbükey olmayan deney problemleriyle sınımlanmaktadır. Yeni yöntemin akışkanlarda kesme gerilimini enküçükleyecek sürüklemeyi azaltma uygulaması sunulmakta ve başarımı, bu çalışmanın bir parçası olarak önerilen doğrusal kestirimler ile elde edilen sonuçlarla karşılaştırılmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF SYMBOLS	xv
LIST OF ACRONYMS/ABBREVIATIONS	xix
1. INTRODUCTION	1
2. LITERATURE SURVEY	4
2.1. Simulation Optimization	4
2.2. Piecewise Linear Approximation	7
2.3. Curve Fitting	7
2.4. Microfluidics	8
3. NEURAL REDUCED GRADIENT METHOD	11
3.1. Characteristics of Simulators and Appropriate Solution Methods	11
3.2. Use of Neural Networks	12
3.2.1. Construction of the Neural Network	13
3.2.2. Estimation of the Objective Gradient	14
3.2.2.1. Finite Difference	15
3.2.2.2. Exact Gradient of the Perceptron	15
3.3. Extensions of Neural Reduced Gradient	16
4. DRAG REDUCTION	17
4.1. Optimization Problem	21
4.2. Constrained Least Squares Estimation	23
4.2.1. Linear Estimation	27
4.2.2. Quadratic Estimation	27
4.2.3. Corrected Linear Estimation	28
4.2.4. Corrected Quadratic Convex Estimation	30

4.2.5.	Piecewise Linear Regression	31
4.2.6.	Summary of Constrained Least Squares Estimation	34
4.2.7.	Solution Methods for Actuator Management Problem Using Con- strained Least Squares Estimation	35
4.2.7.1.	Solution for Linear, Corrected Linear and Quadratic Convex Estimation	35
4.2.7.2.	Solution for Quadratic Estimation	35
4.2.7.3.	Solution for Piecewise Linear Regression	37
4.3.	Neural Networks	40
4.3.1.	Construction and Training of Neural Networks for Actuator Man- agement Problem	41
4.3.2.	Solution Method for Actuator Management Problem Using Neu- ral Networks	44
5.	COMPUTATIONAL EXPERIMENTS	46
5.1.	Test for Neural Reduced Gradient Method	46
5.1.1.	Test Instances for Neural Reduced Gradient Method	46
5.1.2.	Generation of Feasible Patterns	50
5.1.3.	Result of Test Instances	51
5.1.3.1.	Deterministic Case	52
5.1.3.2.	Random Case	66
5.2.	A Real Application: Drag Reduction	83
5.2.1.	Experiment 1	84
5.2.2.	Experiment 2	85
5.2.3.	Experiment 3	86
5.2.4.	Experiment 4	87
5.2.5.	Experiment 5	88
5.2.6.	Experiment 6	89
5.2.7.	Experiment 7	90
5.2.8.	Experiment 8	91
5.2.9.	Experiment 9	92
5.2.10.	Experiment 10	93

5.2.11. Experiment 11	94
6. CONCLUSIONS	95
APPENDIX A: PIECEWISE LINEAR APPROXIMATION MODELS	97
A.1. Disaggregated Convex Combination Models	97
A.2. Multiple Choice Model	99
A.3. Incremental Model	100
A.4. Specially Ordered Sets of Type 2 Model	101
A.5. Selection of the Most Suitable Model	101
APPENDIX B: OPTIMAL SOLUTIONS OF THE TEST PROBLEMS	104
REFERENCES	106

LIST OF FIGURES

Figure 4.1.	Average shear stress over the grid obtained by the reference LEGI simulation.	18
Figure 4.2.	Regions for a 4×4 grid with $r = 1$	19
Figure 4.3.	A part of the actual grid. A point and its neighbours, which lie in the red circle, are depicted.	20
Figure 4.4.	Adjusted R^2 values for LE.	27
Figure 4.5.	Adjusted R^2 values for QE.	28
Figure 4.6.	Sample S_{it} functions in (a), (b), (c) and (d) obtained using LE. . .	28
Figure 4.7.	Adjusted R^2 values for CLE.	29
Figure 4.8.	Sample S_{it} functions in (a), (b), (c) and (d) obtained using QE. .	30
Figure 4.9.	Adjusted R^2 values for CQCE.	31
Figure 4.10.	Sample S_{it} function obtained using PLR.	32
Figure 4.11.	Adjusted R^2 values for PLR with $K = 8$	33
Figure 4.12.	Adjusted R^2 values for PLR with $K = 8$ using three data sets. . .	34
Figure 4.13.	An example for the construction of neural network.	41

Figure 4.14.	R^2 values for neural networks.	43
Figure 5.1.	Measures for test problems using 1000 deterministic patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.	59
Figure 5.2.	Measures for test problems using 2000 deterministic patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.	64
Figure 5.3.	Measures for test problems using finite difference: (a) three replications, (b) five replication and (c) seven replications.	70
Figure 5.4.	Measures for test problems using 1000 random patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.	76
Figure 5.5.	Measures for test problems using 2000 random patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.	81
Figure 5.6.	Graphs for Experiment 1: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	84
Figure 5.7.	Graphs for Experiment 2: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	85
Figure 5.8.	Graphs for Experiment 3: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	86

Figure 5.9.	Graphs for Experiment 4: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	87
Figure 5.10.	Graphs for Experiment 5: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	88
Figure 5.11.	Graphs for Experiment 6: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	89
Figure 5.12.	Graphs for Experiment 7: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.	90
Figure 5.13.	Graphs for Experiment 8: (a) energy usage and (b) actual shear stress.	91
Figure 5.14.	Graphs for Experiment 9: (a) energy usage and (b) actual shear stress.	92
Figure 5.15.	Graphs for Experiment 10: (a) energy usage and (b) actual shear stress.	93
Figure 5.16.	Graphs for Experiment 10: (a) energy usage and (b) actual shear stress.	94

LIST OF TABLES

Table 4.1.	Region counts and their sizes.	19
Table 4.2.	Parameters of the optimization problem.	21
Table 4.3.	Summary of CLSE.	34
Table 4.4.	Region and pattern numbers.	42
Table 4.5.	Regions and actual patterns numbers.	42
Table 4.6.	Parameters of neural networks.	43
Table 5.1.	Results summary for a deterministic, inexpensive simulator.	52
Table 5.2.	Parameters of neural networks for NRG.	53
Table 5.3.	Training time of neural networks with deterministic inputs.	54
Table 5.4.	Results for a deterministic simulator with 1000 patterns and NNRT is used with FD method.	55
Table 5.5.	Results for a deterministic simulator with 1000 patterns and NNRT is used with EG method.	56
Table 5.6.	Results for a deterministic simulator with 1000 patterns and NNCT is used with FD method.	57

Table 5.7.	Results for a deterministic simulator with 1000 patterns and NNCT is used with EG method.	58
Table 5.8.	Results for a deterministic simulator with 2000 patterns and NNRT is used with FD method.	60
Table 5.9.	Results for a deterministic simulator with 2000 patterns and NNRT is used with EG method.	61
Table 5.10.	Results for a deterministic simulator with 2000 patterns and NNCT is used with FD method.	62
Table 5.11.	Results for a deterministic simulator with 2000 patterns and NNCT is used with EG method.	63
Table 5.12.	Results summary for a deterministic simulator with 1000 patterns.	65
Table 5.13.	Results summary for a deterministic simulator with 2000 patterns.	65
Table 5.14.	Results summary for a random, inexpensive simulator.	66
Table 5.15.	Solution for a random, inexpensive simulator with three replications.	67
Table 5.16.	Solution for a random, inexpensive simulator with five replications.	68
Table 5.17.	Solution for a random, inexpensive simulator with seven replications.	69
Table 5.18.	Training time of neural networks with random inputs.	71
Table 5.19.	Results for a random simulator with 1000 patterns and NNRT is used with FD method.	72

Table 5.20.	Results for a random simulator with 1000 patterns and NNRT is used with EG method.	73
Table 5.21.	Results for a random simulator with 1000 patterns and NNCT is used with FD method.	74
Table 5.22.	Results for a random simulator with 1000 patterns and NNCT is used with EG method.	75
Table 5.23.	Results for a random simulator with 2000 patterns and NNRT is used with FD method.	77
Table 5.24.	Results for a random simulator with 2000 patterns and NNRT is used with EG method.	78
Table 5.25.	Results for a random simulator with 2000 patterns and NNCT is used with FD method.	79
Table 5.26.	Results for a random simulator with 2000 patterns and NNCT is used with EG method.	80
Table 5.27.	Results summary for a random simulator with 1000 patterns. . . .	82
Table 5.28.	Results summary for a random simulator with 2000 patterns. . . .	82
Table A.1.	Comparison of piecewise linear approximation models for 12 problems in CPU seconds.	102
Table A.2.	Average performance comparison of piecewise linear approximation models for 12 problems in CPU seconds.	103

LIST OF SYMBOLS

A	constraint matrix
a_t	parameter vector for period t
a_t	common intercept parameter for period t
a_t^k	intercept parameter of line segment k for period
a_{ijt}^k	slope parameter of line segment k during period t for hole pair (i, j)
a_{ijt}^{+1}	regression parameter of the positive linear term during period t for hole pair (i, j)
a_{ijt}^{+2}	regression parameter of the quadratic linear term during period t for hole pair (i, j)
a_{ijt}^{-1}	regression parameter of the negative linear term during period t for hole pair (i, j)
a_{ijt}^{-2}	regression parameter of the negative quadratic term during period t for hole pair (i, j)
b	right-hand side vector
c	cost vector
$\tilde{\mathbf{c}}$	random cost vector
ce_{it}^k	intercept parameter of line segment k used in approximation of energy function of hole i during period t
cs_{it}^k	intercept parameter of line segment k used in approximation of total shear effect function of hole i during period t
D	diagonal matrix
D	number of binary variables in Log and DLog formulations
$\mathcal{D}$	index set for binary variables in Log and DLog formulations
E	total available energy
e_{it}	energy function of hole i during period t
$\mathbf{e}_j$	j^{th} unit vector
E_t	total available energy during period t
F	objective function

f_{it}	air flow through hole i during period t
f_{it}^k	air flow through hole i during period t at break point k
$\mathbf{F}_t$	air flow matrix for period t
g	activation function between input and hidden layers
G	activation function between hidden and output layers
H	number of hidden units
I	number of holes on the membrane
$\mathbf{I}$	index set of holes on the membrane
K	number of line segments
$\mathbf{K}$	index set for line segments
$\mathbf{l}$	lower bound vector
$\mathbf{l}'$	random lower bound vector
l_{it}	minimum air flow through hole i during period t
m	number of constraints
M	number of regions
me_{it}^k	slope parameter of line segment k used in the approximation of energy function of hole i during period t
ms_{it}^k	slope parameter of line segment k used in the approximation of total shear effect function of hole i during period t
N_m	largest number of neighbours
n	number of variables
$\mathbf{N}_i$	index set for neighbours of hole i
O	fitted objective function
$\bar{O}$	estimator of the mean of outputs
O^p	output of pattern p
$\hat{O}^p$	normalized output of pattern p
P	polytope defining the constraint set
P'	random polytope
r	radius of neighbourhood
R^2	coefficient of determination
Rem	remainder function

s	estimator of the standard deviation of outputs
s_j	estimator of the standard deviation of inputs with respect to variable j
s_{ijt}	shear stress effect function of hole i exerted at point j during period t
S_{it}	total shear stress effect function of hole i during period t
$\mathbf{SS}_t$	shear stress vector for period t
SS_{jt}	shear stress at hole j during period t
$\widehat{\mathbf{SS}}_t$	shear stress estimate vector for period t
$\widehat{SS}_{jt}$	shear stress estimate at hole j during period t
T	number of periods
$\mathbf{T}$	index set for periods
$\mathbf{u}$	upper bound vector
$\mathbf{u}'$	random upper bound vector
u_{it}	maximum air flow through hole i during period t
V_h	value of the hidden unit h
W_h	connection weight from hidden unit h to output unit
w_{hj}	connection weight from input unit j to hidden unit h
$\mathbf{x}$	unknown vector
$\mathbf{x}^*$	optimal solution
X^i	abscissa of hole i
$\bar{x}_j$	estimator of the mean of the patterns with respect to the j^{th} input
$\hat{x}_j^p$	normalized j^{th} input of pattern p
$\mathbf{x}_1$	best solution with respect to fitted function
$\mathbf{x}_2$	best solution with respect to actual function
$\mathbf{x}_1^N$	nearest local minimum to point $\mathbf{x}_1$
$\mathbf{x}_2^N$	nearest local minimum to point $\mathbf{x}_2$
Y^i	ordinate of hole i
y_{it}^d	binary variable used in Log and DLog formulations
y_{it}^k	binary variable used in CC, DCC, Inc and MC formulations

z_{tk}	break points of line segment k at period t
α_t^+	absolute increase rate of flow during period t
α_t^-	absolute decrease rate of flow during period t
β_t	multiplier of energy requirement during period t
γ_{it}^k	continuous variables in MC formulation
δ_{it}^k	continuous variables in Inc formulation
λ_{it}^k	continuous variables in CC, Log and SOS2 formulations
$\lambda_{it}^{kl}, \lambda_{it}^{kr}$	continuous variables in DCC and DLog formulations
μ	shape parameter of sigmoid function
ρ_{itk}	one if reference flow at hole i during period t is contained in line segment k and zero otherwise
ϕ_{dk}	parameters in Log formulation
ψ_{dk}	parameters in DLog formulation
ω_1, ω_2	auxiliary variables used in CDC formulation

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
AMP	Actuator Management Problem
C^2	set of twice continuously differentiable functions
CC	Convex Combination Model
CDC	Canonical Form of Difference of Convex
CLE	Corrected Linear Estimation
CLSE	Constrained Least Squares Estimation
CP	Convex Programming
CQCE	Corrected Quadratic Convex Estimation
DC	Difference of Convex
DCC	Disaggregated Convex Combination Model
DLog	Disaggregated Logarithmic Model
EG	Exact Gradient
FD	Finite Difference
GRG	Generalized Reduced Gradient Algorithm
Inc	Incremental Model
LE	Linear Estimation
LEGI	Laboratoire des Ecoulements Géophysiques et Industriels
Log	Logarithmic Model
MC	Multiple Choice Model
MILP	Mixed-Integer Linear Programming
MINLP	Mixed-Integer Nonlinear Programming
NGRG	Neural Generalized Reduced Gradient Method
NLP	Nonlinear Programming
NRG	Neural Reduced Gradient Method
NNCT	Neural Network with Cross Terms
NNRT	Neural Network with Regular Terms

PLR	Piecewise Linear Regression
QCP	Quadratically Constrained Programming
QE	Quadratic Estimation
RG	Reduced Gradient Algorithm
SOS2	Specially Ordered Sets of Type 2

1. INTRODUCTION

In most of the mathematical programming models, a given objective function is minimized subject to some given set of constraints. Depending on the structure of the relations between variables and their types, these problems can be categorized as linear programming (LP), nonlinear programming (NLP), mixed-integer linear programming (MILP) among others and appropriate solution methods can be used to solve them.

However, in some applications, although the constraints of the problem are given, the objective function does not have a closed form expression in terms of the decision variables but it can be calculated using a simulator. In Operations Research literature, such an optimization problem where the objective function is found using simulation is known as a simulation optimization problem. In this thesis, we deal with continuous problems.

Simulation optimization problems are particularly difficult since we do not know the objective function and consequently, its gradient, which is a very important piece of information for optimization algorithms. Therefore, the objective gradient should be accurately approximated so that well-known gradient-based algorithms can be applied to these problems as well.

Our approach here is to construct an estimator to replace the simulator and then, use the gradient of the fitted function as an estimate for the actual objective gradient. For this purpose, we use neural networks and in particular, multi-layer perceptrons to fit appropriate functions for the underlying simulator. Then, reduced gradient algorithm (RG) is applied to find a stationary point of the fitted function subject to the set of known constraints. We call this method Neural Reduced Gradient Method (NRG) and test it with some randomly generated problems as well as difficult benchmark instances from the literature. In order to validate the accuracy and efficiency of the method under different assumptions, several experiments are designed.

One of the possible application areas of simulation optimization is fluid mechanics and especially microfluidics, which deals with fluids in micro-scale. Similar to the miniaturized electronic devices, it is beneficial to have very small mechanical devices in some application areas such as physics, chemistry and biotechnology. However, when the scale is small, the underlying physical principles change considerably. For instance, the behaviour of micro-flows is significantly different than their normal-size counterparts.

A very important problem in this area is drag reduction. Although it is known that turbulence can be completely eliminated, this is only an existence type result and how it can be actually achieved still remains an open question. Basically, drag reduction is an attempt to reach this ultimate goal.

There are a couple of difficulties with drag reduction problem. First of all, turbulence is not a quantifiable notion. Hence, it is more suitable to use shear stress which is easily quantifiable by measurement. Then, turbulence phenomenon is governed by Navier-Stokes Equations, which are actually a system of nonlinear partial differential equations and hence, very hard to solve analytically. In practice, numerical tools such as simulators are used to approximate a solution.

Fluid mechanists dealing with drag reduction usually have an experimentation medium. This thesis is based on a collaborative work with Laboratoire des Ecoulements Géophysiques et Industriels (LEGI) and we use their experimental environment, which consists of micro-scale objects called actuators located on a membrane modelled as a 2D grid. These objects, which can blow and suck air with some magnitude, are used to reduce the shear stress between the membrane and fluid. A simulator is used to approximate shear stress values over the membrane given the air flow amounts since the relation between these two quantities are not known explicitly.

The goal is to minimize the average shear stress over the membrane at the end of the simulation by determining air flow amounts for each actuator through simulation

periods while satisfying some physical constraints. We call this problem Actuator Management Problem (AMP). Here, notice that although only the shear stress value at the end of the simulation is relevant, it is still important to determine all the flow amounts through the simulation since the system is dynamic. Therefore, we prefer to minimize the total shear stress over the membrane through the simulation. Currently, LEGI uses an *ad hoc* control algorithm and its result is used as a reference.

AMP is an exceptionally challenging simulation optimization problem since it is very large-scale with approximately five million variables and the time needed to execute a single simulation run takes almost one hour. Therefore, before applying our proposed fitting procedure with neural networks, we first use least squares estimation which is an easier curve fitting technique. Different statistical models are constructed and some constraints are added so that the fitted functions obey the underlying physical principles. The introduction of these additional constraints significantly enhances the estimation procedure. Then, AMP is solved by an extension of our proposed method.

The remaining of this thesis is structured as follows: The next chapter gives a literature survey on some important topics considered in the thesis. Chapter 3 is devoted to our proposed method, namely Neural Reduced Gradient, for constrained simulation optimization problems whereas Chapter 4 discusses AMP in detail with estimation and optimization models. In Chapter 5, the results obtained using the methods introduced in the previous two chapters are presented. Some important conclusions are drawn and possible future research directions are suggested in Chapter 6. Finally, Appendices A and B contain further discussion on piecewise linear approximation models and optimal solutions of the test problems, respectively.

2. LITERATURE SURVEY

2.1. Simulation Optimization

Simulation optimization is a relatively new and not entirely explored research area in operations research. It arises in applications such as system dynamic simulation, chemical reaction, fluid mechanic experimentation etc., in which a function does not have a closed form but can only be estimated using a “black box” simulation subroutine. [1] and [2] are two excellent review papers on simulation optimization which provide many other examples and comprehensive surveys.

There are two major difficulties in simulation optimization mentioned in [1]: First of all, many simulation subroutines considered in this area are stochastic in nature. Therefore, several replications are needed to compare two different designs in order to be able to identify one of them superior only by statistical terms. The second difficulty is knowing almost nothing about the simulation subroutine. This causes the algorithms to be too general so that the details of a specific problem cannot be taken into consideration.

In [2], simulation optimization problems are categorized into two groups: continuous and discrete. Continuous problems are usually solved by gradient-based methods. For instance, stochastic approximation methods try to estimate the derivative of the simulated function and use known search algorithms, whereas sample path optimization methods transform the problem into its deterministic equivalent and use deterministic optimization methods. On the other hand, discrete problems are generally solved by heuristics such as random search, simulated annealing and evolutionary algorithms. If the number of alternatives is restricted, then ranking and selection or multiple comparison methods can be applied.

Besides the methods mentioned above, there is a tendency to use metamodels to

solve continuous simulation optimization problems in order to decrease the number of simulation runs. In [3], metamodels are categorized into two groups: global and local. Local metamodels are typically first or second order polynomials and updated during the optimization whereas global metamodels are constructed only once using methods such as neural networks, splines etc. As a general methodology, first, a metamodel is formed and then, its quality is validated. Finally, the metamodel is optimized using an appropriate solution procedure.

In [4], a metamodel called Golden Region is utilized. It is a search-based optimization algorithm which iteratively generates feasible subregions and chooses a promising candidate solution in that subregion. Selection of the subregion and new point inside it are governed by a scoring mechanism. There are three criteria in selecting a subregion: space score, metamodel score and quality score. In each iteration, depending on which score criterion is dominant, a subregion is obtained from each region. Then, a new point in that subregion is selected accordingly and simulated. This method has a large number of parameters to be finely tuned.

Another work which uses a metamodel is [5]. In this paper, evolutionary algorithms are used with a metamodel called Taylor Kriging. The fundamental idea is to temporarily replace a simulation model with known fitted values to evaluate simulation inputs such that the total number of simulation runs is reduced. Because of its superior interpolation property, Kriging metamodel is used and its accuracy is further enhanced by Taylor expansion. The algorithm developed is tested using some benchmark functions and its effectiveness is justified by comparing it with other known algorithms such as simulated annealing, evolutionary strategy and particle swarm optimization.

Heuristics are also commonly utilized in solving simulation optimization problems. For instance, in [6], a hybrid genetic algorithm-neural network strategy is used to determine four critical parameters of a pressure vessel. Although the objective function is explicitly given, it is only used to generate random patterns to train the neural network which predicts the actual objective. Then, a genetic algorithm is applied as

the optimization procedure and compared against methods appeared in the literature such as nonlinear integer programming, evolution strategy and penalty approach.

In [7], a flow solver, which is used in numerical computational fluid dynamics, is integrated with a hybrid optimization procedure to determine the shape of an airfoil in 2D and a wing in 3D. Genetic algorithms are chosen to determine promising regions and then, conjugate gradient method is utilized for the final local search. Additionally, neural networks are used to approximate the solver. This is necessary to limit the number of the flow solver runs since they are computationally expensive. [8] uses exactly the same procedures to determine the best yacht fin keel geometry using 3D simulations. The basic difference is that there are two objectives, maximization of hydrodynamic lift and minimization of drag in the latter study, and a compromise solution is obtained.

Simulation programs typically have multiple, stochastic outputs and [9] deals with optimization in such cases where one of the outputs is selected as the actual objective whereas the others are considered as constraints. The statistical tools developed are tested using a simple toy problem and a real inventory control problem. This study only introduces statistical tests which can be used to find out whether a feasible solution satisfies Karush-Kuhn-Tucker first order necessary conditions or not and leaves the open question that how these tests can be incorporated with an optimization procedure.

There are different approaches in solving discrete simulation optimization problems as well. In [10], a strategy is developed to balance three important features of a simulation optimization problem: exploration, exploitation and estimation. Exploration refers to global search mechanism of the algorithm, which tries to find promising subregions of the solution space. Once such subregions are identified, exploitation comes into picture and a local search is applied on each subregion. After deciding the most promising alternatives, enhanced estimation of the objective function is done. Four different algorithms are proposed to balance these aforementioned steps in such a way that the solution method would need as few simulation runs as possible since simulation is assumed to be costly.

2.2. Piecewise Linear Approximation

Piecewise linear functions can be used to approximate nonlinear functions with their accuracy determined by the size of line segments. If the objective is minimization (maximization) and the nonlinear function is convex (concave), then the approximation is straightforward. However, if this is not the case, then some additional variables and constraints are introduced to correctly represent the function as in [11].

Depending on the number of additional variables added to the model, one can come up with different representations of the same problem using its piecewise linear approximation. [12] is a recent, detailed work about such different models. Seven different formulations are given and their computational performances are compared. The polyhedral theory behind piecewise linear approximation models is a very rich research area, as well. For instance, [11] and [13] compare the underlying polyhedra of those models and their LP relaxations.

In [14], a minimization type knapsack problem with a demand constraint and a piecewise linear cost function is considered. This kind of a setting can arise in areas such as production and capacity planning, financial modelling and health-care applications. The formulations in this study are quite general in the sense that they can handle discontinuities in the objective function and different settings of the break points. Heuristics and approximation algorithms are developed for practical and theoretical purposes, respectively.

2.3. Curve Fitting

Curve fitting has always been an important topic in many disciplines such as statistics, engineering and optimization. Its aim is to fit a function with some desired property to a set of data points such that a performance criterion is optimized. Most of the time, the performance criterion is specific to application and in order to obtain satisfactory results, an adequate objective function should be selected as mentioned

in [15]. The sum of square of errors, the sum of absolute deviations or the maximum of absolute deviations can be minimized accordingly.

Multiple Linear Regression is a practical way of function estimation provided that the input data satisfy the assumptions given in [16]: linearity, homoscedasticity or common variance, independence and normality of residuals. Then, least squares method can be utilized to find parameter values and the fitted function is constructed.

Within optimization terminology, fitting problems are convex norm minimization problems as noted in [17]. Although they are unconstrained most of the time, it is possible to add constraints in order to impose some features to the fitted function. [18] and [19] discuss the general statistical properties of the estimators under constraints. An important restriction on the fitted function is convexity, which is discussed in [20] among many others.

As a final note, least squares estimation can be extended to the case where variables are random. This extension is called as least mean squares estimation and the parameters can be determined by taking advantage of the orthogonality conditions in [21], which suffices to solve Yule-Walker Equations. It is also proven that as the stochastic counterpart of its deterministic version, least mean square estimation is equivalent to least squares estimation.

2.4. Microfluidics

Recently, microfluidics has become an emerging area of research. As noted in [22], it has some important applications in physics, chemistry, microbiology, control systems and heat management, energy generation and display technology. Nowadays, the main emphasis is on medical applications of microfluidics. According to [23], microfluidic systems allow sophisticated medical tools to be miniaturized. The so-called "lab-on-a-chip" technology is postulated to have a potential to create portable medical diagnostic systems which can process small volume of complex fluids quickly and efficiently with-

out an expert.

An important phenomenon regarding fluids is mixing. [22] provides a broad review on the physics behind microfluidics and, in particular, on mixing. It is interesting to deal with mixing in micro-flows since the physical principles change drastically. For instance, inertial effects are small enough to be neglected compared to viscous forces in mass transport in microfluidic devices. Therefore, being the ratio of inertial forces to viscous forces, Reynolds Number, which is an important dimensionless quantity in fluid mechanics, becomes very small (between 10^{-6} and 10) and this fact makes flow to be laminar. Since inertial forces are responsible for nonlinearities leading to instabilities and turbulence, mixing becomes unacceptably slow in micro-flows. Therefore, it is reasonable to search for methods which can enhance active mixing in such environments.

There are some interesting problems in microfluidics which are solved by optimization procedures. In [24], a biochip design problem is modelled as a 3D bin-packing problem. Also, an algorithm is proposed for droplet routing over the microfluidic devices, which prevents droplets to mix each other. In [25], design of pin-constrained biochips are considered, in which the problem is formulated as a minimal clique-partitioning problem and solved by a two-phase optimization method.

In [26], testing procedures of the digital microfluidic biochips are considered since their applications such as biodefence or clinical diagnostics are critical by their nature. Different mathematical models are formulated to minimize completion time under non-interference constraints. These models are variations of Vehicle Routing Problem with some additional considerations and three heuristic algorithms are designed since the resulting problem is proven to be NP-hard.

Although the application areas of microfluidics and related optimization problems are very rich, we concentrate on actuators and drag reduction in the scope of this thesis.

Actuators are used to control turbulence, reduce drag and enhance mixing in order to obtain operational savings in many different application areas. As noted in [27], finding the best geometry of an actuator is an important design problem since an optimal design can increase the performance of actuators significantly. Therefore, four important parameters, orifice width, orifice height, cavity height and frequency, are considered. First, these parameters are optimized as single-variable problems to maximize momentum flux using simulation results. Then, all of the parameters are taken into account since they actually interact with each other.

Although many studies in drag reduction literature depend on *ad hoc* control principles, several other methods are developed as well. For instance, [28] and [29] take advantage of suboptimal control by constructing different cost functionals. However, in the latter study, it is shown that this approach does not always outperform *ad hoc* control.

[30] is a survey on control of turbulent boundary layers. Although several control mechanisms such as opposition control, adaptive inverse model and adjoint-based suboptimal control are applied to reduce drag for turbulent channels with blowing and suction as the control input, adaptive inverse model is the only practical method among them. This method, also thoroughly discussed in [31], takes advantage of neural networks to find the correlation between wall actuation and skin-friction drag since this relation is complex due to Navier-Stokes Equations. Also, a simple linear control scheme is proposed.

In [32], a genetic algorithm-based optimization is applied to reduce wall skin friction using local blowing/suction at the wall as the control input. The blowing/suction is assumed to be a function of wall shear stress values and the seven parameters in that particular function are optimized in order to obtain a control scheme which reduces drag considerably. Genetic algorithms are also used in [33]. This time, the voltage of each actuator is assumed to be a function of shear stress fluctuations and parameters are optimized such that mean wall shear stress is minimized.

3. NEURAL REDUCED GRADIENT METHOD

Consider the following optimization problem where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{l} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^n$ are given and the unknown vector $\mathbf{x} \in \mathbb{R}^n$ is to be determined:

$$\min F(\mathbf{x}) \quad (3.1)$$

$$\text{s.t. } \mathbf{Ax} = \mathbf{b} \quad (3.2)$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \quad (3.3)$$

If the objective function F has a closed form in terms of $\mathbf{x}$, then this problem is, in general, an NLP and a stationary point can be found using RG provided that F is differentiable and equality system is nondegenerate [34].

Now, assume that F does not have a closed form but can be calculated using a simulation subroutine given the vector $\mathbf{x}$. In this case, the above problem becomes a simulation optimization problem and RG cannot be used directly. However, in this chapter, we will enhance RG so that it can be applied to simulation optimization problems as well. The quality of the proposed method is tested with some test instances in Chapter 5.

3.1. Characteristics of Simulators and Appropriate Solution Methods

There may be different cases with respect to the characteristic of the simulator mentioned. First of all, suppose that the running time of the simulator is short or equivalently, the simulator is inexpensive. Assume further that the simulator is deterministic. In that case, although F is not known explicitly, its partial derivatives can be approximated using finite difference method as follows:

$$\frac{\partial F}{\partial x_j} \approx \frac{F(\mathbf{x} + \epsilon \mathbf{e}_j) - F(\mathbf{x})}{\epsilon} \quad j = 1, \dots, n \quad (3.4)$$

Here, ϵ is an appropriately small positive number and $\mathbf{e}_j$ is the j^{th} unit vector. Observe that if $\epsilon \rightarrow 0$, this quantity gives the correct partial derivative.

Now, assume that the simulator is again inexpensive but stochastic. In that case, Equation 3.4 cannot give good estimates for the partial derivatives due to randomness. Therefore, we have to replicate the simulations with the same input vectors and then, take the average in order to obtain better estimates as suggested below:

$$\widehat{\frac{\partial F}{\partial x_j}} \approx \frac{\overline{F(\mathbf{x} + \epsilon \mathbf{e}_j)} - \overline{F(\mathbf{x})}}{\epsilon} \quad j = 1, \dots, n \quad (3.5)$$

The choice of the number of replications should be done with care. If it is small, then the estimation might be unreliable. On the other hand, if it is large, then time spent for the simulations may increase considerably.

In short, if the simulator is inexpensive, then the methods mentioned in this section can be applied to estimate the objective gradient. However, most of the time, simulators are used to come up with approximate solutions to complex systems. Therefore, they tend to be expensive by construction. In that case, these methods seem to fail due to the need for an extensive amount of simulation time. Consequently, a new method should be used to overcome this difficulty. So, we propose neural reduced gradient method (NRG) which essentially constructs a simulator for the actual simulator using neural networks, treats it as the objective function and uses RG to find a stationary point.

3.2. Use of Neural Networks

Neural networks and in particular, multilayer perceptrons, are reliable and flexible tools when it comes to function estimation as stated in [35]. They can be efficiently used to explain complex and nonlinear relations provided that they are carefully trained. Also, it is proven in [36] that a neural network with a single hidden layer is enough to

fit any function, which is actually what we are after since this will allow us to estimate the function F and RG can be applied with slight modifications.

3.2.1. Construction of the Neural Network

Suppose that a set of feasible patterns with respect to Constraints 3.2 and 3.3 are given as $(\mathbf{x}^p, O^p) \in \mathbb{R}^{n+1}$ where $O^p = F(\mathbf{x}^p)$. Then, we construct a neural network with n input neurons and a single output neuron. A single hidden layer with H neurons are also included where H is found using a constructive approach suggested in [37], namely, we start with a small network and enlarge it one by one as long as the addition of another unit decreases the test error.

The input neurons are normalized in order to smooth out the possible scale differences using

$$\hat{x}_j^p = \frac{x_j^p - \bar{x}_j}{s_j} \quad (3.6)$$

for every variable j and pattern p where $\bar{x}_j$ and s_j are the estimators of the mean and standard deviation of the patterns with respect to the j^{th} input neuron, respectively.

Also, the output neuron is normalized using

$$\hat{O}^p = \frac{O^p - \bar{O}}{s} \quad (3.7)$$

for every pattern p where $\bar{O}$ and s are the estimators of the mean and standard deviation of the output neuron, respectively.

We can now use backpropagation algorithm, appeared in [37] and many others, on the network with normalized inputs and outputs to estimate function F , denoted by O . As a common practice, 75% of the patterns are used as the training set and the rest is allocated as the test set. Backpropagation algorithm is basically a gradient descent

algorithm with inaccurate line search. The aim is to find the best connection weights from the input unit j to the hidden unit h , denoted by w_{hj} , and from the hidden unit h to the output unit, denoted by W_h , such that the test error is minimized.

Parameters of this algorithm such as the initial learning rate, the decay rate of the learning rate, the number of epochs etc. should be chosen carefully since they considerably affect its performance. Also, early termination, which prevents overlearning, seems an important feature to be taken advantage of. Furthermore, we use online learning, i.e., the connection weights are updated after the input of each pattern.

The activation function between input and hidden units, denoted by g , is taken as the sigmoid function with shape parameter μ given as

$$g(x) = \frac{1}{1 + e^{-2\mu x}}. \quad (3.8)$$

Thanks to the special structure of the sigmoid function, once $g(x)$ is known, g' can be easily calculated as

$$g'(x) = 2\mu g(x)(1 - g(x)). \quad (3.9)$$

On the other hand, the activation function between hidden and output units, denoted by G , is taken as the identity function.

3.2.2. Estimation of the Objective Gradient

Suppose that backpropagation algorithm with appropriate parameters is applied to the neural network constructed above and the best connection weights are determined. Then, we can estimate the value of the objective function at a point $\mathbf{x}$ as

$$O(\mathbf{x}) = \bar{O} + s \sum_{h=0}^H W_h V_h \quad (3.10)$$

where

$$V_h = g \left(\sum_{j=0}^n w_{hj} \hat{x}_j \right) \quad h = 1, \dots, H. \quad (3.11)$$

Here, $\hat{x}_0 = -1$ and $V_0 = -1$. Note that Equation 3.10 comes from the backtransformation of Equation 3.7.

Therefore, even if the actual simulator does not have an explicit formula in terms of $\mathbf{x}$, we are now able to estimate it using the neural network. Now, it suffices to approximate the objective gradient so that we can use RG to solve simulation optimization problem. There are two alternatives given as follows:

3.2.2.1. Finite Difference. The first method that can be used to approximate the objective gradient is to use finite difference again as below:

$$\frac{\partial O}{\partial x_j} \approx \frac{O(\mathbf{x} + \epsilon \mathbf{e}_j) - O(\mathbf{x})}{\epsilon} \quad j = 1, \dots, n. \quad (3.12)$$

3.2.2.2. Exact Gradient of the Perceptron. Since Equation 3.10 gives a differentiable, closed form approximation for the objective function, we can find its gradient analytically using chain rule in a successive way. For every input unit $j = 1, \dots, n$, we have

$$\frac{\partial O}{\partial x_j} = \frac{\partial O}{\partial \hat{O}} \frac{\partial \hat{O}}{\partial \hat{x}_j} \frac{\partial \hat{x}_j}{\partial x_j} \quad (3.13)$$

$$= s \left(\sum_{h=1}^H \frac{\partial \hat{O}}{\partial V_h} \frac{\partial V_h}{\partial \hat{x}_j} \right) \frac{1}{s_j} \quad (3.14)$$

$$= 2\mu \frac{s}{s_j} \sum_{h=1}^H W_h w_{hj} V_h (1 - V_h) \quad (3.15)$$

Note that once all V_h values are determined, both of these methods need $\mathcal{O}(nH)$

operations to approximate the gradient.

3.3. Extensions of Neural Reduced Gradient

So far, we only consider a neural network whose input units are exactly the variables of the problem. Let us call this network as neural network with regular terms (NNRT). Although the result of test instances in Section 5.1 seems promising, we can further enhance NRG by constructing a larger network which also includes cross terms of the original neurons as extra neurons. If originally n input units are used, then there are $\binom{n}{2} = \frac{n(n-1)}{2}$ additional units coming from cross terms. We will refer to this network as neural network with cross terms (NNCT) from now on.

The introduction of these cross terms can be especially helpful if the estimated function is highly nonlinear. Also, however, the size of the network increases considerably which results in a longer training process. Therefore, as in many applications, there is a trade-off between accuracy and speed.

Another important extension of NRG is for a problem with nonlinear constraints as opposed to linear constraint set 3.2 considered in this chapter. However, this is straightforward: We can use generalized reduced gradient algorithm (GRG) [34] instead of RG and hence, apply the same procedures for approximation of the objective gradient. The main difficulty of GRG is due to the correction step needed from time to time once the point generated in some iteration is outside the feasible region and therefore, should be projected back to the constraint space.

This extension, called neural generalized reduced gradient method (NGRG), is applied to AMP, discussed in depth in Chapter 4, and the results can be seen from Section 5.2.

4. DRAG REDUCTION

In this chapter, we discuss an application of simulation optimization in drag reduction, which aims at controlling the turbulence by a practical way. Despite the fact that turbulence is known to be eliminated by the correct procedure, the exact method has not yet been figured out. One of the methods used in practice is to use actuators located on the membrane to reduce the shear stress between the membrane and fluid. The main goal here is to find a schedule for the actuators, which can blow and suck air, so that the total shear stress over the membrane through a planning horizon is minimized subject to the constraints coming from the physical principles.

The experimental environment is developed by LEGI and it is a discretization of the membrane as a 128×256 grid denoted by the index set $\mathbf{I}$ with cardinality $I = 32768$. There is an actuator at every point¹ i whose coordinates are given as (X^i, Y^i) . Initially, a single set of input data based on the simulation run using LEGI's *ad hoc* control scheme is provided. It consists of flow velocity values of each point i on the grid and the resulting shear stress values for $T = 151$ time periods.

Figure 4.1 is the output of the actual simulator and shows the evolution of average shear stress over the grid through simulation. Note that each time period $t \in \mathbf{T}$ consists of 10 iterations and our time period counter is initialized at iteration 400,000.

Normally, the shear stress at some point j is a function of all the flow values on the grid. However, in order to simplify the problem, we assume that the shear stress at some point j is a function of flow values on a neighbourhood $\mathbf{N}_j$ defined as

$$\mathbf{N}_j = \{i \in \mathbf{I} : \|i - j\|_2 \leq r\}. \quad (4.1)$$

Here, r is a predetermined positive integer value for radius of neighbourhood which

¹We use the terms *point* and *hole* interchangeably throughout the thesis.

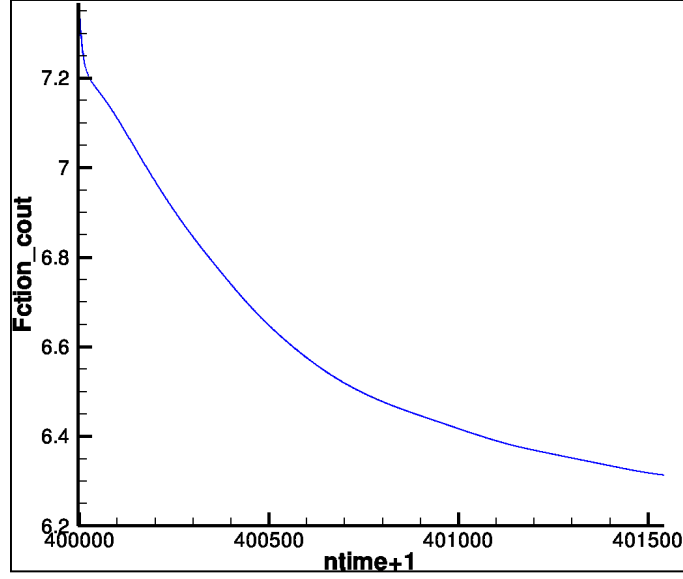


Figure 4.1. Average shear stress over the grid obtained by the reference LEGI simulation.

is taken to be 10 units assuming that any two neighbouring points are separated by one unit of distance. Note that since we only use the Euclidean metric in the thesis, subscript is omitted hereafter.

The choice of this neighbourhood structure has some important consequences which can be better explained by the following two definitions:

Definition 4.1. *Points i and j are said to have the same neighbourhood structure if there exist two integers $\tilde{x}$ and $\tilde{y}$ such that the point $i' \in \mathbf{I}$ with coordinates $(X^{i'}, Y^{i'}) = (X^j + \tilde{x}, Y^j + \tilde{y})$ has the same neighbours as point i , i.e., $\mathbf{N}_i = \mathbf{N}_{i'}$.*

Definition 4.2. *A region is the collection of points on the grid with the same neighbourhood structure.*

Under these definitions, observe that being in the same region is an equivalence relation [38]. Also, if r is a positive integer, then there are precisely $M = (2r + 1)^2$ many distinct regions or equivalence classes on the grid. Figure 4.2 is an example of a 4×4 grid with $r = 1$ which is used to clarify the above definitions.

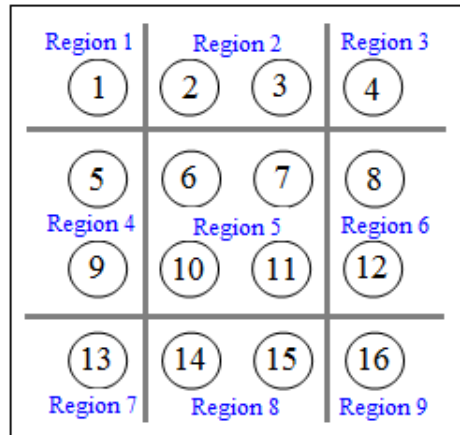


Figure 4.2. Regions for a 4×4 grid with $r = 1$.

As we can see from the above figure, there are nine distinct regions with variable sizes. For instance, Regions 1, 3, 7 and 9 have only one element whereas Regions 2, 4, 6 and 8 have two. Region 5 has the greatest number of elements with four. Due to above definitions, the region at the centre of the grid has always the greatest number of holes. This simple example is used later again in order to visualize the regions and understand how they simplify our problems.

Recall that LEGI's experimentation medium is modelled as a 128×256 grid and we take $r = 10$. Consequently, there are 441 regions. Many of these regions have different sizes which are summarized in the following table:

Table 4.1. Region counts and their sizes.

Number of regions	Number of holes in each region	Size of the region
400	1	1×1
20	108	108×1
20	236	1×236
1	25488	108×236

Holes in the 108×236 region have the largest number of neighbours as $N_m = 317$. As we go from the centre of the grid to perimeters, the number of neighbours of a hole

decreases. Figure 4.3 illustrates a part the actual grid and useful to visualize the defined neighbourhood structure.

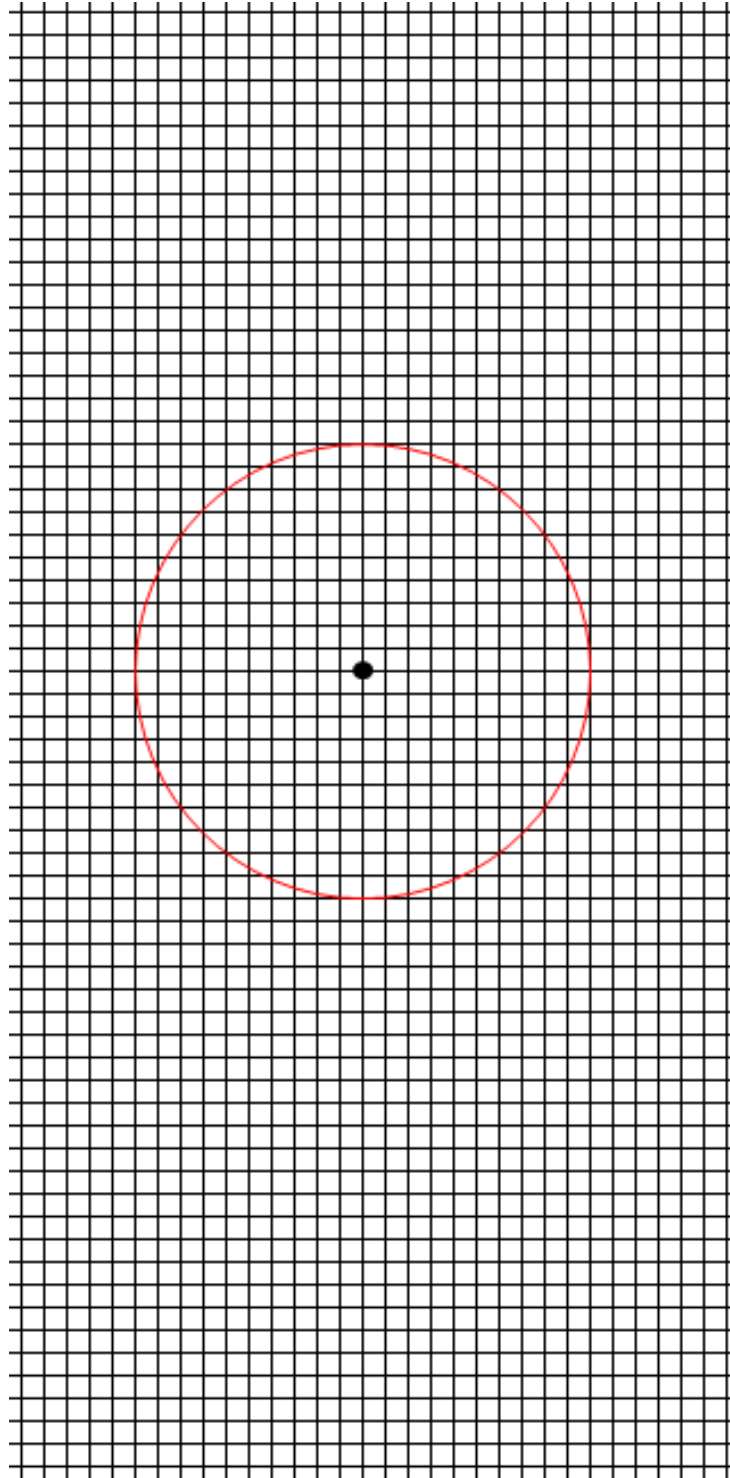


Figure 4.3. A part of the actual grid. A point and its neighbours, which lie in the red circle, are depicted.

After deciding the neighbourhood structure, we need to determine the relation between the flow amounts and the shear stress. Note that this relation is very difficult to identify, if not impossible. In fact, the need of simulation arises from this fact. However, in order to be able to solve the optimization problem given in the next section, we need to come up with a good estimation. Therefore, we try different alternatives which are presented in Sections 4.2 and 4.3. The common characteristic of these alternatives is that first, we propose a functional form with some parameters and then, these parameters are found using an appropriate method. Finally, the accuracy of the estimation is determined by calculating R^2 and adjusted R^2 values.

4.1. Optimization Problem

In this section, we present the mathematical programming formulation of AMP whose indices and parameters are given in Table 4.2:

Table 4.2. Parameters of the optimization problem.

Parameter	Definition
I	number of holes on the membrane, $i, j \in \mathbf{I} = \{1, \dots, I\}$
$\mathbf{N}_j$	set of neighbours of hole j
T	number of periods, $t \in \mathbf{T} = \{1, \dots, T\}$
E	total available energy
l_{it}	minimum air flow through hole i during period t
u_{it}	maximum air flow through hole i during period t

We have a single set of decision variables, denoted by f_{it} , defined as air flow through hole i during period t . Note that these flow variables are unrestricted in sign. When $f_{it} > 0$, air is blown whereas $f_{it} < 0$ implies that air is sucked. Finally, there is no flow activity if $f_{it} = 0$.

Another important function in AMP formulation is the amount of energy spent

at hole i during period t due to f_{it} units of air flow denoted by $e_{it}(f_{it})$.

The objective function of AMP does not have a closed form in terms of the decision variables since it is the result of a simulation subroutine. Let SS_{it} be the shear stress at hole i during period t . The aim in AMP is to minimize the sum of shear stress values over a planning horizon.

Together with the above definitions, AMP is given as follows:

$$\min \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} SS_{it} \quad (4.2)$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} e_{it}(f_{it}) \leq E \quad (4.3)$$

$$\sum_{i \in \mathbf{I}} f_{it} = 0 \quad t \in \mathbf{T} \quad (4.4)$$

$$l_{it} \leq f_{it} \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.5)$$

$$f_{it} \in \mathbb{R} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.6)$$

In the above formulation, the objective function 4.2 minimizes the total shear stress exerted at each point throughout the planning horizon. Constraint 4.3 sets an upper bound on total energy requirement as E . Constraints 4.4 guarantee that net flow in every period t is zero and Constraints 4.5 put an upper and lower bound on the air flow through every hole i in every period t . Finally, Constraint set 4.6 allow flow variables to assume real values. Notice that the objective function does not have decision variables explicitly since it comes from the simulator.

Note that depending on the structure of the estimation and the resulting optimization problem, we occasionally split the energy constraint over the time periods as follows:

$$\sum_{i \in \mathbf{I}} e_{it}(f_{it}) \leq E_t \quad t \in \mathbf{T} \quad (4.7)$$

Here, E_t is the available energy for time period t . Also, it is reasonable to take energy function as $e_{it}(f_{it}) = f_{it}^2$ [39].

Through some experiments, we deduce that if radical changes are allowed in terms of lower and upper bounds on flow variables, then the result of simulation tends to diverge from the reference LEGI output. Therefore, in the experimentation phase, we force f_{it} variables to have the same sign as in the reference LEGI simulation. This is guaranteed by selecting lower and upper bounds as follows where $\alpha_t^- < 1$ and $\alpha_t^+ > -1$ for $t \in \mathbf{T}$:

$$l_{it} = \begin{cases} (1 - \alpha_t^-)f_{it} & \text{if } f_{it} > 0 \\ (1 + \alpha_t^+)f_{it} & \text{if } f_{it} < 0 \end{cases} \quad \text{and} \quad u_{it} = \begin{cases} (1 + \alpha_t^+)f_{it} & \text{if } f_{it} > 0 \\ (1 - \alpha_t^-)f_{it} & \text{if } f_{it} < 0 \end{cases} \quad (4.8)$$

Hence, although f_{it} variables seem to be unrestricted in sign, they are actually either strictly positive or strictly negative due the selection of parameters l_{it} and u_{it} .

Observe that AMP is an NLP, which is difficult to solve optimally unless it is a convex programming (CP) problem. The fact that SS_{it} having no closed form increases the difficulty of this problem even more. If those functions were explicitly known, then algorithms used to solve NLP problems would have been applied. Therefore, before trying to solve AMP, somehow we need to determine SS_{it} functions in a reasonable way. In the next two sections, we use two basic methodologies, namely Constrained Least Squares Estimation (CLSE) and Neural Networks, to approximate them.

4.2. Constrained Least Squares Estimation

Least Square Estimation is a straightforward way of function estimation. However, since it is too simple and the turbulence is a very complex physical phenomenon, some restrictions should be included so that the fitted functions accurately represent the underlying physical phenomenon.

First of all, we define the function $s_{ijt}(f_{it})$ as the shear stress effect of hole i exerted at hole j during period t due to f_{it} units of air flow. Then, we make two critical assumptions: Firstly, the shear stress exerted at a point is expressed as the sum of individual effects of points in its neighbourhood. In other words, the shear stress effect functions are separable and, hence the cross effects are neglected. Secondly, the shear stress effect function at a specific time period is assumed to be a function of flow values at that particular period only which results in a static estimation to a dynamic system. We try to remedy the downside of the first assumption in Section 4.3 where we use Neural Networks.

Under these assumptions, the shear stress estimate at hole j during period t , say $\widehat{SS}_{jt}$, can be calculated as

$$\widehat{SS}_{jt} = a_t + \sum_{i \in \mathbf{N}_j} s_{ijt}(f_{it}) \quad (4.9)$$

where a_t is the shear stress exerted at any hole during period t if there is no flow activity for any hole over the grid, which is another important piece of information and can be seen as the intercept of the statistical model. Note that we assume this value to be the same for all holes over the grid for a given time period t .

Then, the total shear stress exerted at each hole for a specific time period t is

$$\sum_{j \in \mathbf{I}} \widehat{SS}_{jt} = \sum_{j \in \mathbf{I}} \left(a_t + \sum_{i \in \mathbf{N}_j} s_{ijt}(f_{it}) \right) \quad (4.10)$$

$$= \sum_{i \in \mathbf{I}} a_t + \sum_{j \in \mathbf{I}} \sum_{i \in \mathbf{N}_j} s_{ijt}(f_{it}) \quad (4.11)$$

$$= \sum_{i \in \mathbf{I}} a_t + \sum_{i \in \mathbf{I}} \sum_{j \in \mathbf{N}_i} s_{ijt}(f_{it}) \quad \text{since } i \in \mathbf{N}_j \iff j \in \mathbf{N}_i \quad (4.12)$$

$$= \sum_{i \in \mathbf{I}} a_t + \sum_{i \in \mathbf{I}} S_{it}(f_{it}) \quad (4.13)$$

$$= \sum_{i \in \mathbf{I}} [a_t + S_{it}(f_{it})] \quad (4.14)$$

where

$$S_{it}(f_{it}) = \sum_{j \in \mathbf{N}_i} s_{ijt}(f_{it}). \quad (4.15)$$

Observe that S_{it} is the total shear stress effect function of hole i during period t and can be easily calculated once all s_{ijt} functions are known. Throughout this section, S_{it} functions are used instead of s_{ijt} in order to express the optimization models in a more compact way.

Finally, summing Equation 4.14 over the periods give the objective function of AMP as follows:

$$\sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} [a_t + S_{it}(f_{it})] \quad (4.16)$$

In the remaining of this section, f_{it} and SS_{it} denote the flow and shear stress amounts at hole i during period t , respectively, which come from the real data of LEGI. $\widehat{SS}_{it}$ denotes the shear stress estimate of hole i during period t , which is determined by some appropriate parameters a_{ijt} . The aim is to minimize the sum of squared errors, namely,

$$\sum_{j \in \mathbf{I}} (\widehat{SS}_{jt} - SS_{jt})^2 \quad (4.17)$$

or in matrix notation

$$\|\widehat{\mathbf{SS}}_{\mathbf{t}} - \mathbf{SS}_{\mathbf{t}}\|^2 \quad (4.18)$$

for every $t \in \mathbf{T}$ where $\widehat{\mathbf{SS}}_{\mathbf{t}} = \mathbf{F}_{\mathbf{t}} \mathbf{a}_{\mathbf{t}}$ and $\mathbf{F}_{\mathbf{t}}$ is the matrix of flow amounts.

If no further restrictions are imposed on the shear stress functions, then this

unconstrained optimization problem can be efficiently solved by QR or SVD decompositions as suggested in [40]. However, since square matrices of size $I \times I$ are too large, we solve the following equivalent system of equalities in order to find the parameter vector $\mathbf{a}_t$:

$$\mathbf{F}_t^T \widehat{\mathbf{S}} \mathbf{S}_t = (\mathbf{F}_t^T \mathbf{F}_t) \mathbf{a}_t \quad (4.19)$$

Provided that the columns of $\mathbf{F}_t$ are linearly independent, a unique solution exists and this turns out to be the case for our problems. Therefore, Equation 4.19 is solved by LU decomposition using MathNet Numerics Package [41]. However, if some restrictions such as convexity or continuity are forced, then the problem becomes a constrained least squares problem and solved by CPLEX 12.2 [42]. Note that CPLEX 12.2 solves such quadratic programming problems using Barrier Optimization.

The estimations in the next sections are carried out for each time period t independently and parameters denoted by a_{ijt} are the same for all (i, j) pairs if their orientation with respect to each other is the same. Therefore, estimation problems determine a multiple of N_m many parameters.

Recall that the assumption mentioned above is necessary in order to extract as much information as possible from a single data set. Also, due to this assumption, s_{ijt} functions become the same for all (i, j) pairs if their orientation is same. For instance, in Figure 4.2, $s_{15t} \equiv s_{37t}$ and $s_{21t} \equiv s_{87t}$ for every $t \in \mathbf{T}$. This fact enables us to generate different data points for the estimation models using the same simulation data set.

Unless otherwise stated, all the codes are written in Visual Studio 2010 **C#** compiler and tests are carried out on a computer with Microsoft Windows Server 2003 x64 Edition operating system and Intel Xeon CPU X5460 3.16 GHz processor with 16.0 GB RAM.

4.2.1. Linear Estimation

Since there are two modes of air activity as blowing and suction, it is convenient to distinguish them in the statistical model. This simplest model is called Linear Estimation (LE) and it is given as below:

$$\widehat{SS}_{jt} = a_t + \sum_{i \in \mathcal{N}_j} a_{ijt}^{+1} \max\{0, f_{it}\} + \sum_{i \in \mathcal{N}_j} a_{ijt}^{-1} \min\{0, f_{it}\} \quad (4.20)$$

Figure 4.4 shows the evolution of adjusted R^2 values.

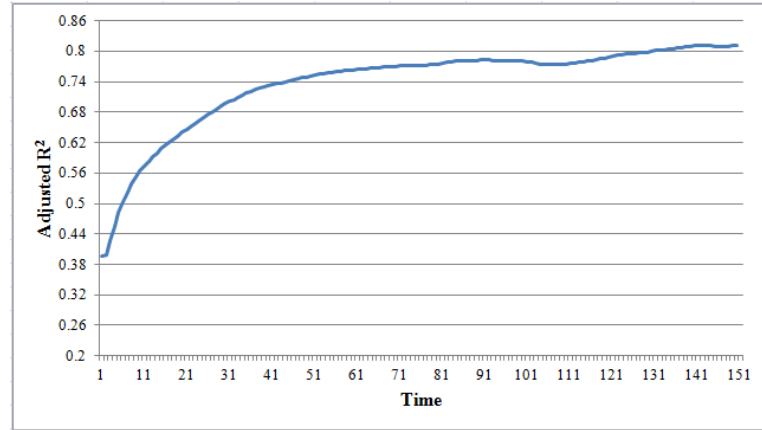


Figure 4.4. Adjusted R^2 values for LE.

4.2.2. Quadratic Estimation

The accuracy of the statistical model in Section 4.2.1 can be improved by introducing quadratic terms. The resulting statistical model called Quadratic Estimation (QE) becomes as below:

$$\begin{aligned} \widehat{SS}_{jt} = a_t &+ \sum_{i \in \mathcal{N}_j} a_{ijt}^{+1} \max\{0, f_{it}\} + \sum_{i \in \mathcal{N}_j} a_{ijt}^{-1} \min\{0, f_{it}\} \\ &+ \sum_{i \in \mathcal{N}_j} a_{ijt}^{+2} \max\{0, f_{it}\}^2 + \sum_{i \in \mathcal{N}_j} a_{ijt}^{-2} \min\{0, f_{it}\}^2 \end{aligned} \quad (4.21)$$

Figure 4.5 shows the evolution of adjusted R^2 values.

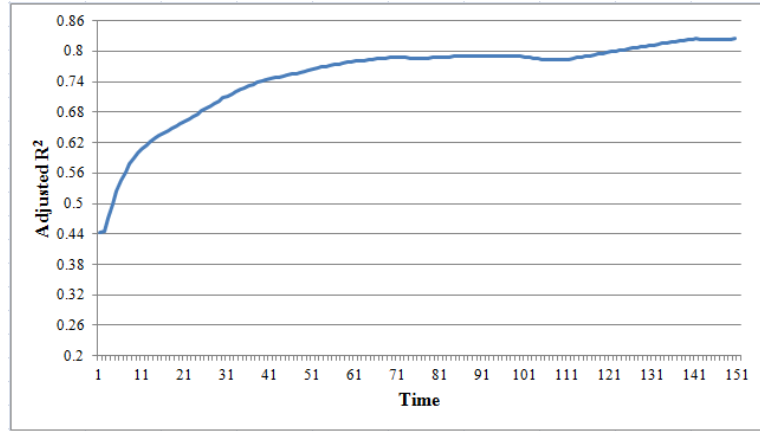


Figure 4.5. Adjusted R^2 values for QE.

4.2.3. Corrected Linear Estimation

The statistical model in Section 4.2.1 yields total shear stress effect functions similar to those given in Figure 4.6.

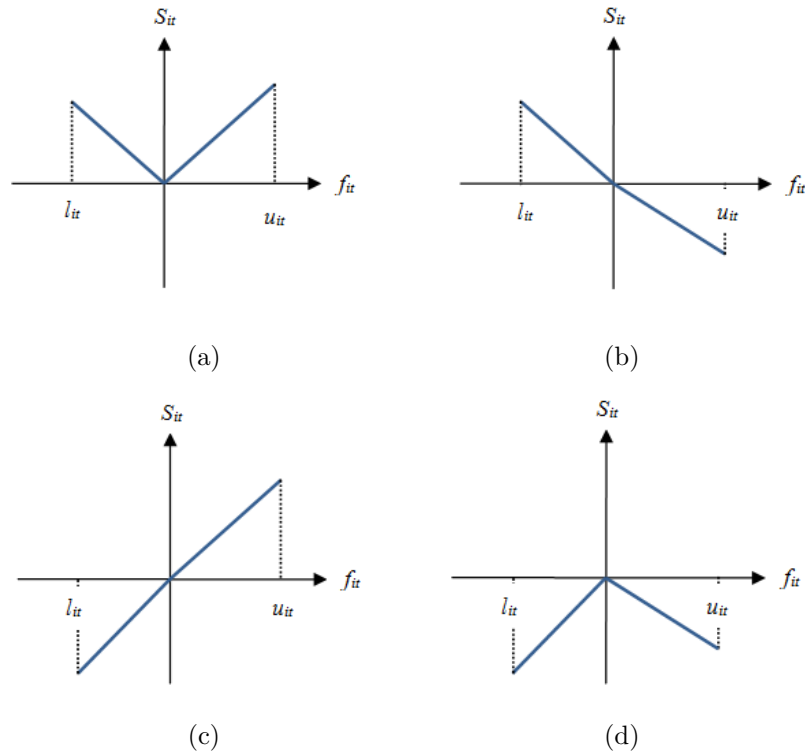


Figure 4.6. Sample S_{it} functions in (a), (b), (c) and (d) obtained using LE.

Among these four alternatives, only the one in Figure 4.6d represents the physical phenomenon the best since it is increasing for negative flow values and decreasing for positive flow values and hence, the shear stress effect decreases as the associated flow becomes larger in absolute terms. Therefore, the following constraints are added to the estimation called Corrected Linear Estimation (CLE) so that every total shear stress effect function obeys these restrictions:

$$\sum_{j \in \mathbf{N}_i} a_{ijt}^{+1} \leq 0 \quad i \in \mathbf{I} \quad (4.22)$$

$$\sum_{j \in \mathbf{N}_i} a_{ijt}^{-1} \geq 0 \quad i \in \mathbf{I} \quad (4.23)$$

Note that the holes in the same region gives exactly the same constraint. Hence, there are actually $2M$ different constraints in the estimation problems.

Figure 4.7 shows the evolution of adjusted R^2 values. As one may expect, since some further restrictions are imposed, these values are smaller than the ones obtained previously in Section 4.2.1. However, having larger adjusted R^2 values does not necessarily mean that model assumptions are correct as pointed out in [16]. In fact, we see that CLE works better than LE.

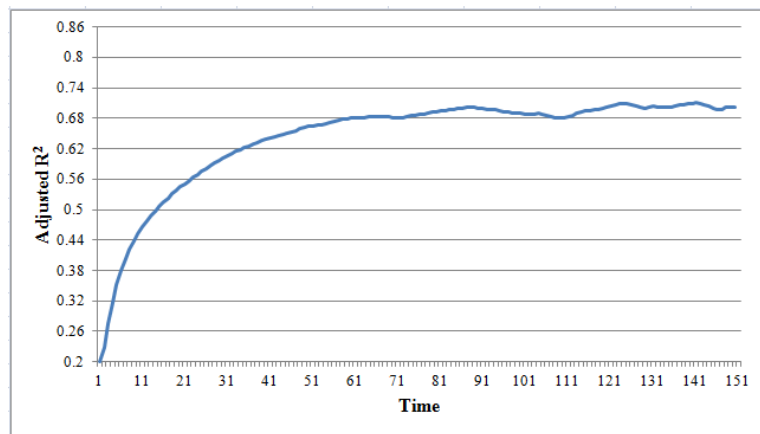


Figure 4.7. Adjusted R^2 values for CLE.

4.2.4. Corrected Quadratic Convex Estimation

The statistical model in Section 4.2.2 can give total shear stress effect functions similar to those given in Figure 4.8.

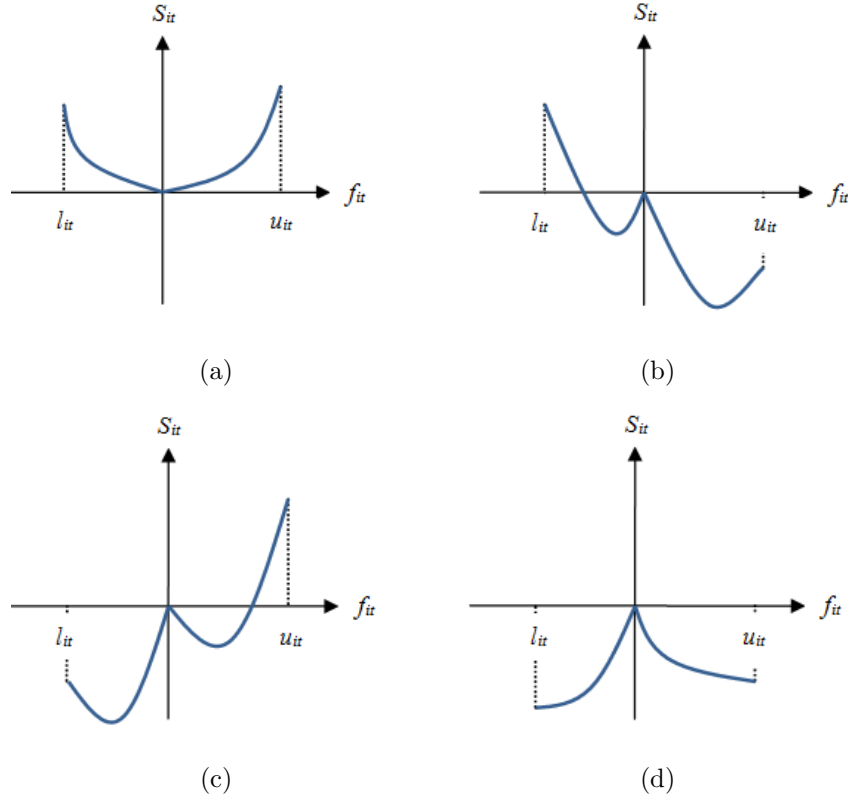


Figure 4.8. Sample S_{it} functions in (a), (b), (c) and (d) obtained using QE.

Due to the same reasoning as in the previous section, only the function in Figure 4.8d represents the correct physical phenomenon the best. Hence, the following constraints should be added to produce total shear stress effect functions which are increasing for negative flow values and decreasing for positive flow values:

$$2u_{it} \sum_{j \in N_i} a_{ijt}^{+2} + \sum_{j \in N_i} a_{ijt}^{+1} \leq 0 \quad i \in \mathbf{I} \quad (4.24)$$

$$2l_{it} \sum_{j \in N_i} a_{ijt}^{-2} + \sum_{j \in N_i} a_{ijt}^{-1} \geq 0 \quad i \in \mathbf{I} \quad (4.25)$$

Here, u_{it} and l_{it} are predetermined upper and lower bound values, respectively, which are taken as $u_{it} = 0.7 \frac{t}{T} f_{it}$ and $l_{it} = 0.7 \frac{t}{T} f_{it}$, $i \in \mathbf{I}$, $t \in \mathbf{T}$. Note that the number of constraints can be reduced to $2M$ by taking advantage of the property of regions again.

For every region, only the largest u_{it} and the smallest l_{it} give relevant constraints since the other constraints coming from other holes in the same region are redundant.

Also, we can force the convexity of the estimated functions by adding the following constraints:

$$\sum_{j \in \mathbf{N}_i} a_{ijt}^{+2} \geq 0 \quad i \in \mathbf{I} \quad (4.26)$$

$$\sum_{j \in \mathbf{N}_i} a_{ijt}^{-2} \geq 0 \quad i \in \mathbf{I} \quad (4.27)$$

Once again, there are exactly $2M$ different constraints each coming from different regions.

Figure 4.9 shows the evolution of adjusted R^2 values for this model called Corrected Quadratic Convex Estimation (CQCE). As one may expect, additional restrictions make these values to be smaller than the ones obtained previously in Section 4.2.2 but larger than the values in Section 4.2.3.

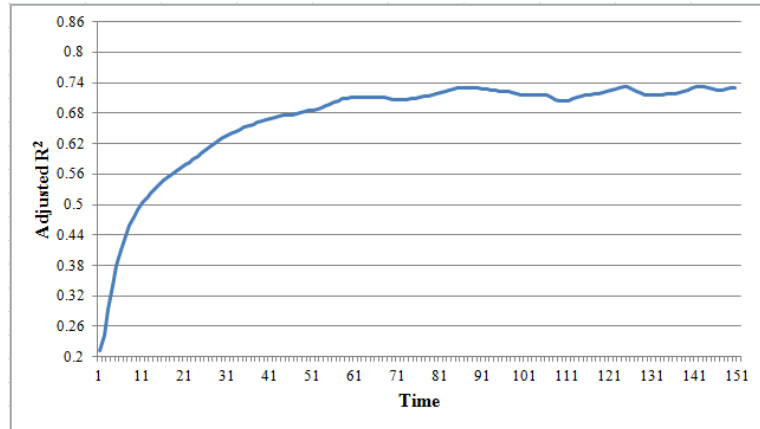


Figure 4.9. Adjusted R^2 values for CQCE.

4.2.5. Piecewise Linear Regression

Turbulence phenomenon is so complex that the statistical models used so far may not be sufficient to obtain accurate estimates for shear stress effect functions since

they are of degree at most two. However, piecewise linear functions are more flexible to explain general relationships provided that an adequate number of line segments, say K , is used. Therefore, we use Piecewise Linear Regression (PLR). Note that LE is a special case of PLR with $K = 2$.

The setting for PLR is as follows: Let $z_{t,k-1}$ and z_{tk} be the break points of the line segment k , $k \in \mathbf{K}$ for some predetermined values of z_{tk} , $k \in \mathbf{K} \cup \{0\}$. Let ρ_{itk} be one if $z_{t,k-1} \leq f_{it} \leq z_{tk}$ and zero otherwise. In particular, choose $z_{t0} = \min\{f_{it} : i \in \mathbf{I}\}$, $z_{t,\frac{K}{2}} = 0$ and $z_{tK} = \max\{f_{it} : i \in \mathbf{I}\}$. Then, the statistical model becomes as follows:

$$\widehat{S}_{jt} = a_t + \sum_{i \in \mathbf{N}_j} \sum_{k \in \mathbf{K}} \rho_{itk} (a_t^k + a_{ijt}^k f_{it}) \quad (4.28)$$

Figure 4.10 shows a sample S_{it} function.

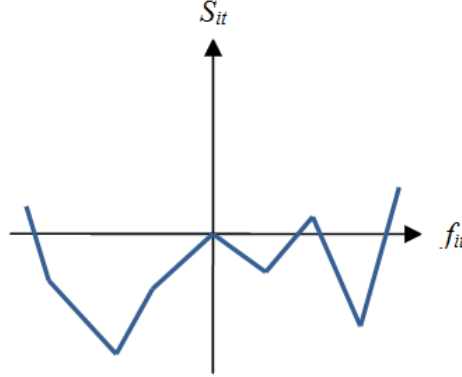


Figure 4.10. Sample S_{it} function obtained using PLR.

For convenience, a_t is still the common intercept whereas a_t^k is the intercept of line segment k for time period t . Therefore, $a_t^{\frac{K}{2}}$ and $a_t^{\frac{K}{2}+1}$ directly become zero since $z_{t,\frac{K}{2}} = 0$. Note that the selection of the other break points is an important problem itself. We should also guarantee continuity at break points. Hence, following constraints are added to the model:

$$\sum_{i \in \mathbf{N}_j} a_t^k + a_{ijt}^k z_{tk} = \sum_{i \in \mathbf{N}_j} a_t^{k+1} + a_{ijt}^{k+1} z_{tk} \quad j \in \mathbf{I}, k \in \mathbf{K} \setminus \{K\} \quad (4.29)$$

The number of these continuity constraints are $(K - 1)M$ since each region can only give one different constraint.

Figure 4.11 shows the evolution of adjusted R^2 values with $K = 8$ line segments. Here, the break points z_{t0} , z_{t4} and z_{t8} are selected as explained above. The remaining ones are taken as $-z_{t1} = z_{t7} = 0.012$, $-z_{t2} = z_{t6} = 0.006$ and $-z_{t3} = z_{t5} = 0.002$ for every period t .

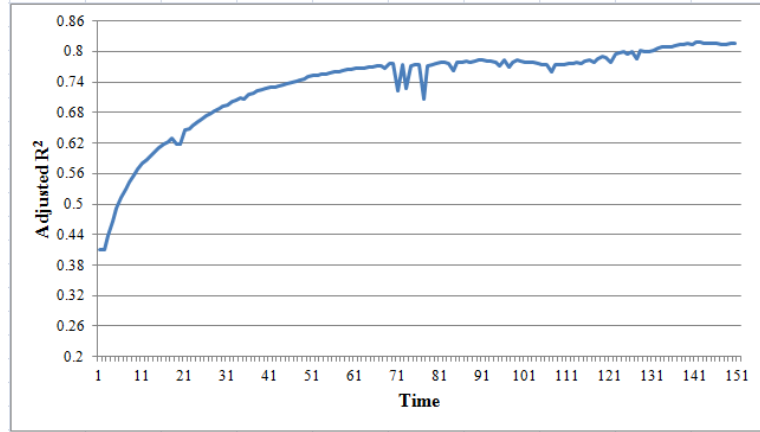


Figure 4.11. Adjusted R^2 values for PLR with $K = 8$.

It is a well-known fact that the quality of statistical estimations increases using more data points. We take advantage of this principle by including three different data sets to our PLR calculations. Our observation here is that although the size of the problem does not change, the resulting problems become numerically unstable. Hence, the selection of CPLEX 12.2 parameters, especially numerical emphasis, barrier algorithm and barrier starting point algorithm [42], become very crucial in obtaining quality results. Unfortunately, there does not exist a single setting of these parameters which works for every problem. Therefore, the only way is to run CPLEX 12.2 with different settings for each estimation problem and take the one which gives the highest adjusted R^2 value. Obviously, this is a very expensive method.

Figure 4.12 shows the evolution of adjusted R^2 values using three data sets. Use of three data sets gives slightly worse results in terms of adjusted R^2 values. However, we still use them since they are more reliable and estimate shear stress values more

realistically.

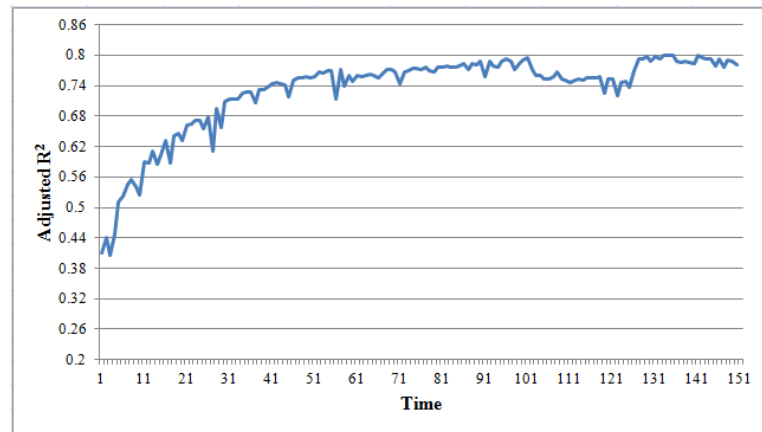


Figure 4.12. Adjusted R^2 values for PLR with $K = 8$ using three data sets.

4.2.6. Summary of Constrained Least Squares Estimation

Table 4.3 summarizes CLSE models used so far. As we might expect, the size of the problems gets larger while new parameters and constraints are added. Consequently, time needed for the estimation increases as the models become more complicated.

Table 4.3. Summary of CLSE.

Estimation Type	Number of Data Sets	Number of Parameters	Number of Constraints	CPU hours
LE	1	$1 + 2N_m$	-	12.8
QE	1	$1 + 4N_m$	-	19.9
CLE	1	$1 + 2N_m$	$2M$	13.1
CQCE	1	$1 + 4N_m$	$4M$	24.1
PLR	1	$1 + KN_m$	$(K - 1)M$	110.5
PLR	3	$1 + KN_m$	$(K - 1)M$	179.6

Note that the computation time required for PLR using three data sets only reflects the time elapsed using a fixed setting of parameters. Actual time is much more

than 179.6 hours due to the reasons explained earlier.

4.2.7. Solution Methods for Actuator Management Problem Using Constrained Least Squares Estimation

After obtaining the necessary parameters, we are able to identify the objective function. Recall that AMP is an NLP problem. Observe also that, because of the structure of the estimates used, the objective function is not differentiable at $f_{it} = 0$, $i \in \mathbf{I}$; $t \in \mathbf{T}$. However, due to the choice of lower and upper bound constraints on the flow values, either positive or negative part of each individual total shear stress effect function is used and hence, this is not a major problem for LE, QE, CLE and CQCE. For PLR, by construction, there are $K - 1$ nondifferentiable points for each S_{it} and they are specially treated by the binary variables.

Recall that Constraints 4.3-4.6 define a convex set. We have three different types of objective functions and the solution procedures for each of them are given as below.

4.2.7.1. Solution for Linear, Corrected Linear and Quadratic Convex Estimation. Using these estimation types, AMP becomes a CP problem since the objective is convex. Technically, such problems in which there are quadratic constraints (in our case, only energy) and the objective is either linear or quadratic is called as a quadratically constrained programming (QCP) problem and can be solved by CPLEX 12.2 [42] using Barrier Optimization provided that it is CP. Although we use CPLEX 12.2 in our experiments, an NLP solver such as KNITRO 6.0 [43] would be a better option in terms of efficiency. Finally, note that we have IT many continuous variables.

4.2.7.2. Solution for Quadratic Estimation. AMP becomes a Difference of Convex (DC) optimization problem, which is one of the most difficult problems in NLP, if this estimation type is used. Considering the size of AMP, it is practically impossible to be solved by DC optimization solution algorithm. Nevertheless, we present the appropriate solution procedure for the sake of completeness.

In order to solve a DC optimization problem, it should be in Canonical DC (CDC) form since the relevant algorithms work only under this assumption. Any twice continuously differentiable or, in short, C^2 function can be decomposed into its convex and concave parts as proven in [44]. Following the steps in [45], AMP can be put into its CDC form by taking advantage of the convexity of the constraint set. First of all, identify the convex and concave total shear stress effect functions and rewrite AMP as below:

$$\min \sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ convex}}} [a_t + S_{it}(f_{it})] + \sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ concave}}} [a_t + S_{it}(f_{it})] \quad (4.30)$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} f_{it}^2 \leq E \quad (4.31)$$

$$\sum_{i \in \mathbf{I}} f_{it} = 0 \quad t \in \mathbf{T} \quad (4.32)$$

$$l_{it} \leq f_{it} \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.33)$$

$$f_{it} \in \mathbb{R} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.34)$$

We now introduce an auxiliary variable, say ω_1 , in order to rewrite AMP as follows:

$$\min \quad \omega_1 \quad (4.35)$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} f_{it}^2 \leq E \quad (4.36)$$

$$\sum_{i \in \mathbf{I}} f_{it} = 0 \quad t \in \mathbf{T} \quad (4.37)$$

$$l_{it} \leq f_{it} \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.38)$$

$$\sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ convex}}} [a_t + S_{it}(f_{it})] + \sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ concave}}} [a_t + S_{it}(f_{it})] - \omega_1 \leq 0 \quad (4.39)$$

$$\omega_1, f_{it} \in \mathbb{R} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.40)$$

Finally, CDC form of AMP can be obtained by introducing another auxiliary variable ω_2 as follows:

$$\min w_1 \tag{4.41}$$

$$\text{s.t. } \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} f_{it}^2 \leq E \tag{4.42}$$

$$\sum_{i \in \mathbf{I}} f_{it} = 0 \quad t \in \mathbf{T} \tag{4.43}$$

$$l_{it} \leq f_{it} \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \tag{4.44}$$

$$\sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ convex}}} [a_t + S_{it}(f_{it})] - \omega_1 - \omega_2 \leq 0 \tag{4.45}$$

$$\omega_2 + \sum_{\substack{(i,t) \in \mathbf{I} \times \mathbf{T} \\ S_{it} \text{ concave}}} [a_t + S_{it}(f_{it})] \leq 0 \tag{4.46}$$

$$\omega_1, \omega_2, f_{it} \in \mathbb{R} \quad i \in \mathbf{I}; t \in \mathbf{T} \tag{4.47}$$

Here, we assume that the decomposition of the objective function into its convex and concave parts are given. If the objective function is of degree two, then this decomposition is straightforward. However, if the degree of the objective function is greater than two, then coming up with this decomposition may not be a trivial task.

4.2.7.3. Solution for Piecewise Linear Regression. Under PLR, AMP becomes a piecewise linear approximation problem which is an MILP, in general. We have developed seven different such models with different characteristics following [12]. However, we only use Logarithmic (Log) and Convex Combination (CC) Models in our experiments for AMP. The other five models and their formulations together with a detailed discussion on why Log and CC models are preferred can be found in Appendix A.

In order to adequately represent these models, we assume that the following data are available: Let K be the number of line segments used in the approximation of each function, $k \in \mathbf{K} = \{1, \dots, K\}$ (for convenience, we further assume that $K = 2^D$, for

some $D \in \mathbb{Z}^+$ and $\mathbf{D} = \{1, \dots, D\}$). Assume that (f_{it}^{k-1}, f_{it}^k) represent the end points of the line segment k used in the piecewise linear approximation of functions S_{it} and e_{it} . Furthermore, suppose that total shear stress effect of hole i during period t due to f_{it}^k units of air flow is given as $S_{it}(f_{it}^k)$ whereas energy spent at the same hole is given as $e_{it}(f_{it}^k)$. By a previous assumption, let $e_{it}(f_{it}^k) = (f_{it}^k)^2$.

In Log and CC models, following substitutions are done in AMP formulation:

$$f_{it} = \sum_{k \in \mathbf{K} \cup \{0\}} f_{it}^k \lambda_{it}^k \quad (4.48)$$

$$S_{it}(f_{it}) = \sum_{k \in \mathbf{K} \cup \{0\}} S_{it}(f_{it}^k) \lambda_{it}^k \quad (4.49)$$

$$e_{it}(f_{it}) = \sum_{k \in \mathbf{K} \cup \{0\}} (f_{it}^k)^2 \lambda_{it}^k \quad (4.50)$$

Using the above substitutions, it is possible to rewrite AMP as follows:

$$\min \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \left[a_t + \sum_{k \in \mathbf{K} \cup \{0\}} S_{it}(f_{it}^k) \lambda_{it}^k \right] \quad (4.51)$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{K} \cup \{0\}} (f_{it}^k)^2 \lambda_{it}^k \leq E \quad (4.52)$$

$$\sum_{i \in \mathbf{I}} \sum_{k \in \mathbf{K} \cup \{0\}} f_{it}^k \lambda_{it}^k = 0 \quad t \in \mathbf{T} \quad (4.53)$$

$$l_{it} \leq \sum_{k \in \mathbf{K} \cup \{0\}} f_{it}^k \lambda_{it}^k \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.54)$$

$$\sum_{k \in \mathbf{K} \cup \{0\}} \lambda_{it}^k = 1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.55)$$

$$\lambda_{it}^k \geq 0 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \cup \{0\} \quad (4.56)$$

If shear stress effect functions are all convex, then the above formulation is sufficient. However, we know that they are not necessarily convex. Therefore, we need to use some additional variables and constraints for those functions which are not convex

so that piecewise linear approximation becomes valid. In fact, the distinction between two convex combination models lies here: The basic model CC has K additional binary variables for each nonconvex S_{it} function in the formulation. On the other hand, the logarithmic model Log has only $\log_2 K = D$ binary variables.

The additional formulation for CC is as follows:

$$\lambda_{it}^0 \leq y_{it}^1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.57)$$

$$\lambda_{it}^k \leq y_{it}^k + y_{it}^{k-1} \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \setminus \{K\} \quad (4.58)$$

$$\lambda_{it}^K \leq y_{it}^K \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.59)$$

$$\sum_{k \in \mathbf{K}} y_{it}^k = 1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (4.60)$$

$$y_{it}^k = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (4.61)$$

The additional formulation for Log is given as below:

$$\sum_{\substack{k \in \mathbf{K} \\ \phi_{dk}=1}} \phi_{dk} \lambda_{it}^k \leq y_{it}^d \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (4.62)$$

$$\sum_{\substack{k \in \mathbf{K} \\ \phi_{dk}=-1}} -\phi_{dk} \lambda_{it}^k \leq 1 - y_{it}^d \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (4.63)$$

$$y_{it}^d = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (4.64)$$

Here, ϕ_{dk} is the entry of a $D \times K$ matrix which is constructed according to the following rule:

$$\phi_{dk} = \begin{cases} \text{sgn} \left(\frac{K}{2} - k \right) & \text{if } d = 1 \\ \text{sgn} \left(2^{D-d} - \text{Rem}(k, 2^{D-d+2}) - 2^{D-d+1} \right) & \text{if } d \geq 2 \end{cases} \quad (4.65)$$

Here, $\text{Rem}(a, b)$ is the product of $\text{sgn}(a)$ with the remainder obtained when $|a|$ is

divided by b , $a \in \mathbb{Z}$ and $b \in \mathbb{Z}^+$. Note that this simple formula is valid since S_{it} is univariate. For the general case, refer to [12] and [46].

Observe that Log has ITD and CC has $IT(K+1)$ binary variables. Additionally, they both have $IT(K+1)$ many continuous variables. Therefore, they are very large-scale optimization problems and intractable to solve with the forms given above. Here, we make two simplifications: First of all, we relax Constraint 4.52 which results in a separation in the formulation in terms of time periods. Hence, instead of solving a single very large-scale problem, we now need to solve T many smaller but still large-scale MILPs. By relaxing this constraint, we change the problem entirely. However, we observe that the resulting solutions do not violate the relaxed constraint most of the time. Therefore, in practice, this approach works.

A second simplification is done in terms of binary variables. Due to the choice of lower and upper bounds, some of the line segments is never used. Therefore, equating the binary variables corresponding to those line segments reduces the number of binary variables significantly. In general, only two or three line segments are contained between lower and upper bounds. Notice that this simplification is completely legitimate since it does not change the structure of the problem.

Although the first simplification can be applied to both models, we can only use the second one in CC since the binary variables in Log do not have a physical meaning and hence, the second simplification is inapplicable. Therefore, we only use Log in one of the experiments whereas in the other experiments, we exploit this property of CC.

4.3. Neural Networks

After using CLSE, we decide to enhance our estimation procedure by using neural networks. Our purpose here is two-fold: On one hand, we try to find better results than the ones obtained using CLSE in terms of AMP. On the other hand, we have a chance to test the efficiency of our proposed NGRG to a real, large-scale problem.

4.3.1. Construction and Training of Neural Networks for Actuator Management Problem

Recall that the shear stress at some point j is a function of all the flow values on the grid. In terms of neural networks, it corresponds to a network with I many input and output neurons. However, this type of a network is very hard to train since it is very large and there does not exist a sufficient number of patterns. Instead of this approach, we construct smaller but numerous networks assuming that the network of the points in the same region is the same. In this case, input neurons are the flow amounts of the neighbours and the output neuron is the shear stress value.

The patterns used in backpropagation algorithm are constructed as follows: Since the network of the points in the same region is identical, each point can be seen as a different pattern. Figure 4.13 is an example of how the neural network corresponding to $i = 7$ is constructed in Figure 4.2. Note that there are three more patterns coming from the same region.

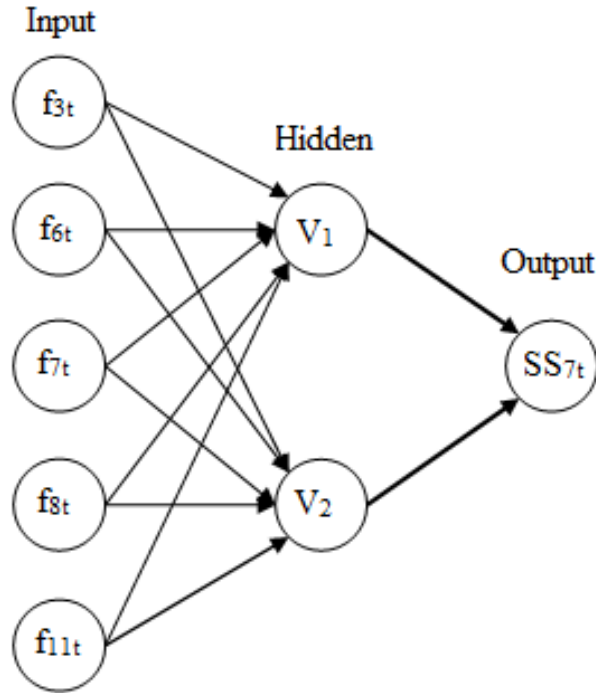


Figure 4.13. An example for the construction of neural network.

Using the same reasoning as above, the number of patterns that can be extracted from a single set of data for AMP is given in Table 4.4.

Table 4.4. Region and pattern numbers.

Size of the region	Number of regions	Number of patterns
1×1	400	1
108×1	20	108
1×236	20	236
108×236	1	25488

Observe that 1×1 regions have only one pattern coming from a single set of data which is obviously not enough for the training of neural network. For 108×1 and 1×236 regions, the number of patterns is larger but still insufficient. Therefore, we use three different sets of input data accumulated through our experimentations using CLSE. We decide to use CQCE for 1×1 regions instead of constructing neural networks since there is still insufficient number of patterns. Also, the number of patterns for the 108×236 region becomes too many if all the patterns coming from three data sets are used. Hence, we randomly select 10000 out of 25488 possible patterns from each data set. The number of patterns for each region actually used is given in Table 4.5.

Table 4.5. Regions and actual patterns numbers.

Size of the region	Number of regions	Number of patterns
108×1	20	324
1×236	20	708
108×236	1	30000

Note that for each time period t , we need to run backpropagation algorithm for 41 different neural networks to estimate underlying shear stress function corresponding to each region.

Fine-tuning of parameters of backpropagation algorithm is essential in order to obtain the best performance. For this purpose, we experiment with different settings and realize that the values given in Table 4.6 give the best overall results.

Table 4.6. Parameters of neural networks.

Number of hidden units	7
Initial learning rate	0.1, 0.01 or 0.001
Decay rate of learning rate	0.02
Number of epochs	50
Number of replications	5

Note that we use a fixed number of hidden layer units to limit the time spent in the training. Also, initial learning rate is adjusted according to the size of regions. It is taken as 0.1, 0.01 and 0.001 for regions with size 108×1 , 1×236 and 108×236 , respectively.

The quality of neural network estimation is measured by calculating R^2 values but not adjusted R^2 . This is mainly because the number of parameters is variable with respect to different neural networks. Observe that since the sample size is quite large, the values can be still comparable. Figure 4.14 shows the evolution of R^2 values.

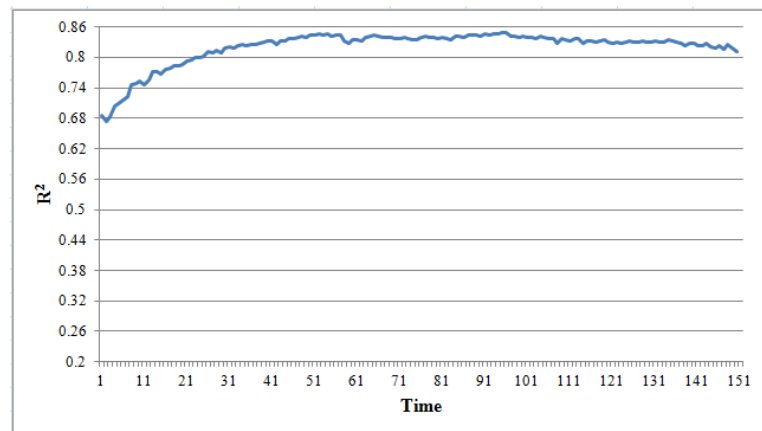


Figure 4.14. R^2 values for neural networks.

Observe that these values are significantly better than the ones obtained so far, especially for the first 50 time periods. This allows us to enlarge the feasible region by selecting larger α_t^+ parameters since neural networks seem to give better function estimates. As a final note, this estimation takes about 10 CPU days and hence, requires much more processing time than any of CLSE models.

4.3.2. Solution Method for Actuator Management Problem Using Neural Networks

Once the neural networks are constructed, the objective function is identified and AMP can be solved. Since energy constraint is nonlinear, we need to use NGRG instead of NRG mentioned in Section 3.3. Furthermore, the energy constraint is split for every period t in order to solve T many smaller problems instead a single, large-scale problem. The corresponding energy upper bound for period t is taken as

$$E_t = \beta_t \sum_{i \in \mathbf{I}} f_{it}^2 \quad (4.66)$$

where $\beta_t \geq 1$.

The initial feasible solution for each subproblem t is generated using CPLEX 12.2 [42]. We terminate GRG after 500 iterations. This number may seem small, however, it takes more than four CPU days to execute the problem even with this setting. Therefore, it is also practically impossible to initialize the algorithm with different points.

The main difficulty in the implementation of NGRG is due to the correction step needed when a solution obtained in an iteration is outside the feasible region. Let us denote the infeasible solution obtained in solving the problem corresponding to some time period t by $\tilde{f}_{it}$, for $i \in \mathbf{I}$. Then, this solution should be projected back to the constraint space. The general procedure is to solve a nonlinear system of equalities starting from $\tilde{f}_{it}$ until a feasible solution is obtained. However, we decide to find the

nearest feasible solution by solving the following QCP using CPLEX 12.2:

$$\min \sum_{i \in \mathbf{I}} (f_{it} - \tilde{f}_{it})^2 \quad (4.67)$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} f_{it}^2 \leq E_t \quad (4.68)$$

$$\sum_{i \in \mathbf{I}} f_{it} = 0 \quad (4.69)$$

$$l_{it} \leq f_{it} \leq u_{it} \quad i \in \mathbf{I} \quad (4.70)$$

$$f_{it} \in \mathbb{R} \quad i \in \mathbf{I} \quad (4.71)$$

5. COMPUTATIONAL EXPERIMENTS

5.1. Test for Neural Reduced Gradient Method

The accuracy of the proposed method NRG is tested using the problems given in this section. Their optimal solutions and some further features of the functions are provided in Appendix B. The test results found under different assumptions are presented in Section 5.1.3 in detail.

5.1.1. Test Instances for Neural Reduced Gradient Method

NRG is tested using 19 problem instances. The first five problems, denoted by $QP(m, n)$, are randomly generated quadratic programming problems where m is the number of constraints and n is the number of variables. The mathematical formulation is as follows:

$$\min \mathbf{x}^T \mathbf{D} \mathbf{x} + \mathbf{c}^T \mathbf{x} \quad (5.1)$$

$$\text{s.t. } \mathbf{A} \mathbf{x} = \mathbf{b} \quad (5.2)$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \quad (5.3)$$

Here, $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with positive entries, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{l} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^n$ are some given vectors and $\mathbf{x} \in \mathbb{R}^n$ is the vector of unknowns. Therefore, these are convex quadratic programming problems with their sizes given as below:

- (i) $QP(3, 5)$
- (ii) $QP(6, 8)$
- (iii) $QP(8, 12)$
- (iv) $QP(9, 15)$
- (v) $QP(12, 20)$

The remaining 14 problems are taken from the literature and only one of the sources they appear is given. A single constraint which includes all the variables with nonzero coefficients is added so that the problems become nondegenerate and hence, RG can be applied.

(vi) Matyas Function [47]:

$$\min 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \quad (5.4)$$

$$\text{s.t. } x_1 + x_2 \leq 2 \quad (5.5)$$

$$-2 \leq x_1, x_2 \leq 2 \quad (5.6)$$

(vii) Trid Function [47]:

$$\min \sum_{j=1}^n (x_j - 1)^2 - \sum_{j=2}^n x_j x_{j-1} \quad (5.7)$$

$$\text{s.t. } \sum_{j=1}^n x_j \leq \frac{n(n+1)(n+2)}{6} \quad (5.8)$$

$$-n^2 \leq x_j \leq n^2 \quad j = 1, \dots, n \quad (5.9)$$

In the literature, either $n = 6$ or $n = 10$ is used. We prefer the former one.

(viii) Zakharov Function [47]:

$$\min \sum_{j=1}^n x_j^2 + \left(\sum_{j=1}^n 0.5jx_j \right)^2 + \left(\sum_{j=1}^n 0.5jx_j \right)^4 \quad (5.10)$$

$$\text{s.t. } \sum_{j=1}^n x_j \leq 2 \quad (5.11)$$

$$-1 \leq x_j \leq 2 \quad j = 1, \dots, n \quad (5.12)$$

Although any positive integer n is applicable, we use $n = 3$.

(ix) Colville Function [47]:

$$\begin{aligned} \min \quad & 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 \\ & + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1) \end{aligned} \quad (5.13)$$

$$\text{s.t. } x_1 + x_2 \leq 6 \quad (5.14)$$

$$-2 \leq x_1, x_2, x_3, x_4 \leq 2 \quad (5.15)$$

(x) Beale Function [47]:

$$\min \quad (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2 \quad (5.16)$$

$$\text{s.t. } x_1 + x_2 \leq 6 \quad (5.17)$$

$$-4 \leq x_1, x_2 \leq 4 \quad (5.18)$$

(xi) Rosenbrock Function [47]:

$$\min \quad (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \quad (5.19)$$

$$\text{s.t. } x_1 + x_2 \leq 3 \quad (5.20)$$

$$-1 \leq x_1, x_2 \leq 2 \quad (5.21)$$

(xii) Dixon & Price Function [47]:

$$\min \quad (x_1 - 1)^2 + \sum_{j=2}^n j(2x_j^2 - x_{j-1})^2 \quad (5.22)$$

$$\text{s.t. } \sum_{j=1}^n x_j \leq \frac{n}{2} + 1 \quad (5.23)$$

$$-1 \leq x_j \leq 2 \quad j = 1, \dots, n \quad (5.24)$$

Although any positive integer n is applicable, we use $n = 5$.

(xiii) Ravindran Function [48]:

$$\min 2x_1^2 + 4x_1x_2^3 - 10x_1x_2 + x_2^2 \quad (5.25)$$

$$\text{s.t. } x_1 + x_2 \leq 3 \quad (5.26)$$

$$-1.75 \leq x_1, x_2 \leq 1.75 \quad (5.27)$$

(xiv) Winston Function [49]:

$$\min x_1^2x_2 + x_2^3x_1 - x_1x_2 \quad (5.28)$$

$$\text{s.t. } x_1 + x_2 \leq 1 \quad (5.29)$$

$$0.2 \leq x_1, x_2 \leq 0.5 \quad (5.30)$$

(xv) Himmelblau Function [50]:

$$\min (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \quad (5.31)$$

$$\text{s.t. } x_1 + x_2 \leq 6 \quad (5.32)$$

$$-4 \leq x_1, x_2 \leq 4 \quad (5.33)$$

(xvi) Branin Function [47]:

$$\min \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10 \quad (5.34)$$

$$\text{s.t. } x_1 + x_2 \leq 15 \quad (5.35)$$

$$-5 \leq x_1 \leq 11; 1 \leq x_2 \leq 15 \quad (5.36)$$

(xvii) Six-Hump Function [51]:

$$\min 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \quad (5.37)$$

$$\text{s.t. } x_1 + x_2 \leq 4 \quad (5.38)$$

$$-2.5 \leq x_1, x_2 \leq 2.5 \quad (5.39)$$

(xviii) Goldstein & Price Function [52]:

$$\min \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \times \quad (5.40)$$

$$\left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \right]$$

$$\text{s.t. } x_1 + x_2 + x_3 + x_4 \leq 4 \quad (5.41)$$

$$-2 \leq x_1, x_2, x_3, x_4 \leq 2 \quad (5.42)$$

(xix) Broyden Tridiagonal Function [53]:

$$\min \left[(3 - 2x_1)x_1 - 2x_2 + 1 \right]^2 + \sum_{j=2}^{n-1} \left[(3 - 2x_j)x_j - x_{j-1} - 2x_{j+1} + 1 \right]^2 \quad (5.43)$$

$$+ \left[(3 - 2x_n)x_n - x_{n-1} + 1 \right]^2$$

$$\text{s.t. } \sum_{j=1}^n x_j \leq 0 \quad (5.44)$$

$$-2 \leq x_j \leq 1 \quad j = 1, \dots, n \quad (5.45)$$

In the literature, $n = 10$ is preferred and we use it as well.

5.1.2. Generation of Feasible Patterns

In a real application such as drag reduction, feasible patterns have already been given. However, for the above mentioned test problems, this is not the case and we need to generate set of feasible patterns to train the neural networks. In short, we want to find some elements of the polytope $P = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$. For this purpose, we define another polytope $P' = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}, \mathbf{l}' \leq \mathbf{x} \leq \mathbf{u}'\}$ where $\mathbf{l} \leq \mathbf{l}' \leq \mathbf{u}' \leq \mathbf{u}$. Consider the following LP

$$\min \tilde{\mathbf{c}}^T \mathbf{x} \quad (5.46)$$

$$\text{s.t. } \mathbf{x} \in P' \quad (5.47)$$

where $\tilde{\mathbf{c}} \in \mathbb{R}^n$ is a randomly generated cost vector.

If $P' \neq \emptyset$, then an optimal solution, say $\mathbf{x}^*$, exists. Since $P' \subseteq P$ by construction, we conclude that $\mathbf{x}^* \in P$. Hence, by selecting different $\mathbf{l}'$, $\mathbf{u}'$ and $\tilde{\mathbf{c}}$ vectors randomly, it is possible to generate feasible patterns for our problems. Note that the generation of each pattern suffices to solve an LP. If it is infeasible, then we can ignore that particular problem and generate other random vectors.

5.1.3. Result of Test Instances

In order to validate NRG, we solve the test problems under different assumptions. First of all, we assume that the simulator is deterministic in Section 5.1.3.1 and then, random in Section 5.1.3.2. Then, 1000 and 2000 patterns are used to train NNRT and NNCT. Finally, finite difference (FD) and exact gradient (EG) are used to estimate objective gradient. Hence, we design an experiment with 16 settings.

Every problem is solved 10 times with different initial points for each setting. We terminate the algorithm after 25000 iterations unless the norm of the estimated reduced gradient becomes less than a small positive number earlier. Although finding average performance measures for NRG are easy such as the average execution time $\overline{\text{CPU}}$ in seconds or the average actual objective function value $\overline{F(\mathbf{x})}$, determining the best outcome among these 10 replications may be nontrivial. Here, we use two criteria: The first one is to select the point, say $\mathbf{x}_1$, which gives the smallest fitted objective function value. However, this point may not yield the best actual result. Therefore, as a second criterion, we find the point, say $\mathbf{x}_2$, which gives the smallest actual objective function value at the expense of 10 additional simulation runs.

We should also need some measures to assess the performance of the method. Calculating the relative deviation is inapplicable since the global minimum of some of the functions is zero. Hence, we decide to find the absolute difference from the nearest local minimum, i.e., $F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$ or $F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$ where $\mathbf{x}_1^N$ and $\mathbf{x}_2^N$ are the nearest local minima from points $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively.

Another measure could be the distance from the nearest local minimum i.e., $\|\mathbf{x}_1 - \mathbf{x}_1^N\|$ or $\|\mathbf{x}_2 - \mathbf{x}_2^N\|$. However, since the variables in each function have different orders of magnitude, we scale them by dividing $\|\mathbf{u} - \mathbf{l}\|$, which is the length of the diagonal of the box defined by the lower and upper bounds of the variables. Note that this measure is highly dependent on the choice of the bounds $\mathbf{u}$ and $\mathbf{l}$.

5.1.3.1. Deterministic Case. We first solve the problems assuming that the simulator is inexpensive and hence, using FD directly is possible. Note that in this case $\mathbf{x}_1 = \mathbf{x}_2$. This experiment also allows us to test the accuracy of RG we develop. The results are given in Table 5.1.

Table 5.1. Results summary for a deterministic, inexpensive simulator.

Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	0.0	1.968	0.000	0.002	0.017
QP(6, 8)	0.0	-8.970	0.000	0.008	0.059
QP(8, 12)	2.0	-13.758	0.000	0.001	0.006
QP(9, 15)	2.0	-11.888	0.000	0.003	0.012
QP(12, 20)	3.0	-6.059	0.000	0.001	0.006
Matyas	0.0	0.000	0.000	0.008	0.150
Trid	0.0	-50.000	0.000	0.000	0.000
Zakharov	0.0	0.000	0.000	0.000	0.000
Colville	0.0	4.879	0.000	0.000	0.000
Beale	0.0	0.000	0.000	0.001	0.012
Rosenbrock	0.0	0.002	0.000	0.009	0.222
Dixon & Price	0.0	0.200	0.000	0.000	0.000
Ravindran	0.0	-2.877	0.000	0.001	0.020
Winston	0.0	-0.072	0.000	0.003	0.667
Himmelblau	0.0	0.000	0.000	0.000	0.000
Branin	0.0	0.399	0.000	0.000	0.000
Six-Hump	0.0	-0.718	0.000	0.001	0.020
Goldstein & Price	0.0	30.001	0.000	0.001	0.025
Broyden	0.9	2.390	0.000	0.001	0.011
Average	0.4	-2.869	0.000	0.002	0.065

The above table shows that our RG works efficiently and accurately since it finds the global minimum of each problem in at least one of 10 replications. Although it converges to a local minimum for some problems such as Branin [47], Goldstein & Price [52] and Six-Hump [51] depending of the initial points, this is an expected result since RG has only a local convergence. Therefore, it is also advisable to start NRG with multiple initial points so that different local minima can be detected.

Now, suppose that the simulator is expensive and using FD is not possible directly. Assume further that we are given 1000 and 2000 feasible patterns. In order to use NRG, we need to train the neural networks using these patterns. The following parameters are used in backpropagation algorithm for the given network type:

Table 5.2. Parameters of neural networks for NRG.

Parameter	NNRT	NNCT
Initial learning rate	0.1	0.01
Decay rate of learning rate	0.01	0.01
Minimum number of epochs	500	50
Maximum number of epochs	1000	1000
Number of replications	10	5

The initial number of hidden units is selected as five or half of the number of input units, whichever is greater, and it is increased in a constructive way as mentioned earlier. Backpropagation algorithm is terminated if test error increases during five consecutive epochs in order to prevent overlearning. We use less number of replications to train NNCT in order to limit processing time.

Table 5.3 shows time needed to train 1000 and 2000 patterns if NNRT and NNCT are used. As we can expect, the training of 2000 patterns takes twice as much as training 1000 patterns. Also, the training of NNCT takes significantly more time than training NNRT for both 1000 and 2000 patterns. Later, we see that allotting more time for

training pays off in terms of accuracy.

Table 5.3. Training time of neural networks with deterministic inputs.

Problem	1000 patterns		2000 patterns	
	NNRT	NNCT	NNRT	NNCT
QP(3, 5)	122	764	637	2617
QP(6, 8)	182	3195	485	11370
QP(8, 12)	174	10608	380	30039
QP(9, 15)	274	11030	610	44816
QP(12, 20)	436	30997	872	92621
Matyas	149	726	287	614
Trid	171	2137	574	22307
Zakharov	196	320	411	1080
Colville	305	399	471	2536
Beale	149	210	481	473
Rosenbrock	157	206	381	933
Dixon & Price	122	783	946	2685
Ravindran	111	162	220	472
Winston	149	204	297	628
Himmelblau	313	150	301	780
Branin	143	204	234	636
Six-Hump	146	125	307	785
Goldstein & Price	247	190	378	631
Broyden	184	802	742	5159
Average	196.3	3326.9	474.4	11641.2

We design eight experiments under the assumption that the simulator is deterministic and expensive. First, we generate 1000 feasible patterns and construct both NNRT and NNCT. For 10 different initial points, FD and EG methods are used to estimate the objective gradient. Results for each problem can be seen in Tables 5.4-5.7. Then, the same experiments are conducted assuming that 2000 patterns are available and results are given in Tables 5.8-5.11.

Table 5.4. Results for a deterministic simulator with 1000 patterns and NNRT is used with FD method.

Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	1.3	2.567	0.592	0.567	4.707	0.591	0.568	4.713
QP(6, 8)	0.0	-7.444	1.448	0.826	6.487	1.407	0.813	6.389
QP(8, 12)	0.5	-13.361	0.382	0.328	1.950	0.382	0.328	1.950
QP(9, 15)	5.0	-2.309	9.587	1.600	6.671	9.561	1.598	6.665
QP(12, 20)	8.2	-1.056	5.004	1.426	6.244	4.993	1.425	6.239
Matyas	0.4	0.049	0.010	0.708	12.512	0.010	0.708	12.512
Trid	1.1	-47.470	2.530	2.730	4.423	2.528	2.730	4.422
Zakharov	1.1	0.115	0.115	0.331	6.374	0.114	0.330	6.351
Colville	2.0	13.637	13.638	1.059	13.242	13.633	1.059	13.242
Beale	1.0	15.633	47.163	3.902	34.489	2.147	0.814	7.193
Rosenbrock	1.0	5.558	16.898	2.032	47.894	0.694	1.260	29.706
Dixon & Price	1.0	4.000	3.998	1.049	15.639	3.997	1.049	15.639
Ravindran	1.0	-2.417	0.312	0.373	7.545	0.312	0.373	7.545
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.6	66.423	92.859	1.394	12.322	39.994	1.415	12.506
Branin	0.8	15.301	18.088	2.015	9.478	2.165	2.222	10.452
Six-Hump	1.0	0.503	1.535	0.759	10.739	1.534	0.759	10.737
Goldstein & Price	1.0	3930.847	3904.960	1.372	24.250	3890.602	1.370	24.225
Broyden	0.2	55.900	207.827	3.420	36.052	9.628	2.427	25.578
Average	1.4	212.443	227.734	1.369	15.142	209.700	1.124	12.250

Table 5.5. Results for a deterministic simulator with 1000 patterns and NNRT is used with EG method.

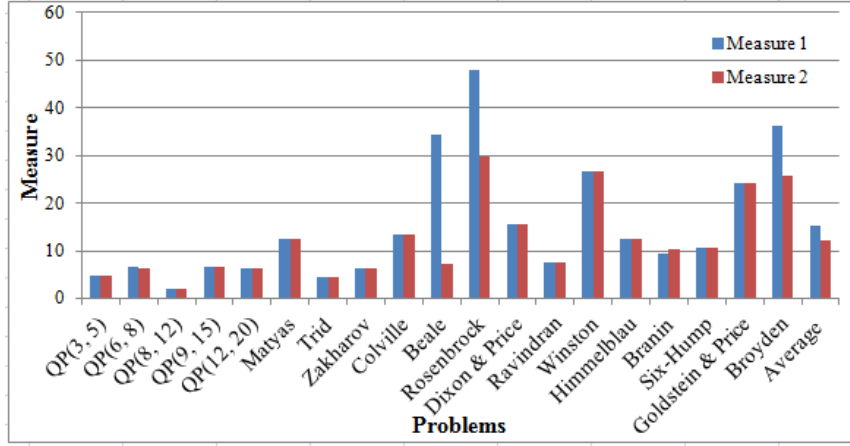
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	1.0	2.560	0.593	0.567	4.703	0.591	0.568	4.713
QP(6, 8)	0.0	-7.605	1.427	0.820	6.437	0.804	0.615	4.829
QP(8, 12)	0.0	-13.380	0.395	0.332	1.974	0.217	0.272	1.617
QP(9, 15)	3.8	-2.309	9.587	1.600	6.671	9.562	1.598	6.665
QP(12, 20)	6.1	-1.055	4.996	1.425	6.239	4.996	1.425	6.239
Matyas	0.3	0.052	0.010	0.707	12.500	0.010	0.707	12.500
Trid	0.9	-41.310	2.535	2.730	4.423	2.528	2.730	4.423
Zakharov	1.0	0.138	0.115	0.331	6.369	0.115	0.331	6.369
Colville	1.0	10.887	3.713	1.862	23.277	3.713	1.862	23.277
Beale	1.0	34.292	47.402	3.902	34.490	2.139	0.813	7.185
Rosenbrock	1.0	5.564	16.891	2.032	47.894	0.695	1.260	29.706
Dixon & Price	1.0	4.001	4.000	1.049	15.639	3.997	1.049	15.639
Ravindran	1.0	-2.907	0.312	0.373	7.545	0.312	0.373	7.545
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.7	77.159	92.855	1.394	12.322	40.003	1.414	12.494
Branin	0.7	15.523	18.088	2.015	9.478	2.165	2.222	10.452
Six-Hump	1.0	0.503	1.535	0.758	10.725	1.533	0.759	10.734
Goldstein & Price	1.0	1795.034	3906.639	1.372	24.250	366.871	0.602	10.650
Broyden	0.3	63.112	207.794	3.420	36.054	11.706	2.592	27.324
Average	1.1	102.116	227.310	1.411	15.667	23.788	1.121	12.055

Table 5.6. Results for a deterministic simulator with 1000 patterns and NNCT is used with FD method.

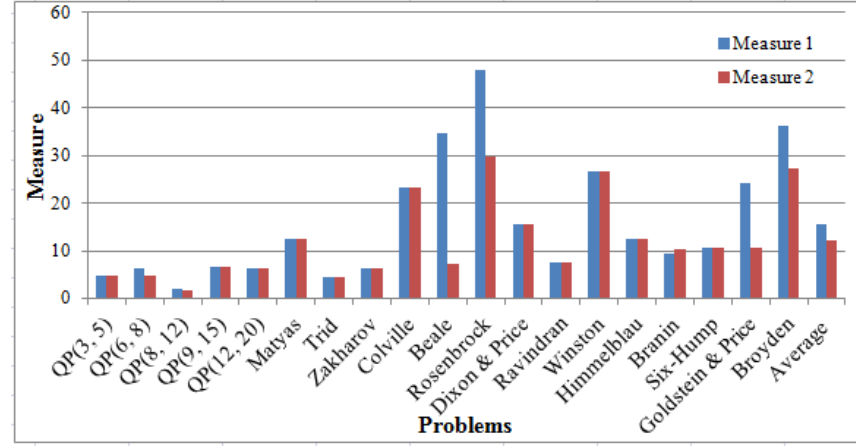
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	103.6	1.986	0.018	0.103	0.856	0.018	0.103	0.856
QP(6, 8)	7.9	-8.974	0.000	0.016	0.128	0.000	0.005	0.040
QP(8, 12)	201.1	-13.752	0.006	0.047	0.281	0.003	0.038	0.227
QP(9, 15)	1192.3	-11.881	0.007	0.052	0.217	0.007	0.052	0.217
QP(12, 20)	2480.0	-5.769	0.290	0.320	1.399	0.290	0.320	1.399
Matyas	21.0	0.000	0.000	0.044	0.775	0.000	0.044	0.775
Trid	157.0	-49.870	0.130	1.143	1.852	0.130	1.143	1.852
Zakharov	40.0	0.002	0.002	0.024	0.461	0.002	0.024	0.461
Colville	69.0	9.648	10.208	2.516	31.449	6.586	0.528	6.599
Beale	19.4	23.987	13.331	0.491	4.341	13.116	6.471	57.195
Rosenbrock	13.2	1.360	0.093	0.761	17.938	0.093	0.761	17.938
Dixon & Price	107.3	1.885	0.927	0.767	11.438	0.926	0.768	11.442
Ravindran	17.3	-2.499	0.026	0.116	2.345	0.026	0.116	2.345
Winston	0.3	-0.071	0.001	0.017	3.923	0.001	0.017	3.923
Himmelblau	19.4	0.820	0.863	0.164	1.447	0.814	0.144	1.269
Branin	18.4	0.674	0.181	0.170	0.800	0.181	0.170	0.800
Six-Hump	14.9	1.412	1.569	0.546	7.726	1.388	0.496	7.012
Goldstein & Price	19.9	1127.125	156.450	0.320	5.663	481.215	0.634	11.200
Broyden	1.8	9.639	9.434	2.101	22.146	9.401	2.102	22.156
Average	237.0	57.143	10.186	0.512	6.062	27.063	0.733	7.774

Table 5.7. Results for a deterministic simulator with 1000 patterns and NNCT is used with EG method.

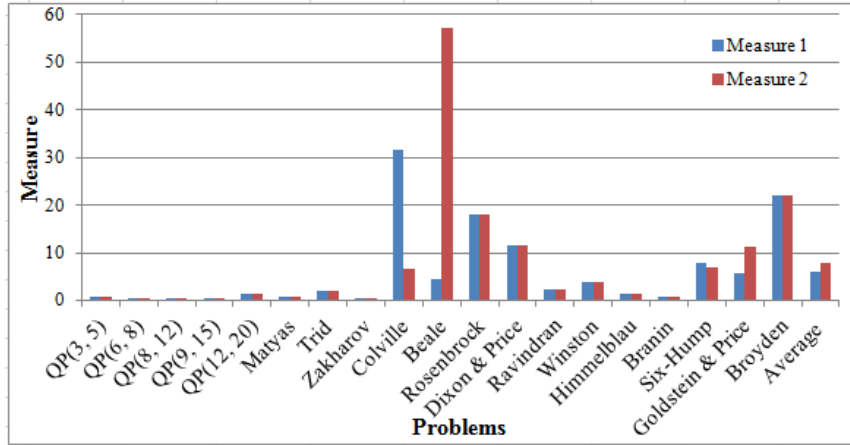
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	84.0	1.986	0.018	0.103	0.853	0.018	0.103	0.853
QP(6, 8)	4.9	-8.976	0.000	0.016	0.128	0.000	0.016	0.128
QP(8, 12)	330.2	-13.752	0.006	0.047	0.282	0.005	0.046	0.273
QP(9, 15)	703.1	-11.881	0.007	0.052	0.219	0.007	0.052	0.219
QP(12, 20)	1297.8	-5.769	0.290	0.320	1.399	0.290	0.320	1.399
Matyas	16.3	0.000	0.000	0.040	0.713	0.000	0.040	0.713
Trid	107.7	-49.870	0.130	1.143	1.852	0.130	1.143	1.852
Zakharov	34.0	0.002	0.002	0.025	0.477	0.002	0.025	0.477
Colville	56.0	9.548	10.236	2.517	31.460	6.601	0.528	6.603
Beale	17.0	15.891	13.228	0.490	4.333	13.116	6.471	57.195
Rosenbrock	17.0	1.734	4.005	2.000	47.141	0.214	0.791	18.645
Dixon & Price	77.8	1.439	0.927	0.768	11.445	0.926	0.768	11.442
Ravindran	14.7	-2.840	0.026	0.116	2.344	0.026	0.116	2.344
Winston	0.6	-0.071	0.001	0.030	7.157	0.001	0.030	7.157
Himmelblau	17.0	0.829	0.864	0.164	1.447	0.813	0.144	1.272
Branin	16.3	0.822	0.184	0.170	0.802	0.184	0.170	0.802
Six-Hump	13.2	1.396	1.566	0.545	7.706	1.113	0.422	5.963
Goldstein & Price	17.0	988.824	156.513	0.320	5.663	481.215	0.634	11.200
Broyden	126.4	9.409	9.377	2.105	22.188	9.374	2.105	22.190
Average	155.3	49.406	10.388	0.577	7.769	27.054	0.733	7.933



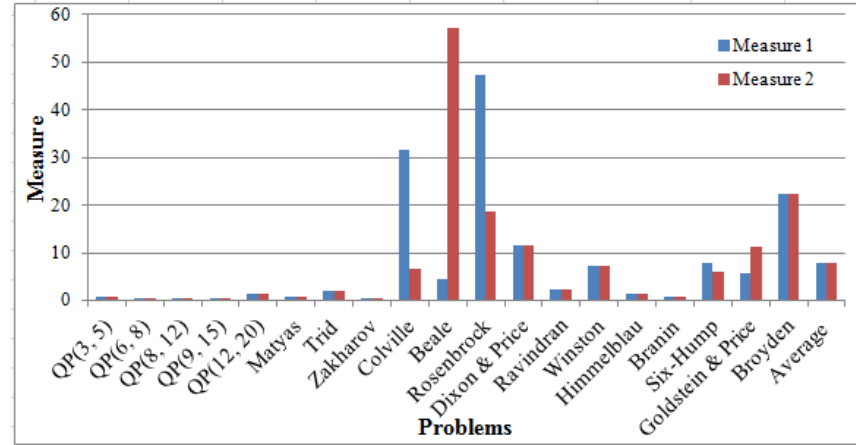
(a)



(b)



(c)



(d)

Figure 5.1. Measures for test problems using 1000 deterministic patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.

Table 5.8. Results for a deterministic simulator with 2000 patterns and NNRT is used with FD method.

Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	1.6	2.293	0.324	0.498	4.136	0.324	0.498	4.136
QP(6, 8)	0.0	-8.653	0.222	0.324	2.547	0.161	0.275	2.162
QP(8, 12)	3.3	-12.775	0.981	0.608	3.622	0.981	0.608	3.622
QP(9, 15)	5.1	-4.024	7.861	1.501	6.257	7.858	1.500	6.255
QP(12, 20)	8.1	0.174	6.233	1.455	6.368	6.230	1.455	6.368
Matyas	0.2	0.103	0.010	0.707	12.502	0.010	0.707	12.506
Trid	2.0	-44.094	5.906	7.560	12.248	5.906	7.560	12.248
Zakharov	1.0	0.045	0.045	0.103	1.977	0.045	0.103	1.977
Colville	1.1	16.327	16.273	1.114	13.927	16.273	1.114	13.927
Beale	1.0	2.374	0.011	0.275	2.430	0.010	0.274	2.422
Rosenbrock	1.0	1.626	1.345	1.272	29.970	0.021	0.279	6.582
Dixon & Price	2.0	4.654	3.428	1.075	16.023	3.428	1.075	16.023
Ravindran	0.8	-2.561	0.110	0.236	4.769	0.110	0.236	4.769
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.5	32.782	10.908	0.442	3.906	0.280	0.124	1.093
Branin	0.8	3.625	9.660	2.192	10.312	1.139	1.319	6.205
Six-Hump	1.0	0.917	1.130	0.498	7.045	1.128	0.498	7.045
Goldstein & Price	0.0	5278.000	5275.000	1.414	25.000	5275.000	1.414	25.000
Broyden	0.0	11.005	4.828	2.685	28.297	4.828	2.685	28.297
Average	1.6	277.987	281.278	1.267	11.474	280.197	1.149	9.859

Table 5.9. Results for a deterministic simulator with 2000 patterns and NNRT is used with EG method.

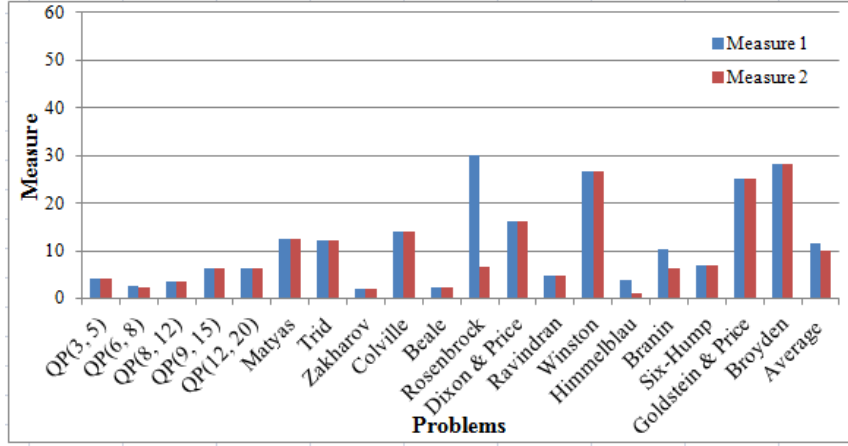
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	1.5	2.297	0.326	0.500	4.152	0.324	0.498	4.136
QP(6, 8)	0.0	-8.790	0.200	0.307	2.414	0.174	0.286	2.250
QP(8, 12)	3.1	-12.775	0.982	0.608	3.623	0.982	0.608	3.623
QP(9, 15)	4.0	-4.021	7.865	1.501	6.259	7.858	1.500	6.255
QP(12, 20)	6.0	0.174	6.232	1.455	6.368	6.229	1.455	6.369
Matyas	0.1	0.097	0.010	0.707	12.502	0.010	0.707	12.502
Trid	1.0	-34.651	5.906	7.560	12.247	5.906	7.560	12.247
Zakharov	1.0	0.046	0.046	0.103	1.977	0.045	0.102	1.959
Colville	1.0	16.331	16.339	1.114	13.925	16.285	1.114	13.927
Beale	1.0	0.011	0.011	0.274	2.422	0.010	0.274	2.423
Rosenbrock	1.0	1.088	1.355	1.271	29.953	0.023	0.280	6.593
Dixon & Price	1.8	3.429	3.428	1.075	16.023	3.428	1.075	16.023
Ravindran	0.7	-1.855	0.110	0.236	4.769	0.110	0.236	4.769
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.6	34.062	10.910	0.442	3.906	10.901	0.442	3.903
Branin	1.0	1.760	1.687	1.275	5.999	1.139	1.319	6.205
Six-Hump	1.0	0.830	1.129	0.498	7.045	1.733	0.680	9.612
Goldstein & Price	0.0	5278.000	5275.000	1.414	25.000	5275.000	1.414	25.000
Broyden	0.0	17.473	8.293	2.702	28.484	7.045	2.378	25.065
Average	1.3	278.602	281.044	1.219	11.250	280.906	1.160	9.976

Table 5.10. Results for a deterministic simulator with 2000 patterns and NNCT is used with FD method.

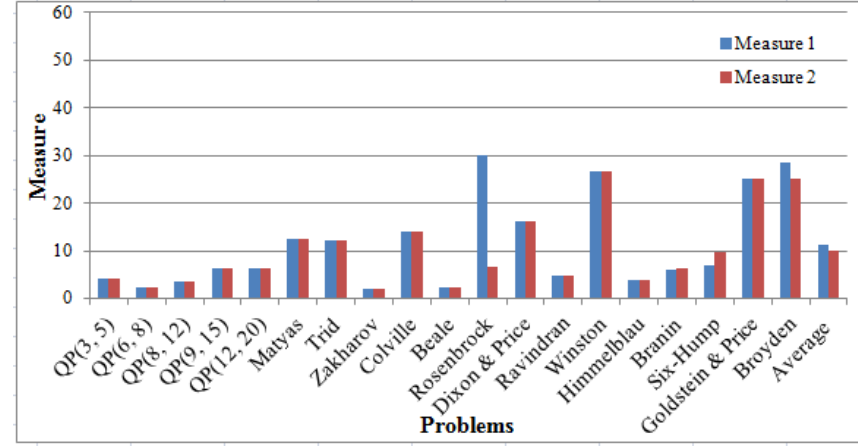
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	128.5	1.995	0.026	0.125	1.035	0.024	0.115	0.955
QP(6, 8)	18.7	-8.970	0.001	0.027	0.211	0.001	0.027	0.211
QP(8, 12)	1362.3	-13.756	0.002	0.024	0.145	0.002	0.024	0.145
QP(9, 15)	2299.1	-11.876	0.011	0.074	0.309	0.011	0.074	0.309
QP(12, 20)	4664.3	-5.792	0.267	0.316	1.383	0.267	0.316	1.383
Matyas	38.0	0.000	0.000	0.113	2.000	0.000	0.113	2.000
Trid	313.0	-49.971	0.029	0.529	0.857	0.028	0.529	0.857
Zakharov	60.8	0.010	0.006	0.055	1.057	0.001	0.034	0.648
Colville	136.0	4.039	3.464	1.767	22.089	3.455	1.767	22.089
Beale	38.0	22.626	4.997	0.393	3.471	4.997	0.393	3.471
Rosenbrock	38.3	0.812	0.046	0.059	1.402	0.037	0.059	1.386
Dixon & Price	211.1	1.439	0.980	0.517	7.708	0.978	0.517	7.704
Ravindran	30.1	-2.620	0.022	0.098	1.972	0.014	0.082	1.659
Winston	2.6	-0.071	0.000	0.013	3.000	0.000	0.013	3.000
Himmelblau	38.6	0.398	0.395	0.111	0.985	0.391	0.110	0.974
Branin	38.0	0.517	0.133	0.374	1.759	0.085	0.164	0.771
Six-Hump	38.1	-0.485	0.120	0.166	2.345	0.120	0.166	2.345
Goldstein & Price	38.0	731.522	76.700	0.238	4.206	184.958	0.515	9.105
Broyden	187.5	11.810	11.248	2.287	24.103	3.260	0.475	5.002
Average	509.5	35.875	5.181	0.383	4.212	10.454	0.289	3.369

Table 5.11. Results for a deterministic simulator with 2000 patterns and NNCT is used with EG method.

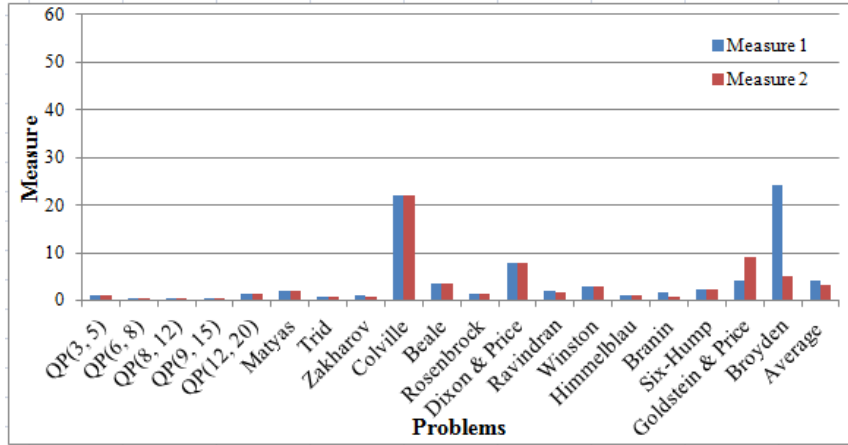
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	32.1	2.137	0.028	0.120	0.998	0.026	0.129	1.071
QP(6, 8)	19.4	-8.975	0.001	0.021	0.168	0.000	0.009	0.068
QP(8, 12)	869.2	-13.756	0.002	0.024	0.145	0.002	0.024	0.145
QP(9, 15)	1079.8	-11.876	0.011	0.074	0.309	0.011	0.074	0.309
QP(12, 20)	2419.4	-5.792	0.267	0.316	1.385	0.267	0.316	1.385
Matyas	29.7	0.000	0.000	0.113	2.000	0.000	0.113	2.000
Trid	211.4	-49.426	0.029	0.529	0.857	0.029	0.529	0.857
Zakharov	59.6	0.015	0.005	0.055	1.058	0.005	0.055	1.058
Colville	110.0	3.758	3.485	1.768	22.094	3.462	1.767	22.089
Beale	33.0	15.589	5.053	0.394	3.479	5.000	0.393	3.471
Rosenbrock	30.6	1.193	0.040	0.059	1.386	0.034	0.059	1.394
Dixon & Price	165.5	1.266	0.979	0.517	7.704	0.977	0.517	7.704
Ravindran	33.0	-2.614	0.022	0.098	1.972	0.022	0.098	1.972
Winston	2.9	-0.071	0.001	0.013	3.064	0.000	0.010	2.427
Himmelblau	34.0	0.407	0.395	0.111	0.985	0.391	0.110	0.974
Branin	30.8	0.523	0.136	0.374	1.759	0.087	0.164	0.773
Six-Hump	24.5	-0.596	0.121	0.166	2.345	0.120	0.166	2.345
Goldstein & Price	33.2	188.859	185.389	0.515	9.105	185.064	0.515	9.105
Broyden	183.9	13.057	11.139	2.256	23.780	3.259	0.475	5.012
Average	284.3	7.037	10.900	0.396	4.452	10.461	0.291	3.377



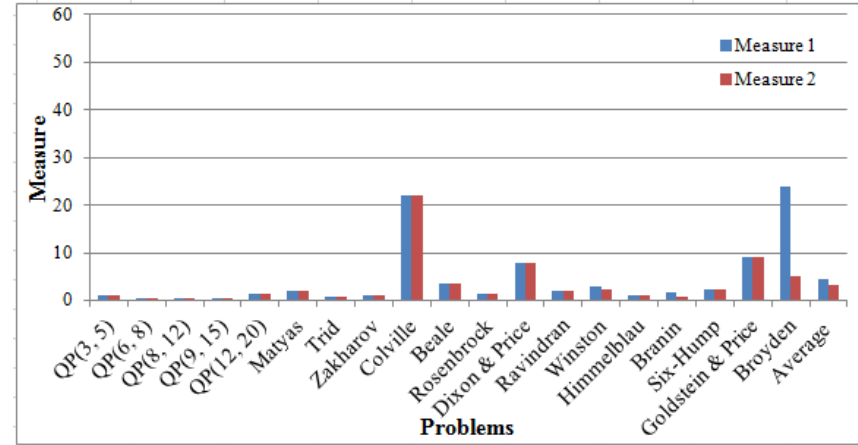
(a)



(b)



(c)



(d)

Figure 5.2. Measures for test problems using 2000 deterministic patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.

Figure 5.1 and Table 5.12 summarize Tables 5.4-5.7. Observe that since different initial points are used, the results of FD and EG are slightly different. If the algorithm was started at the same initial points, then the results would have been exactly the same. The only difference in that regard is that FD takes more CPU time. Also, the use of NNCT needs significantly more time to be executed while its accuracy is much better in terms of both of the performance measures. In short, we can say that the use of NNCT with EG is preferable in terms of accuracy and speed.

Table 5.12. Results summary for a deterministic simulator with 1000 patterns.

Network	Finite Difference			Exact Gradient		
	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
NNRT	1.4	15.142	12.250	1.1	15.667	12.055
NNCT	237.0	6.062	7.774	155.3	7.769	7.933

Furthermore, we can see that a decision based on the second criterion yields better results for NNRT. However, both performance criteria give slightly different measures for NNCT, which makes it difficult to select any of them.

Figure 5.2 and Table 5.13 summarize Tables 5.8-5.11. Since we double the number of patterns, NRG gives better results in terms of both performance criteria. We can make almost the same observations as we have done earlier. The only difference is that the second performance criterion always gives better results than the first one. We can conclude that as the accuracy increases, the second criterion outperforms the first one.

Table 5.13. Results summary for a deterministic simulator with 2000 patterns.

Network	Finite Difference			Exact Gradient		
	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
NNRT	1.6	11.474	9.859	1.3	11.250	9.976
NNCT	509.5	4.212	3.369	284.3	4.452	3.377

It is interesting to note that for Colville, NNRT gives more accurate results unexpectedly. This fact tells us that NNCT is not properly trained for this problem. Therefore, instead of using a fixed setting for the training of all problems, it may be a better idea to fine-tune parameters with respect to each problem. However, this can be a costly procedure.

5.1.3.2. Random Case. Now, let us consider a simulator with random outputs. For this purpose, we again use the same functions but allow their outcomes to deviate at most 1% from the deterministic value following a uniform random variable. Our aim here is to minimize the expected value of the simulator subject to the previous constraints.

First, assume again that the simulator is inexpensive and using FD directly is possible. Since the simulator has random outputs, we need to replicate simulation runs in order to have accurate gradient estimates. We experiment with three, five and seven replications and results are given in Tables 5.15-5.17 which are summarized in Table 5.14 and Figure 5.3.

Table 5.14. Results summary for a random, inexpensive simulator.

Replication	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
3	0.6	10.078	9.928
5	0.8	7.617	7.690
7	0.9	8.463	8.440

Observe that our performance measures get closer to each other as the number of replications increase. We can deduce that using five replications gives the best outcome in terms of both performance measures among these three alternatives. However, note that the selection of this number is only specific to the problems solved and this level of stochasticity.

Table 5.15. Solution for a random, inexpensive simulator with three replications.

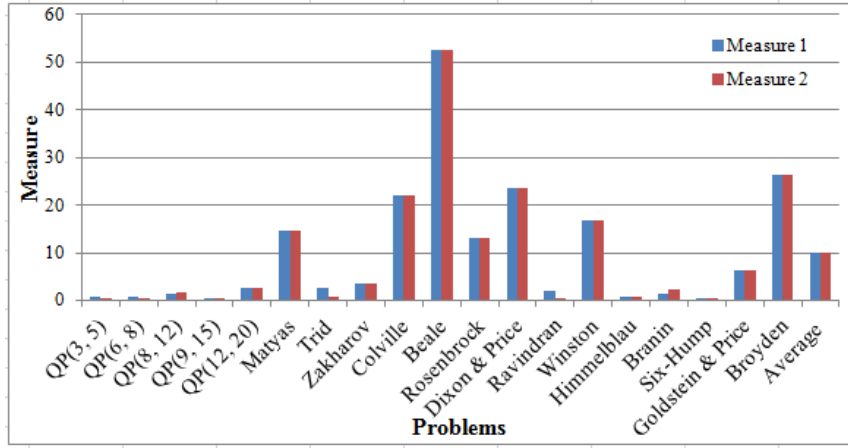
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	0.0	2.097	0.011	0.086	0.717	0.002	0.030	0.246
QP(6, 8)	1.0	-8.912	0.018	0.092	0.724	0.008	0.028	0.217
QP(8, 12)	2.0	-13.085	0.261	0.223	1.326	0.243	0.306	1.824
QP(9, 15)	2.0	-11.807	0.062	0.146	0.610	0.057	0.111	0.461
QP(12, 20)	4.0	22.184	2.732	0.622	2.724	2.732	0.622	2.724
Matyas	0.0	1.056	0.015	0.825	14.585	0.015	0.825	14.585
Trid	1.0	-49.297	0.331	1.588	2.572	0.115	0.505	0.819
Zakharov	0.0	1.768	0.056	0.184	3.548	0.056	0.184	3.548
Colville	0.0	40.777	2.297	1.761	22.008	2.297	1.761	22.008
Beale	0.0	11152.012	0.913	5.927	52.392	0.913	5.927	52.392
Rosenbrock	0.0	51.000	0.839	0.554	13.053	0.839	0.554	13.053
Dixon & Price	0.1	1.338	0.914	1.575	23.482	0.914	1.575	23.482
Ravindran	0.0	-3.690	0.021	0.096	1.937	0.002	0.028	0.569
Winston	0.0	-0.058	0.003	0.071	16.667	0.003	0.071	16.667
Himmelblau	0.0	13.680	0.297	0.088	0.776	0.297	0.088	0.776
Branin	0.0	6.866	0.383	0.319	1.499	0.378	0.512	2.409
Six-Hump	0.0	5.201	0.005	0.027	0.377	0.005	0.027	0.377
Goldstein & Price	0.0	22993.420	24.709	0.356	6.300	24.709	0.356	6.300
Broyden	1.1	2.161	1.513	2.484	26.179	1.513	2.484	26.179
Average	0.6	1800.353	1.862	0.896	10.078	1.847	0.842	9.928

Table 5.16. Solution for a random, inexpensive simulator with five replications.

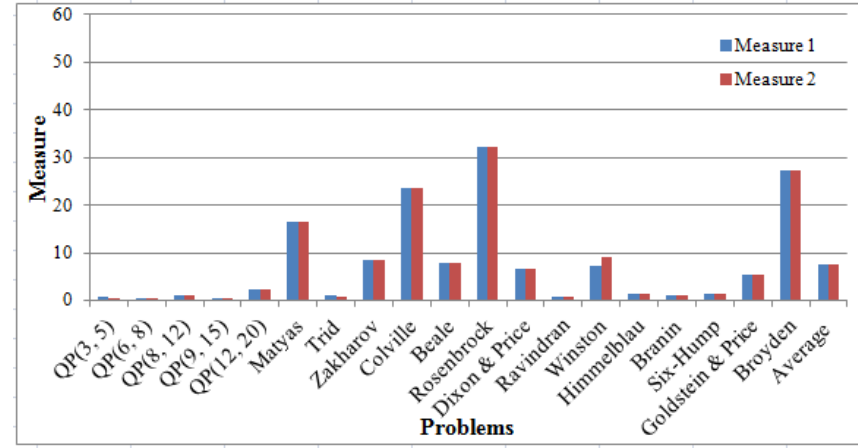
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	0.3	3.063	0.020	0.076	0.628	0.004	0.056	0.466
QP(6, 8)	1.0	-8.705	0.005	0.050	0.393	0.004	0.035	0.277
QP(8, 12)	2.0	-12.746	0.138	0.157	0.936	0.138	0.157	0.936
QP(9, 15)	3.0	-11.818	0.021	0.077	0.320	0.021	0.077	0.320
QP(12, 20)	4.2	22.971	1.509	0.493	2.159	1.509	0.493	2.159
Matyas	0.0	0.650	0.017	0.927	16.389	0.017	0.927	16.389
Trid	1.0	-48.855	0.447	0.582	0.943	0.121	0.470	0.762
Zakharov	0.0	2.149	0.249	0.434	8.357	0.249	0.434	8.357
Colville	1.0	96.623	2.726	1.883	23.531	2.726	1.883	23.531
Beale	0.0	11692.578	0.310	0.883	7.804	0.310	0.883	7.804
Rosenbrock	0.0	2.089	0.899	1.359	32.042	0.899	1.359	32.042
Dixon & Price	1.0	1.574	0.824	0.448	6.682	0.824	0.448	6.682
Ravindran	0.0	-2.231	0.003	0.039	0.788	0.003	0.039	0.788
Winston	0.0	-0.059	0.001	0.030	7.180	0.001	0.038	9.022
Himmelblau	0.0	18.625	1.094	0.168	1.486	1.094	0.168	1.486
Branin	0.0	9.731	0.100	0.247	1.162	0.100	0.247	1.162
Six-Hump	0.0	2.120	0.079	0.093	1.315	0.079	0.093	1.315
Goldstein & Price	0.0	14397.234	140.031	0.307	5.423	140.031	0.307	5.423
Broyden	2.0	4.365	1.340	2.580	27.191	1.340	2.580	27.191
Average	0.8	1377.335	7.885	0.570	7.617	7.867	0.563	7.690

Table 5.17. Solution for a random, inexpensive simulator with seven replications.

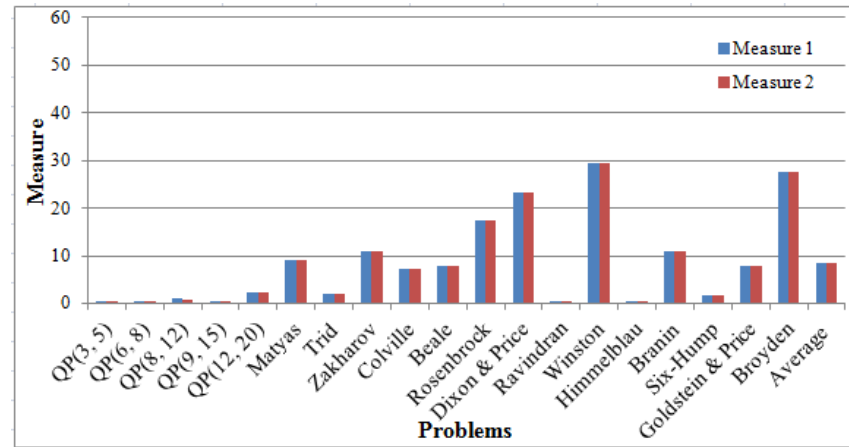
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	0.1	2.173	0.005	0.044	0.362	0.005	0.044	0.362
QP(6, 8)	1.0	-8.881	0.010	0.065	0.511	0.009	0.067	0.523
QP(8, 12)	2.0	-13.385	0.179	0.184	1.096	0.050	0.110	0.654
QP(9, 15)	3.0	-11.804	0.045	0.099	0.413	0.045	0.099	0.413
QP(12, 20)	5.0	19.458	1.323	0.550	2.408	1.323	0.550	2.408
Matyas	0.0	0.977	0.005	0.513	9.064	0.005	0.513	9.064
Trid	1.0	-47.238	0.717	1.239	2.008	0.717	1.239	2.008
Zakharov	0.0	2.061	0.317	0.562	10.809	0.317	0.562	10.809
Colville	1.0	48.822	1.068	0.585	7.308	1.068	0.585	7.308
Beale	0.0	9755.096	0.395	0.888	7.847	0.395	0.888	7.847
Rosenbrock	0.0	23.006	0.148	0.740	17.439	0.148	0.740	17.439
Dixon & Price	1.0	6.002	1.065	1.564	23.319	1.065	1.564	23.319
Ravindran	0.0	-3.365	0.005	0.023	0.461	0.005	0.023	0.461
Winston	0.0	-0.051	0.005	0.124	29.322	0.005	0.124	29.322
Himmelblau	0.0	23.244	0.178	0.065	0.579	0.178	0.065	0.579
Branin	0.0	27.612	2.403	2.307	10.852	2.403	2.307	10.852
Six-Hump	0.0	2.726	0.095	0.115	1.620	0.095	0.115	1.620
Goldstein & Price	0.0	41489.696	255.776	0.445	7.861	255.776	0.445	7.861
Broyden	3.0	2.433	1.353	2.610	27.516	1.353	2.610	27.516
Average	0.9	2700.978	13.952	0.670	8.463	13.945	0.666	8.440



(a)



(b)



(c)

Figure 5.3. Measures for test problems using finite difference: (a) three replications, (b) five replication and (c) seven replications.

Now, suppose that the simulator is expensive and using FD directly is not applicable. Assume further that we are given 1000 and 2000 feasible patterns. In order to use NRG, we train the neural networks using the parameters given in Table 5.2. Table 5.18 gives execution time needed to train 1000 and 2000 patterns if NNRT and NNCT are used. Note that this table is very similar to Table 5.3 and the same conclusions are valid.

Table 5.18. Training time of neural networks with random inputs.

Problem	1000 patterns		2000 patterns	
	NNRT	NNCT	NNRT	NNCT
QP(3, 5)	131	764	625	2617
QP(6, 8)	196	3189	491	11390
QP(8, 12)	194	5715	379	20923
QP(9, 15)	308	11055	622	44877
QP(12, 20)	450	31613	1121	92856
Matyas	168	737	293	2097
Trid	187	2142	613	4494
Zakharov	208	328	397	1839
Colville	304	403	434	2542
Beale	159	210	457	474
Rosenbrock	162	209	381	933
Dixon & Price	132	785	896	2646
Ravindran	107	158	362	474
Winston	145	158	296	628
Himmelblau	227	159	293	766
Branin	152	206	223	623
Six-Hump	148	141	293	484
Goldstein & Price	266	185	392	639
Broyden	213	812	694	3617
Average	203.0	3103.6	487.5	10258.9

We design eight experiments with the same setting as done in the previous section for the deterministic case. Results using 1000 patterns are given in Tables 5.19-5.22 whereas results using 2000 patterns are given in Tables 5.23-5.26.

Table 5.19. Results for a random simulator with 1000 patterns and NNRT is used with FD method.

Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	1.0	2.562	0.595	0.570	4.731	0.593	0.571	4.741
QP(6, 8)	0.0	-7.481	1.578	0.861	6.767	1.205	0.753	5.915
QP(8, 12)	0.5	-13.248	0.491	0.396	2.359	0.491	0.396	2.359
QP(9, 15)	5.0	-2.329	9.544	1.595	6.649	9.544	1.595	6.649
QP(12, 20)	8.3	-0.524	5.534	1.400	6.129	5.525	1.399	6.124
Matyas	0.3	0.082	0.010	0.707	12.502	0.010	0.707	12.502
Trid	1.4	-47.457	2.546	2.712	4.393	2.539	2.711	4.392
Zakharov	0.9	0.185	0.131	0.358	6.893	0.131	0.358	6.893
Colville	2.0	13.411	13.021	3.127	39.093	9.450	2.006	25.076
Beale	1.0	16.521	2.097	0.824	7.282	2.080	0.823	7.273
Rosenbrock	1.0	7.121	16.779	2.032	47.886	0.693	1.261	29.721
Dixon & Price	1.0	3.766	3.766	1.032	15.389	3.765	1.032	15.389
Ravindran	1.0	-2.426	0.316	0.376	7.604	0.315	0.376	7.604
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.7	45.743	47.448	1.008	8.913	13.385	0.574	5.077
Branin	0.4	15.254	18.033	2.009	9.449	2.154	2.217	10.427
Six-Hump	1.0	0.549	1.582	0.771	10.904	1.580	0.772	10.915
Goldstein & Price	1.0	3927.523	3906.341	1.372	24.250	3890.553	1.370	24.225
Broyden	0.0	21.849	32.454	1.785	18.817	8.828	2.097	22.109
Average	1.4	209.528	213.804	1.213	14.036	208.045	1.112	12.319

Table 5.20. Results for a random simulator with 1000 patterns and NNRT is used with EG method.

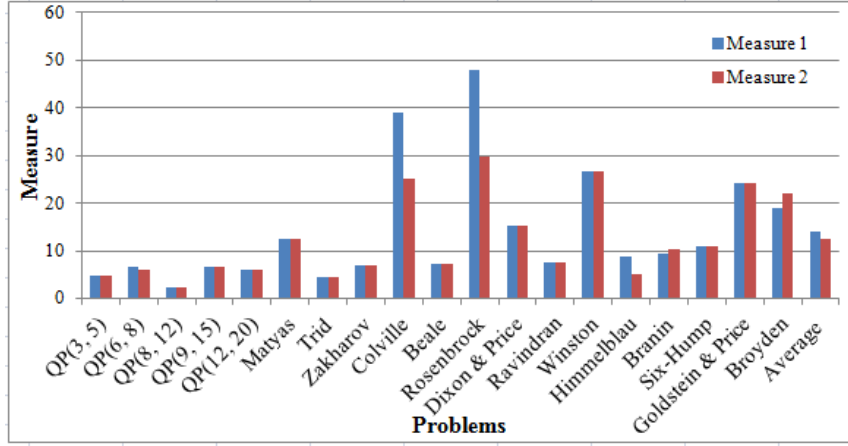
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	0.5	2.567	0.595	0.569	4.724	0.593	0.571	4.741
QP(6, 8)	0.0	-7.433	1.578	0.861	6.767	1.491	0.838	6.580
QP(8, 12)	0.4	-13.259	0.497	0.397	2.367	0.493	0.397	2.363
QP(9, 15)	3.6	-2.334	9.544	1.594	6.648	9.544	1.594	6.648
QP(12, 20)	6.3	-0.524	5.538	1.401	6.131	5.526	1.399	6.124
Matyas	0.2	0.012	0.010	0.707	12.502	0.010	0.707	12.502
Trid	1.1	-47.459	2.546	2.711	4.393	2.539	2.711	4.392
Zakharov	1.0	0.126	0.131	0.358	6.893	0.072	0.253	4.868
Colville	1.4	14.284	13.035	3.128	39.099	13.021	3.127	39.093
Beale	1.0	34.995	47.540	3.902	34.490	2.077	0.822	7.266
Rosenbrock	1.0	5.505	16.738	2.032	47.886	0.693	1.261	29.721
Dixon & Price	1.0	3.766	3.766	1.032	15.389	3.766	1.032	15.389
Ravindran	1.0	-2.904	0.315	0.376	7.604	0.315	0.376	7.604
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	0.8	62.918	47.448	1.008	8.913	13.385	0.574	5.077
Branin	0.7	10.965	18.938	2.107	9.909	2.154	2.216	10.423
Six-Hump	1.0	0.549	1.580	0.772	10.915	1.579	0.772	10.912
Goldstein & Price	1.0	1440.002	3908.505	1.372	24.250	368.326	0.602	10.635
Broyden	0.2	37.508	147.946	2.244	23.655	7.612	2.490	26.249
Average	1.2	81.012	222.435	1.405	15.748	22.800	1.150	12.488

Table 5.21. Results for a random simulator with 1000 patterns and NNCT is used with FD method.

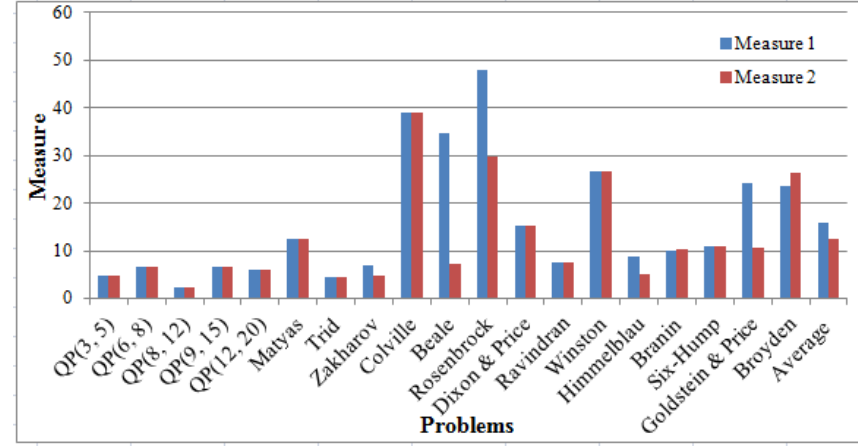
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	63.8	1.987	0.019	0.103	0.856	0.019	0.103	0.856
QP(6, 8)	30.6	-8.976	0.000	0.011	0.089	0.000	0.011	0.089
QP(8, 12)	540.0	-13.754	0.007	0.052	0.310	0.002	0.024	0.142
QP(9, 15)	1191.6	-11.882	0.006	0.041	0.172	0.006	0.041	0.172
QP(12, 20)	2478.3	-5.747	0.312	0.313	1.372	0.312	0.313	1.372
Matyas	21.0	0.000	0.000	0.038	0.675	0.000	0.038	0.675
Trid	157.2	-49.881	0.119	1.082	1.752	0.118	1.081	1.751
Zakharov	37.1	0.002	0.002	0.024	0.461	0.002	0.024	0.461
Colville	69.0	6.730	6.412	0.529	6.619	6.330	2.371	29.633
Beale	19.5	18.775	13.432	0.493	4.359	13.378	0.492	4.350
Rosenbrock	16.0	1.388	0.106	0.758	17.877	0.106	0.758	17.877
Dixon & Price	107.5	1.282	0.863	0.786	11.718	0.863	0.786	11.718
Ravindran	20.2	-2.561	0.019	0.099	2.001	0.019	0.099	2.001
Winston	0.3	-0.071	0.001	0.030	7.075	0.001	0.030	7.075
Himmelblau	19.5	0.759	0.861	0.164	1.447	0.187	0.061	0.542
Branin	19.9	0.675	0.240	0.284	1.335	0.235	0.284	1.335
Six-Hump	18.0	1.621	1.579	0.549	7.766	1.572	0.548	7.746
Goldstein & Price	20.0	1515.313	156.147	0.321	5.680	155.164	0.320	5.665
Broyden	49.0	9.631	9.355	2.107	22.210	9.355	2.107	22.210
Average	256.8	77.121	9.973	0.410	4.935	9.877	0.500	6.088

Table 5.22. Results for a random simulator with 1000 patterns and NNCT is used with EG method.

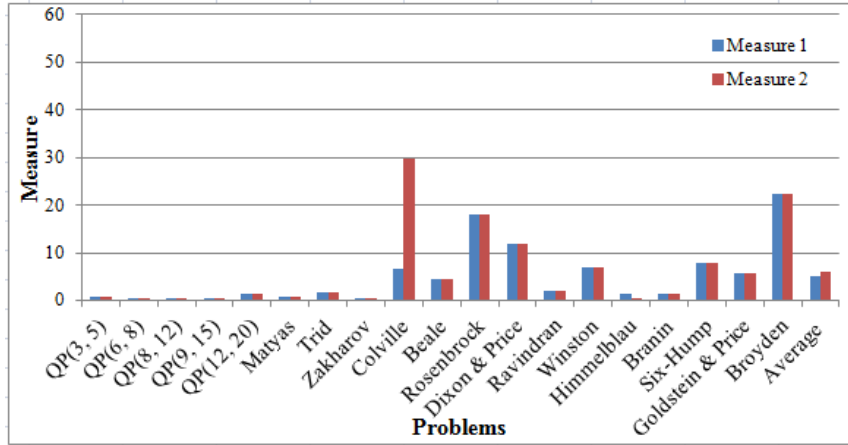
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	68.5	1.987	0.019	0.104	0.862	0.018	0.108	0.899
QP(6, 8)	22.3	-8.976	0.000	0.011	0.089	0.000	0.011	0.089
QP(8, 12)	453.1	-13.755	0.003	0.032	0.191	0.003	0.032	0.191
QP(9, 15)	703.6	-11.882	0.006	0.041	0.173	0.006	0.041	0.173
QP(12, 20)	1301.5	-5.746	0.312	0.313	1.370	0.312	0.313	1.370
Matyas	16.3	0.000	0.000	0.039	0.688	0.000	0.039	0.688
Trid	118.5	-49.881	0.119	1.082	1.752	0.119	1.082	1.752
Zakharov	23.6	0.055	0.002	0.024	0.461	0.002	0.024	0.461
Colville	55.6	6.361	6.365	0.529	6.610	6.338	2.371	29.633
Beale	17.0	13.331	13.412	0.492	4.350	12.576	6.467	57.165
Rosenbrock	17.0	1.737	4.008	2.000	47.141	0.220	0.793	18.690
Dixon & Price	63.2	1.546	0.864	0.786	11.720	0.863	0.786	11.718
Ravindran	16.2	-2.846	0.019	0.099	2.001	0.019	0.099	2.001
Winston	0.2	-0.071	0.001	0.031	7.307	0.001	0.031	7.307
Himmelblau	17.0	0.780	0.861	0.164	1.447	0.176	0.060	0.529
Branin	17.0	0.674	0.235	0.284	1.335	0.234	0.283	1.332
Six-Hump	14.4	1.274	1.580	0.549	7.766	0.401	0.220	3.112
Goldstein & Price	17.0	977.109	155.843	0.321	5.680	467.649	0.628	11.100
Broyden	95.1	9.439	9.356	2.108	22.217	9.346	2.103	22.168
Average	159.8	48.481	10.158	0.474	6.482	26.225	0.815	8.967



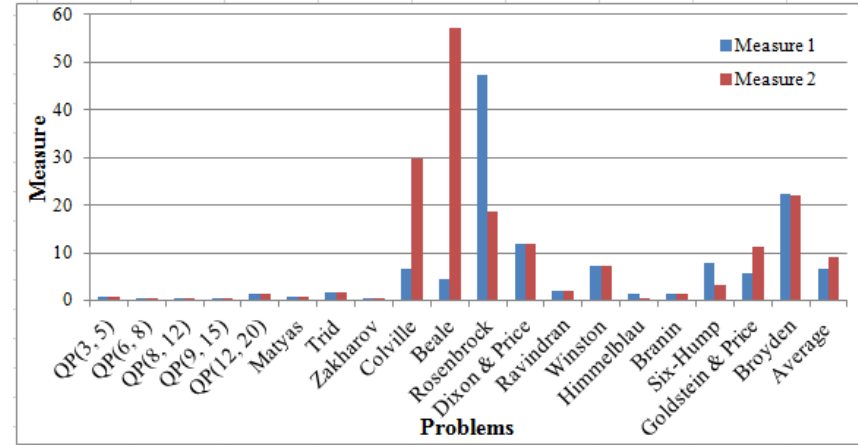
(a)



(b)



(c)



(d)

Figure 5.4. Measures for test problems using 1000 random patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.

Table 5.23. Results for a random simulator with 2000 patterns and NNRT is used with FD method.

Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	1.5	2.297	0.328	0.501	4.163	0.325	0.500	4.152
QP(6, 8)	0.0	-8.800	0.193	0.302	2.374	0.158	0.273	2.143
QP(8, 12)	3.5	-12.760	0.996	0.616	3.670	0.995	0.616	3.668
QP(9, 15)	5.0	-6.966	4.920	1.100	4.588	4.916	1.099	4.583
QP(12, 20)	8.7	-2.807	3.251	1.119	4.898	3.250	1.118	4.896
Matyas	0.3	0.043	0.010	0.707	12.502	0.010	0.707	12.502
Trid	1.1	-42.616	7.385	4.254	6.892	7.381	4.254	6.891
Zakharov	1.0	0.047	0.047	0.104	2.001	0.047	0.104	2.001
Colville	1.1	24.099	54.362	3.708	46.351	16.480	1.111	13.881
Beale	1.0	0.073	0.073	0.261	2.309	0.069	0.260	2.301
Rosenbrock	1.0	2.118	2.005	1.361	32.078	0.015	0.258	6.073
Dixon & Price	2.0	4.602	3.378	1.066	15.896	3.378	1.066	15.896
Ravindran	0.9	-2.591	0.058	0.165	3.338	0.058	0.165	3.338
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	1.0	31.704	10.880	0.441	3.894	0.283	0.126	1.110
Branin	0.9	2.561	9.660	2.192	10.312	1.144	1.322	6.219
Six-Hump	0.9	0.877	1.147	0.505	7.147	1.672	0.662	9.365
Goldstein & Price	0.0	5278.000	5275.000	1.414	25.000	5275.000	1.414	25.000
Broyden	0.0	9.343	7.033	2.384	25.132	7.033	2.384	25.132
Average	1.6	277.851	283.197	1.175	12.591	280.117	0.924	9.254

Table 5.24. Results for a random simulator with 2000 patterns and NNRT is used with EG method.

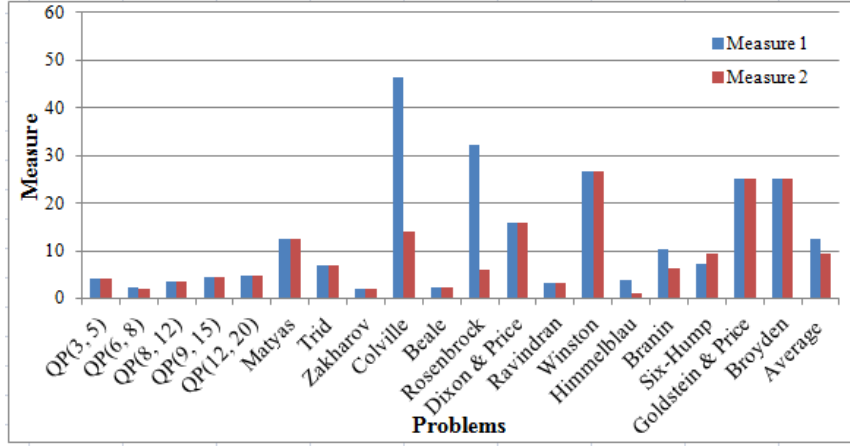
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	1.4	2.297	0.326	0.500	4.152	0.326	0.500	4.152
QP(6, 8)	0.0	-8.794	0.177	0.290	2.275	0.151	0.267	2.097
QP(8, 12)	3.0	-12.761	0.995	0.616	3.668	0.995	0.616	3.668
QP(9, 15)	1.8	-6.949	4.921	1.100	4.589	4.872	1.097	4.573
QP(12, 20)	6.0	-2.808	3.253	1.119	4.900	3.247	1.118	4.895
Matyas	0.6	0.017	0.010	0.707	12.502	0.010	0.707	12.502
Trid	1.0	-42.613	7.384	4.253	6.890	7.381	4.253	6.891
Zakharov	1.0	0.047	0.047	0.104	2.001	0.047	0.104	2.001
Colville	1.0	24.099	54.338	3.708	46.351	16.477	1.111	13.881
Beale	1.0	3.671	0.076	0.261	2.309	0.069	0.260	2.301
Rosenbrock	1.0	2.514	1.995	1.361	32.076	1.990	1.361	32.076
Dixon & Price	1.5	3.379	3.379	1.066	15.896	3.378	1.066	15.896
Ravindran	0.8	-2.589	0.058	0.164	3.318	0.058	0.164	3.318
Winston	0.0	-0.063	0.009	0.113	26.676	0.009	0.113	26.676
Himmelblau	1.0	33.830	10.901	0.442	3.903	10.881	0.441	3.894
Branin	1.0	1.508	0.204	0.194	0.910	0.204	0.194	0.910
Six-Hump	0.8	0.475	1.155	0.507	7.174	0.029	0.085	1.201
Goldstein & Price	0.0	5278.000	5275.000	1.414	25.000	5275.000	1.414	25.000
Broyden	0.0	11.669	4.768	2.300	24.248	4.768	2.300	24.248
Average	1.2	278.154	282.579	1.064	12.044	280.521	0.904	10.009

Table 5.25. Results for a random simulator with 2000 patterns and NNCT is used with FD method.

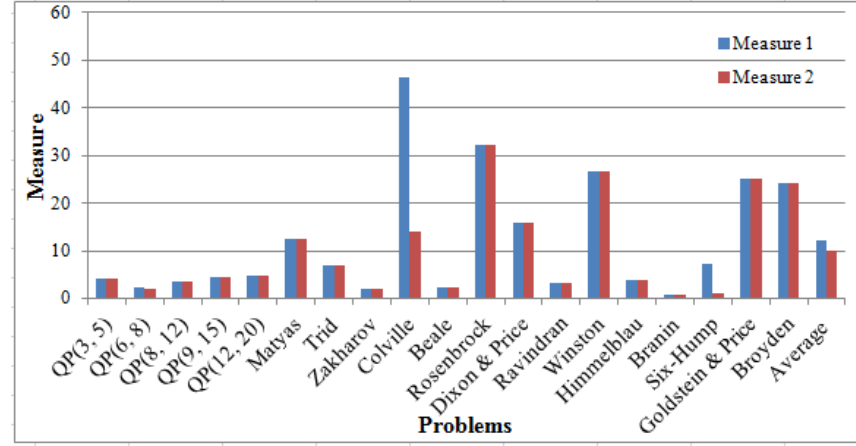
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
QP(3, 5)	183.7	1.994	0.025	0.119	0.987	0.024	0.119	0.990
QP(6, 8)	10.8	-8.976	0.000	0.020	0.157	0.000	0.020	0.157
QP(8, 12)	1357.6	-13.756	0.003	0.032	0.192	0.002	0.032	0.191
QP(9, 15)	2298.5	-11.881	0.007	0.054	0.226	0.007	0.054	0.226
QP(12, 20)	4665.1	-5.782	0.277	0.317	1.388	0.277	0.317	1.388
Matyas	39.0	0.000	0.000	0.025	0.451	0.000	0.025	0.451
Trid	306.0	-49.943	0.057	0.665	1.077	0.056	0.663	1.073
Zakharov	70.5	0.014	0.006	0.040	0.774	0.006	0.040	0.774
Colville	136.0	6.510	3.467	1.770	22.126	3.458	1.770	22.126
Beale	38.0	22.432	5.378	0.394	3.483	5.295	0.393	3.470
Rosenbrock	30.5	0.111	0.107	0.518	12.212	0.103	0.518	12.200
Dixon & Price	211.0	1.335	0.965	0.499	7.440	0.965	0.499	7.440
Ravindran	34.4	-2.624	0.019	0.090	1.812	0.019	0.090	1.812
Winston	4.3	-0.071	0.001	0.013	3.073	0.000	0.010	2.369
Himmelblau	38.1	0.393	0.391	0.110	0.974	0.387	0.109	0.963
Branin	38.1	0.528	0.128	0.366	1.722	0.127	0.366	1.722
Six-Hump	38.2	0.131	1.164	0.736	10.410	1.162	0.734	10.378
Goldstein & Price	34.7	1239.028	75.971	0.237	4.192	75.969	0.237	4.192
Broyden	186.8	13.426	14.919	2.403	25.326	3.969	0.498	5.245
Average	511.6	62.783	5.415	0.443	5.159	4.833	0.342	4.061

Table 5.26. Results for a random simulator with 2000 patterns and NNCT is used with EG method.

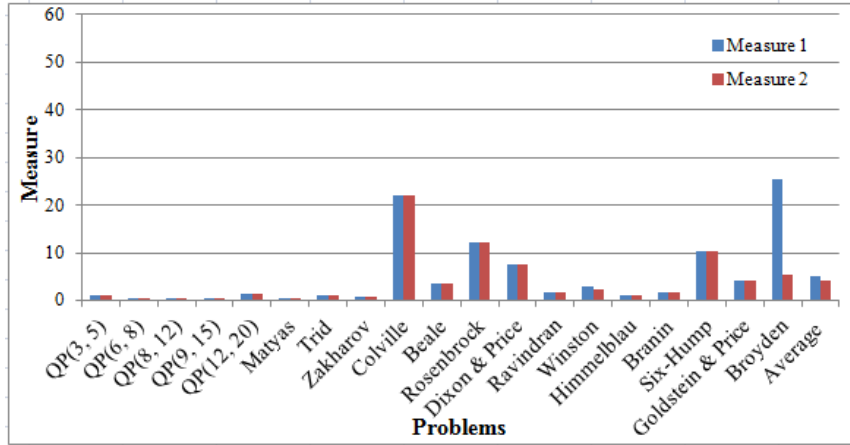
Problem	$\overline{\text{CPU}}(\text{s})$	$\overline{F(\mathbf{x})}$	with respect to fitted function			with respect to actual function		
			$F(\mathbf{x}_1) - F(\mathbf{x}_1^N)$	$\ \mathbf{x}_1 - \mathbf{x}_1^N\ $	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{1}\ }$	$F(\mathbf{x}_2) - F(\mathbf{x}_2^N)$	$\ \mathbf{x}_2 - \mathbf{x}_2^N\ $	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{1}\ }$
QP(3, 5)	147.8	1.994	0.026	0.118	0.983	0.025	0.119	0.987
QP(6, 8)	13.5	-8.976	0.000	0.017	0.130	0.000	0.017	0.130
QP(8, 12)	872.2	-13.756	0.002	0.032	0.191	0.002	0.032	0.191
QP(9, 15)	1352.0	-11.881	0.007	0.055	0.231	0.007	0.055	0.231
QP(12, 20)	2424.7	-5.782	0.277	0.317	1.388	0.277	0.317	1.388
Matyas	34.0	0.000	0.000	0.022	0.388	0.000	0.022	0.388
Trid	230.0	-49.943	0.057	0.665	1.077	0.057	0.665	1.077
Zakharov	52.2	0.114	0.006	0.040	0.774	0.006	0.040	0.774
Colville	110.0	5.081	3.471	1.770	22.126	3.455	1.770	22.126
Beale	33.0	16.440	5.363	0.394	3.487	5.285	0.393	3.470
Rosenbrock	33.6	0.107	0.112	0.517	12.191	0.102	0.518	12.200
Dixon & Price	164.0	1.230	0.966	0.499	7.440	0.965	0.499	7.440
Ravindran	26.6	-2.611	0.017	0.088	1.785	0.017	0.088	1.785
Winston	2.5	-0.071	0.001	0.025	5.911	0.001	0.025	5.911
Himmelblau	34.0	0.389	0.387	0.109	0.964	0.387	0.109	0.964
Branin	33.0	0.529	0.132	0.367	1.726	0.128	0.367	1.726
Six-Hump	31.3	0.122	1.162	0.734	10.378	1.076	0.506	7.155
Goldstein & Price	33.5	843.102	75.971	0.238	4.200	184.275	0.514	9.090
Broyden	304.8	10.963	15.595	2.554	26.917	3.965	0.498	5.248
Average	312.2	41.424	5.450	0.451	5.383	10.528	0.345	4.331



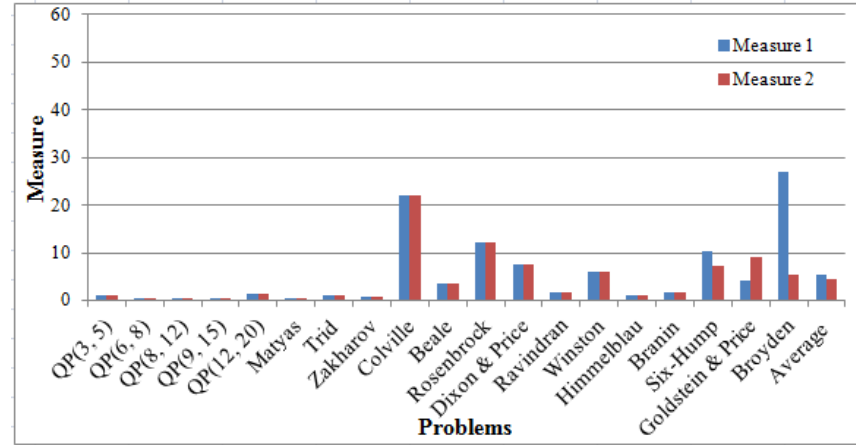
(a)



(b)



(c)



(d)

Figure 5.5. Measures for test problems using 2000 random patterns: (a) NNRT with FD, (b) NNRT with EG, (c) NNCT with FD and (d) NNCT with EG.

Figure 5.4 and Table 5.27 summarize Tables 5.19-5.22 and is quite similar to Table 5.12. Hence, almost the same conclusions can be drawn such as NNCT outperforms NNRT in terms of accuracy and EG is better than FD in terms of speed. On the other hand, there is an interesting difference that if NNCT is used and the accuracy is measured in terms of the first criterion, the results obtained using random patterns are slightly better than the ones with deterministic patterns.

Table 5.27. Results summary for a random simulator with 1000 patterns.

Network	Finite Difference			Exact Gradient		
	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
NNRT	1.4	14.036	12.319	1.2	15.748	12.488
NNCT	256.8	4.935	6.088	159.8	6.482	8.967

Figure 5.5 and Table 5.28 summarize Tables 5.23-5.26 and is similar to 5.13. Therefore, it is possible to infer the same conclusions.

Table 5.28. Results summary for a random simulator with 2000 patterns.

Network	Finite Difference			Exact Gradient		
	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$\overline{\text{CPU}}(\text{s})$	$100 \frac{\ \mathbf{x}_1 - \mathbf{x}_1^N\ }{\ \mathbf{u} - \mathbf{l}\ }$	$100 \frac{\ \mathbf{x}_2 - \mathbf{x}_2^N\ }{\ \mathbf{u} - \mathbf{l}\ }$
NNRT	1.6	12.591	9.254	1.2	12.044	10.009
NNCT	511.6	5.159	4.061	312.2	5.383	4.331

Another important comparison is between Table 5.14 and the table above. In terms of accuracy measured by our performance criteria, it is beneficial to apply NRG even if the simulator is inexpensive provided that NNCT is used. Also, NNCT trained with 2000 patterns outperforms FD with five replications for every function except Broyden in terms of average actual objective value and this fact shows its stability as well. Hence, we can argue that constructing a global approximator can give better results than using a local estimator.

5.2. A Real Application: Drag Reduction

In this section, we present the results of our experiments on drag reduction. Although we carry out more than 20 experiments, only the most relevant results are presented here. Recall that we use two methodologies in the estimation of objective function of AMP, namely CLSE and neural networks. The corresponding experiments for the former are given in Sections 5.2.1-5.2.7 whereas Sections 5.2.8-5.2.11 are devoted to the latter.

We have some general observations about the experiments. First of all, we force the total energy requirement to be at least as much as the reference, which is 251.28. This is guaranteed either by the type of estimation such as CLE and CQCE or by the selection of negative deviation parameter as $\alpha_t^- \leq 0$, for $t \in \mathbf{T}$. This is necessary since it is only possible to outperform LEGI's *ad hoc* control simulator by using more energy.

All of the parameters are selected in such a way that they increase in absolute terms as time period t gets larger. This is beneficial since any radical deviations from the reference in the early phase of the simulation cannot be recovered later due to the inherent dynamic structure. Hence, we decide to gradually change the flow values as suggested [39].

Once any shear stress reduction is obtained, another simulation is carried out by selecting a larger feasible solution space. We continue in this manner until the shear stress reduction amount begins to decrease. Empirically, we can conjecture that there is an optimal amount of energy value using which the simulator gives the best outcome.

In each of the following sections, the estimation type, the choice of energy and deviation parameters are explained. Then, shear stress estimate, energy usage and actual shear stress graphs are given. Unfortunately, none of the estimation methods give reasonable shear stress estimate graphs but they are still given for the first seven experiments. We skip these graphs for other experiments.

5.2.1. Experiment 1

In this experiment, CLE is used and E is set to 600. The other parameters are taken as $\alpha_t^+ = \alpha_t^- = \frac{t}{T}$, for $t \in \mathbf{T}$.

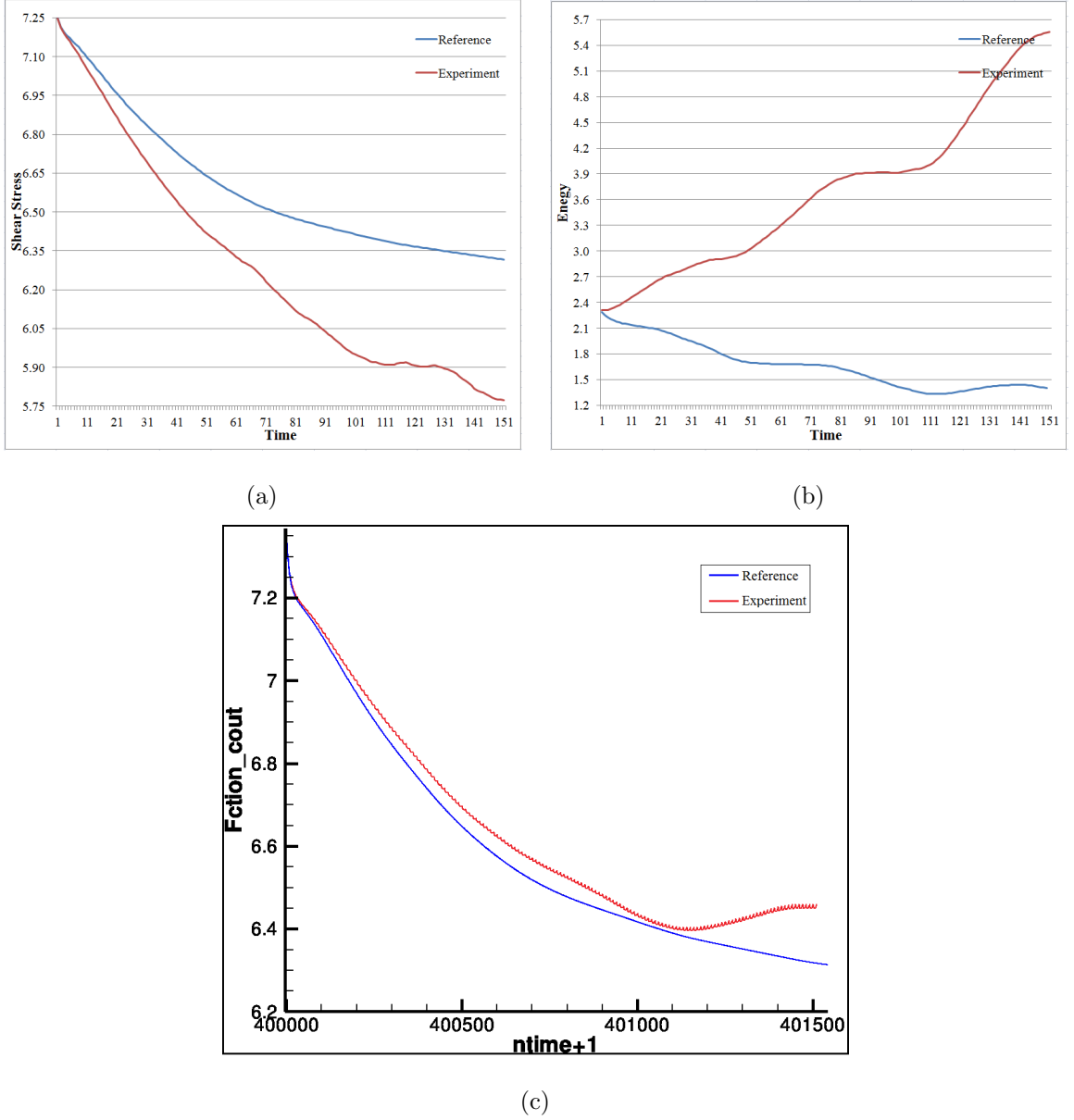


Figure 5.6. Graphs for Experiment 1: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

The total energy usage in this experiment is 555.39. As it can be seen from Figure 5.6c, shear stress reduction is not achieved at the end.

5.2.2. Experiment 2

In this experiment, CQCE is used and E is set to 300. The other parameters are taken as $\alpha_t^+ = \alpha_t^- = 0.7 \frac{t}{T}$, for $t \in T$.

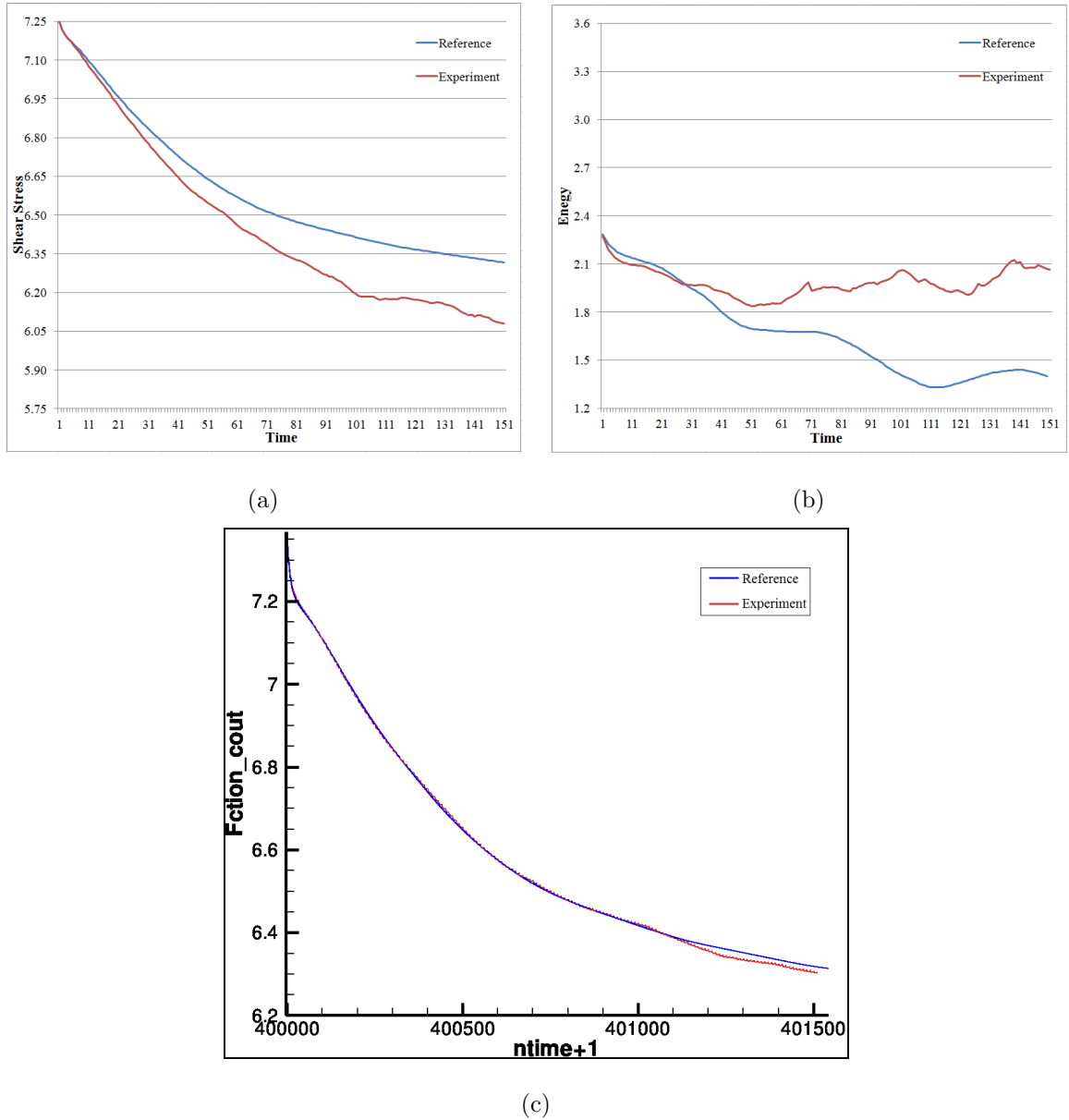


Figure 5.7. Graphs for Experiment 2: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

In this experiment, the total energy usage is 300 and at the end, about 0.1% shear stress reduction is obtained.

5.2.3. Experiment 3

In this experiment, PLR with single data set is used and AMP is solved by Log. The parameters are taken as $\alpha_t^+ = \alpha_t^- = 0.7 \frac{t}{T}$, for $t \in \mathbf{T}$.

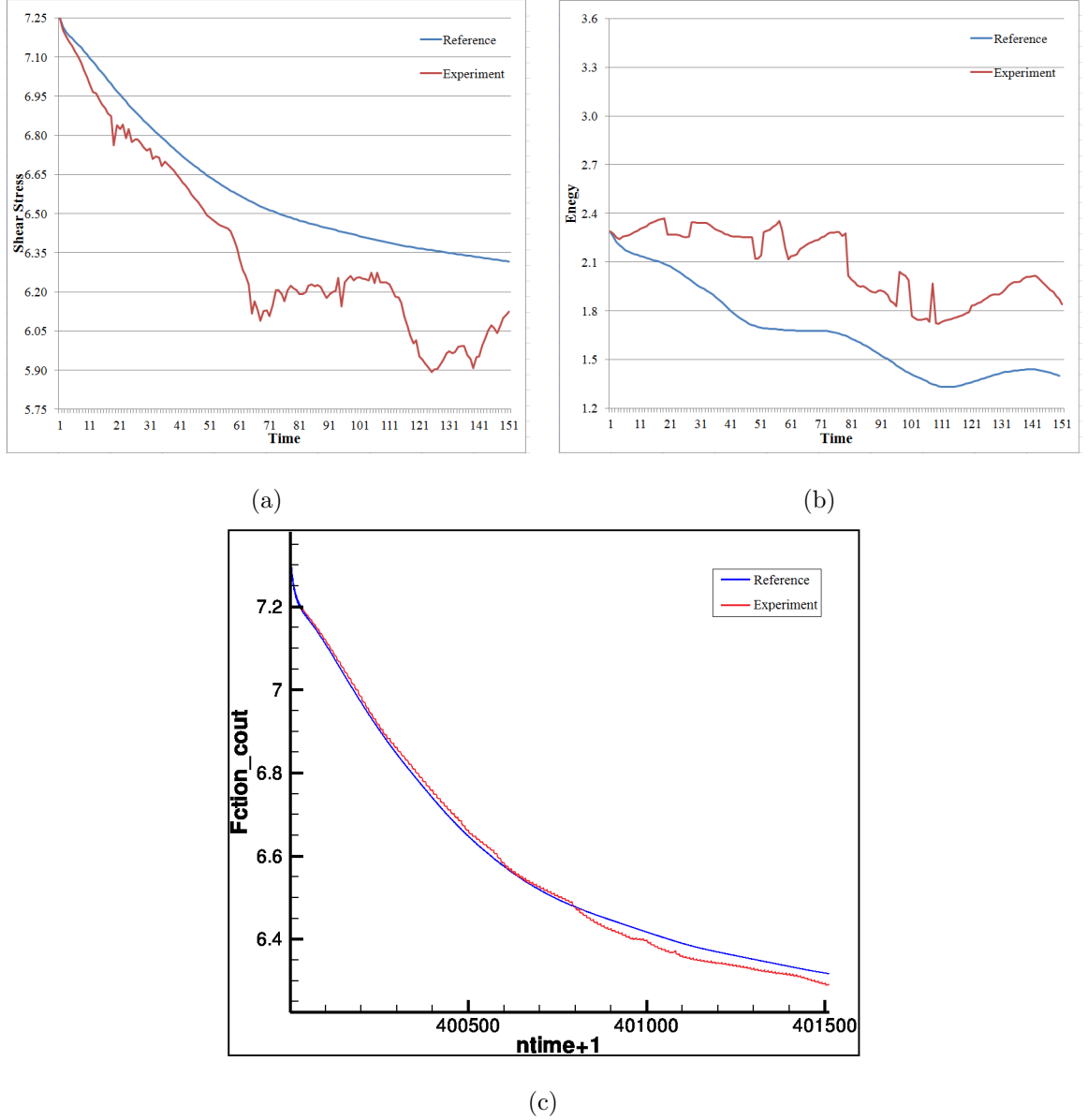


Figure 5.8. Graphs for Experiment 3: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

In this experiment, the total energy usage is 315.15 and at the end, 0.39% shear stress reduction is obtained.

5.2.4. Experiment 4

In this experiment, PLR with three data sets is used and AMP is solved by CC. The parameters are taken as $\alpha_t^+ = 0.7\frac{t}{T}$ and $\alpha_t^- = -0.1\frac{t}{T}$, for $t \in T$.

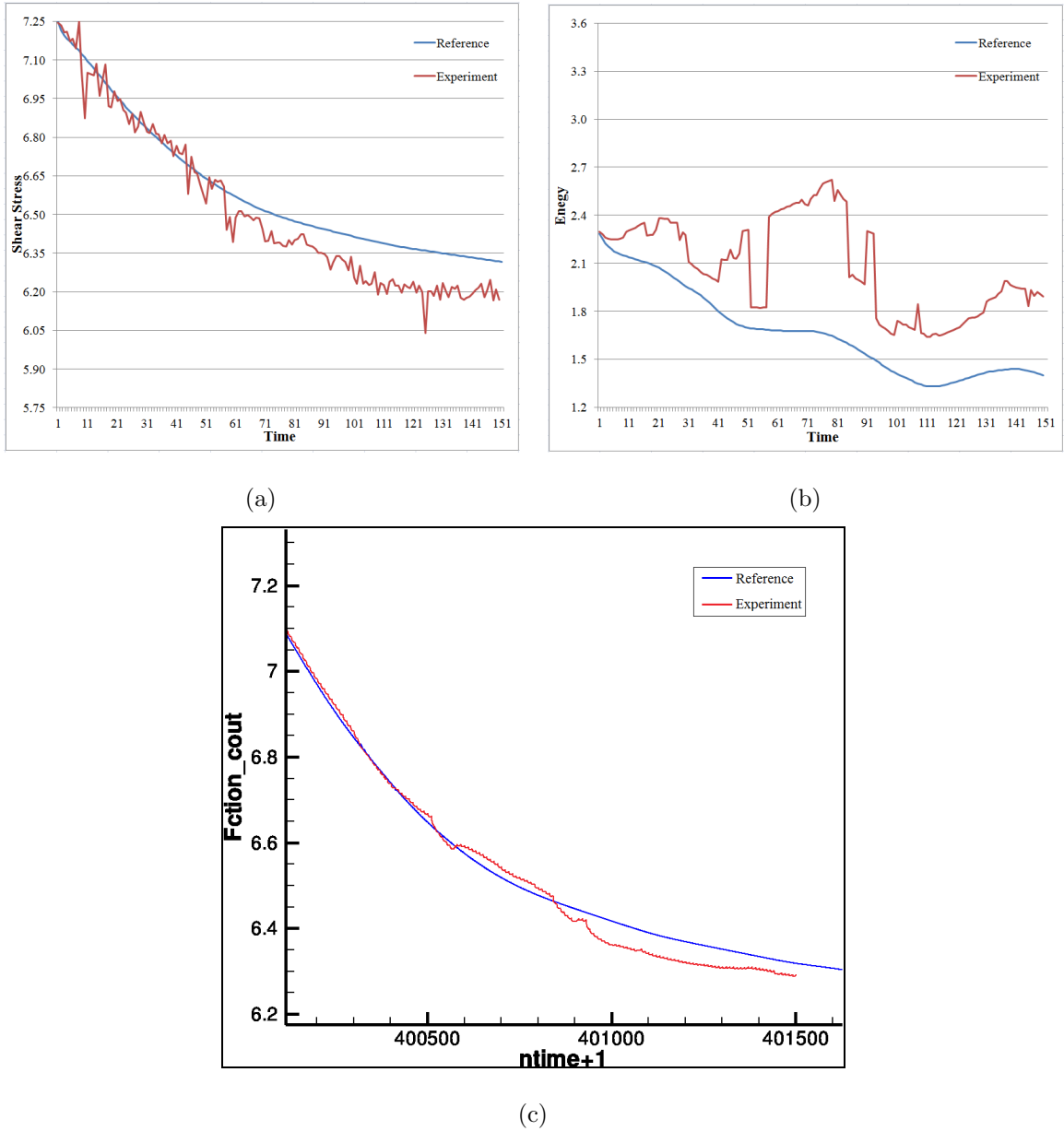


Figure 5.9. Graphs for Experiment 4: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

In this experiment, the total energy usage is 312.17 and at the end, 0.44% shear stress reduction is obtained.

5.2.5. Experiment 5

In this experiment, PLR with three data sets is used and AMP is solved by CC. The parameters are taken as $\alpha_t^+ = 0.05 + 0.7\frac{t}{T}$ and $\alpha_t^- = -0.15\frac{t}{T}$, for $t \in \mathbf{T}$.

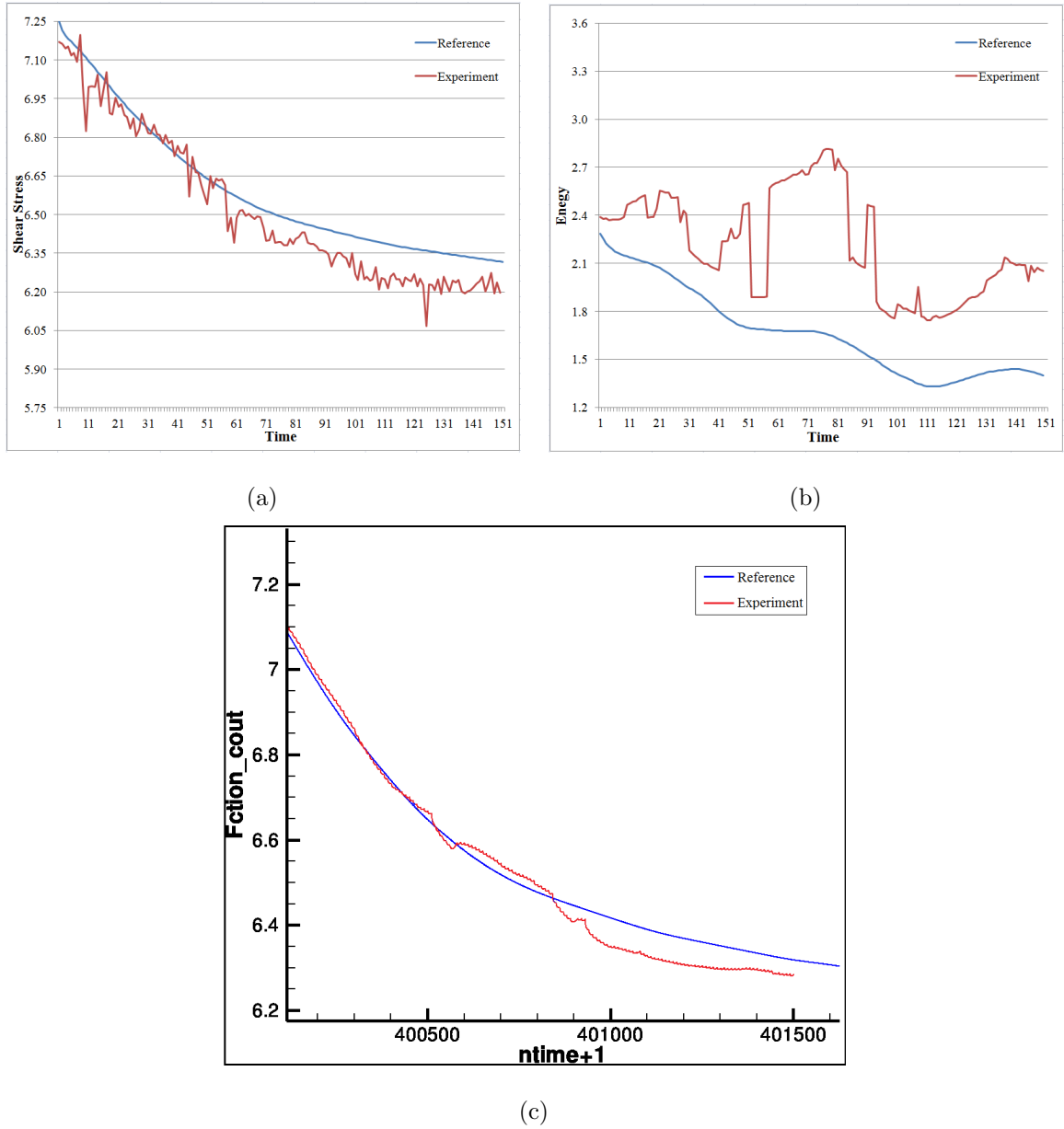


Figure 5.10. Graphs for Experiment 5: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

In this experiment, the total energy usage is 332.14 and at the end, 0.55% shear stress reduction is obtained.

5.2.6. Experiment 6

In this experiment, PLR with three data sets is used and AMP is solved by CC. The parameters are taken as $\alpha_t^+ = 0.1 + 0.7\frac{t}{T}$ and $\alpha_t^- = -0.2\frac{t}{T}$, for $t \in \mathbf{T}$.

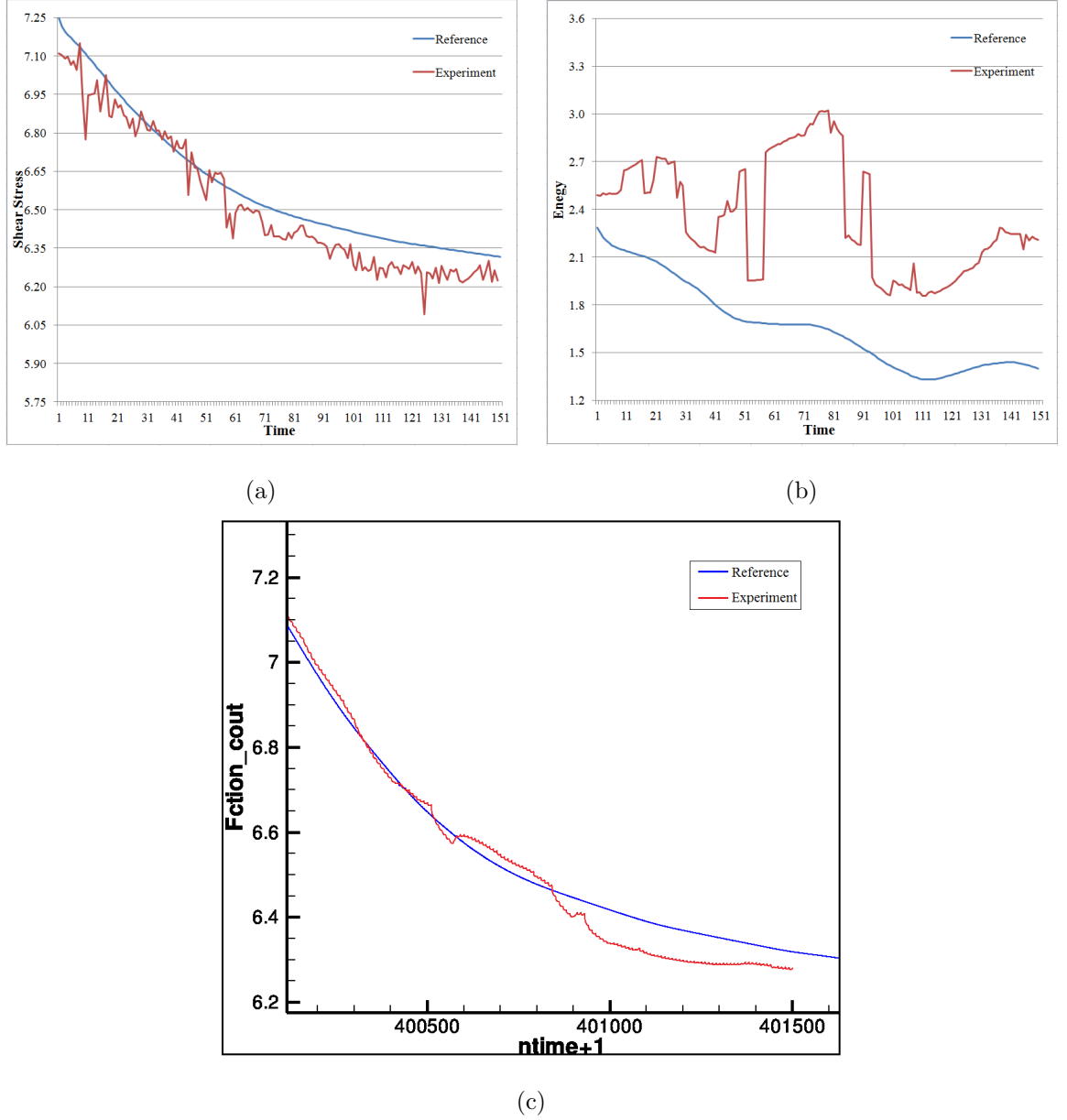


Figure 5.11. Graphs for Experiment 6: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

In this experiment, the total energy usage is 353.04 and at the end, 0.61% shear stress reduction is obtained.

5.2.7. Experiment 7

In this experiment, PLR with three data sets is used and AMP is solved by CC. The parameters are taken as $\alpha_t^+ = 0.25 + 0.65\frac{t}{T}$ and $\alpha_t^- = -0.2\frac{t}{T}$, for $t \in \mathbf{T}$.

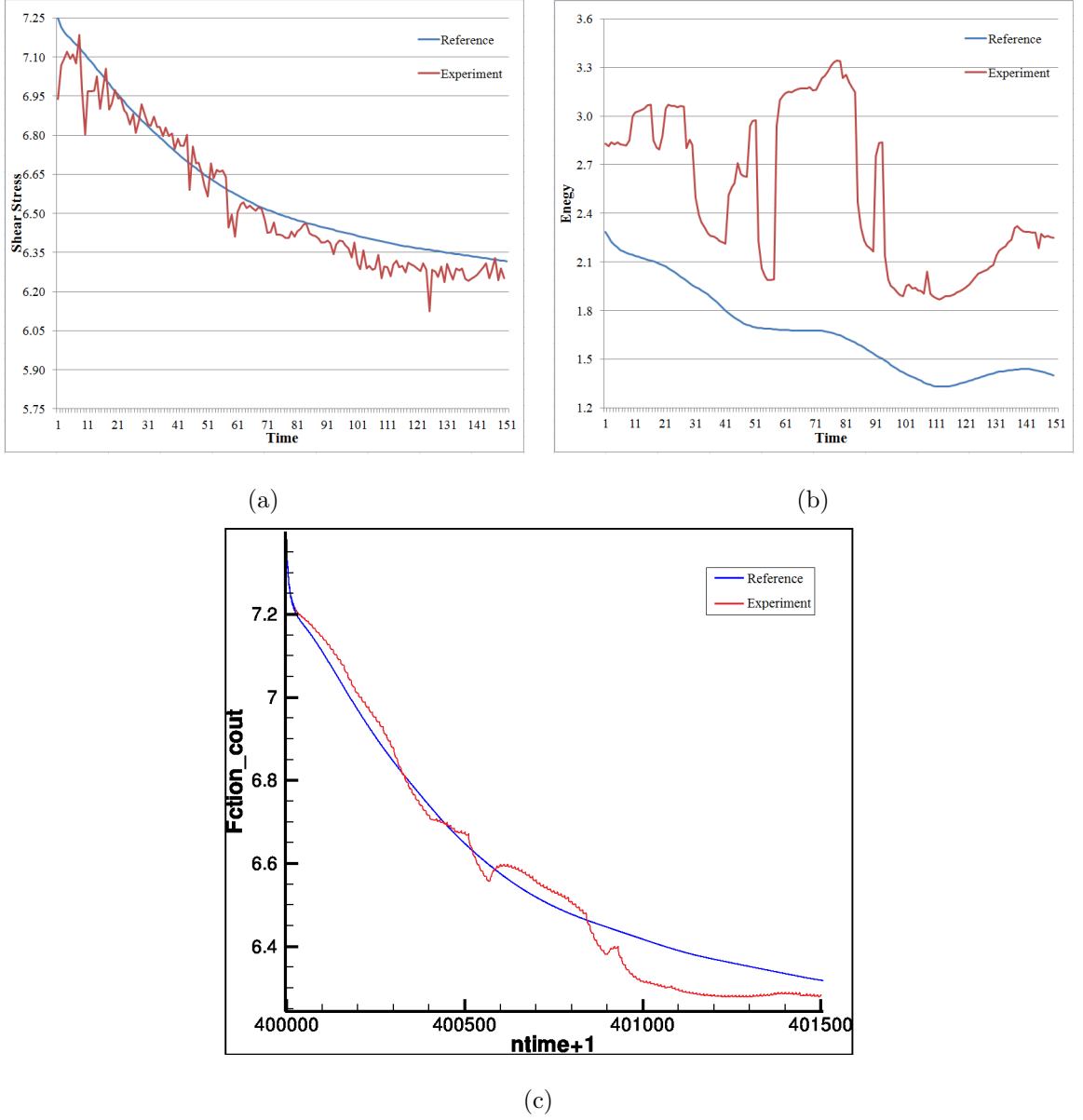
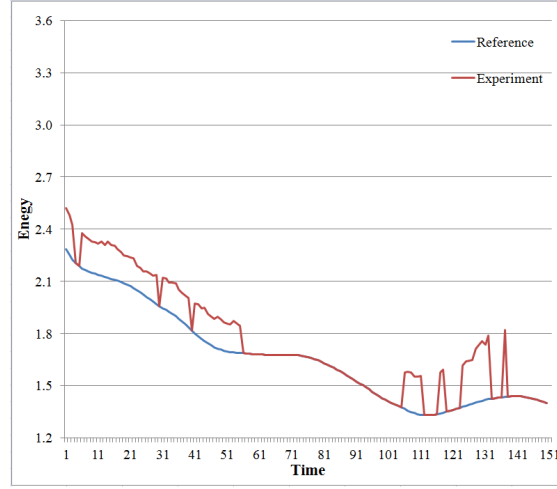


Figure 5.12. Graphs for Experiment 7: (a) estimated shear stress, (b) energy usage and (c) actual shear stress.

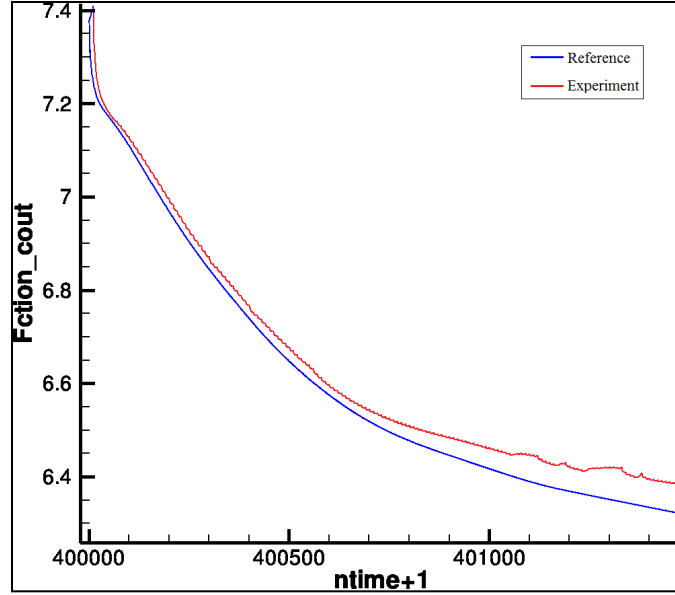
In this experiment, the total energy usage is 378.12 and at the end, 0.57% shear stress reduction is obtained.

5.2.8. Experiment 8

In this experiment, NGRG is used. The parameters are taken as $\beta_t = 1.1 + \min \{0.5, 0.65 \frac{t}{T}\}$, $\alpha_t^+ = 0.25 + 0.65 \frac{t}{T}$ and $\alpha_t^- = 0$, for $t \in T$.



(a)



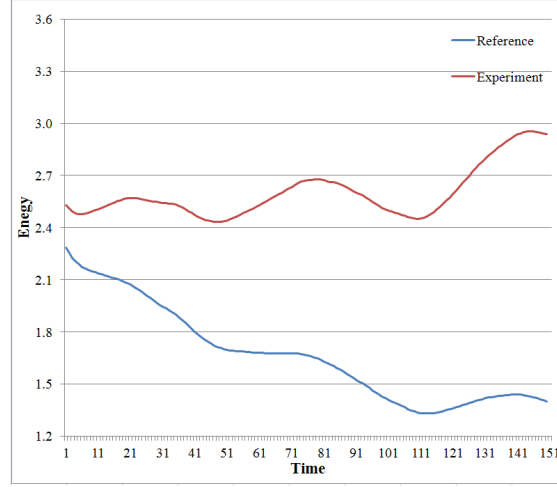
(b)

Figure 5.13. Graphs for Experiment 8: (a) energy usage and (b) actual shear stress.

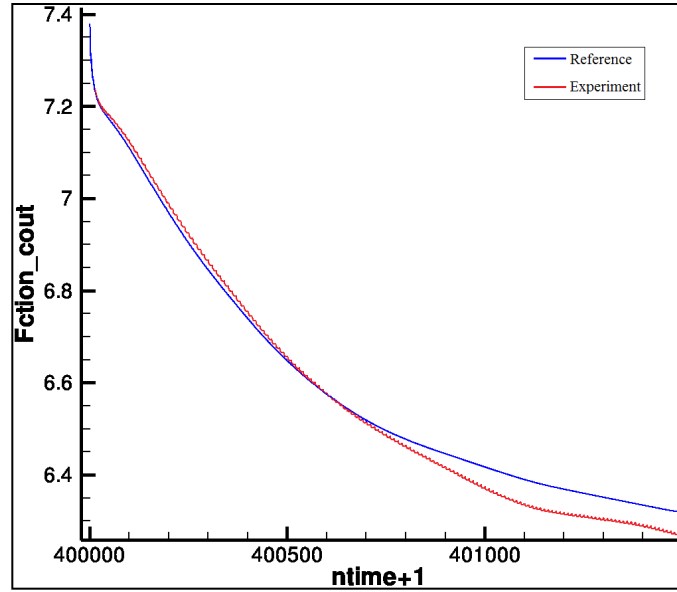
The total energy usage in this experiment is 263.64. As it can be seen from Figure 5.13b, shear stress reduction is not achieved at the end.

5.2.9. Experiment 9

In this experiment, NGRG is used. The parameters are taken as $\beta_t = 1.1 + \frac{t}{T}$, $\alpha_t^+ = 0.3 + 0.7\frac{t}{T}$ and $\alpha_t^- = -0.05 - 0.2\frac{t}{T}$, for $t \in \mathbf{T}$.



(a)



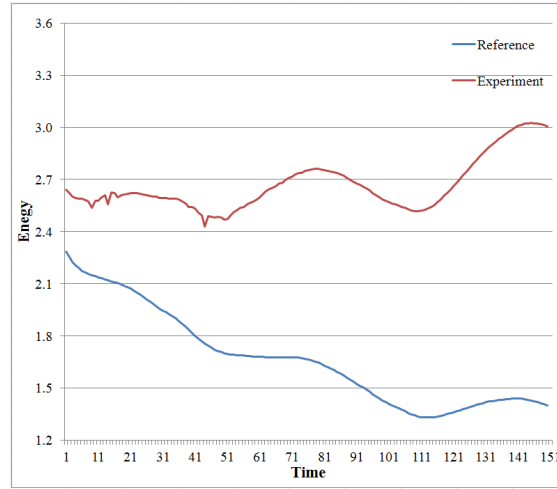
(b)

Figure 5.14. Graphs for Experiment 9: (a) energy usage and (b) actual shear stress.

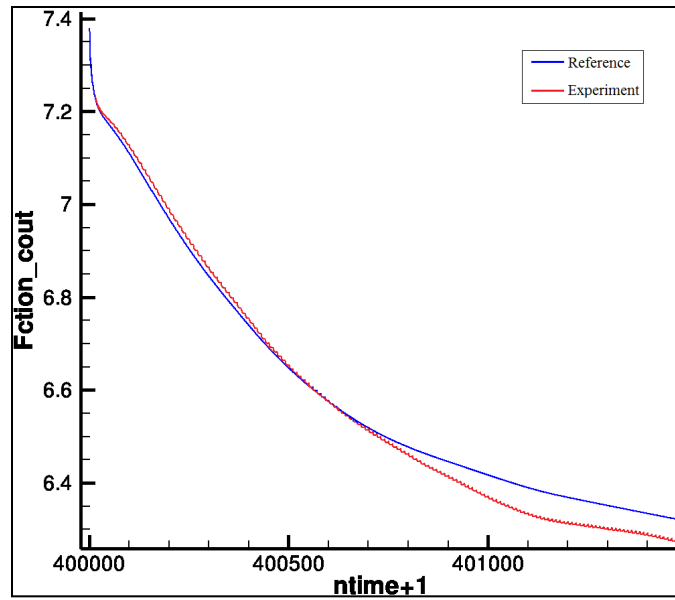
In this experiment, the total energy usage is 389.64 and at the end, 0.74% shear stress reduction is obtained.

5.2.10. Experiment 10

In this experiment, NGRG is used. The parameters are taken as $\beta_t = 1.15 + \frac{t}{T}$, $\alpha_t^+ = 0.3 + 0.7\frac{t}{T}$ and $\alpha_t^- = -0.05 - 0.2\frac{t}{T}$, for $t \in \mathbf{T}$.



(a)



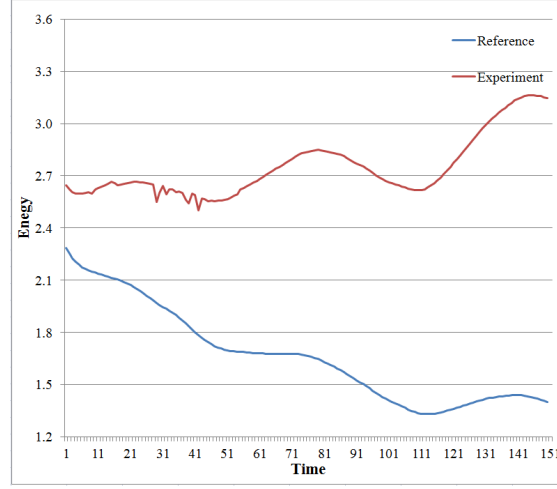
(b)

Figure 5.15. Graphs for Experiment 10: (a) energy usage and (b) actual shear stress.

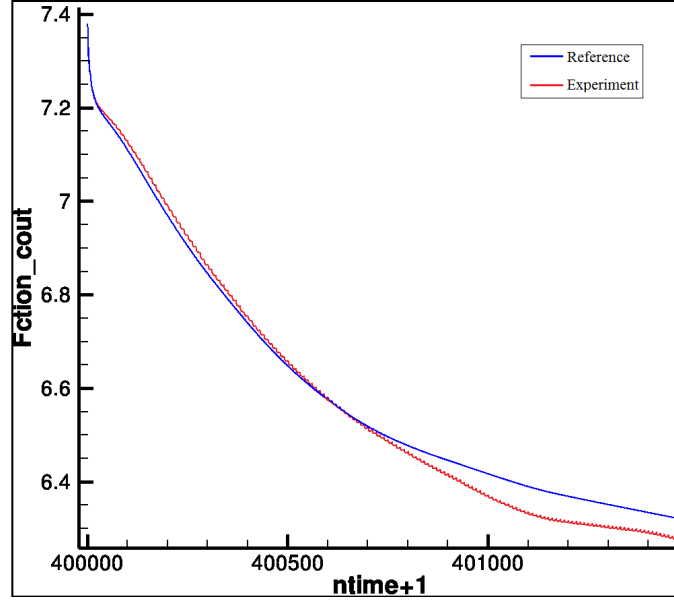
In this experiment, the total energy usage is 399.90 and at the end, 0.73% shear stress reduction is obtained.

5.2.11. Experiment 11

In this experiment, NGRG is used. The parameters are taken as $\beta_t = 1.15 + 1.1\frac{t}{T}$, $\alpha_t^+ = 0.4 + 0.7\frac{t}{T}$ and $\alpha_t^- = -0.05 - 0.2\frac{t}{T}$, for $t \in \mathbf{T}$.



(a)



(b)

Figure 5.16. Graphs for Experiment 10: (a) energy usage and (b) actual shear stress.

In this experiment, the total energy usage is 411.95 and at the end, 0.68% shear stress reduction is obtained.

6. CONCLUSIONS

In this thesis, we develop Neural Reduced Gradient (NRG) algorithm which is based on the approximation of the objective gradient for constrained minimization problems using neural networks. This method is tested with 19 problems. The results show that accurate and efficient solutions can be obtained provided that neural network with cross products (NNCT) is constructed and exact gradient is used. NRG especially works well if the objective function is convex.

We also consider a particular optimization problem coming from fluid mechanics, namely drag reduction. First of all, Actuator Management Problem (AMP) is modelled using mathematical programming techniques whose objective is the result of a simulator and constraints come from physical principles. Then, the objective function is estimated using constrained least squares estimation (CLSE) and we observe that corrected quadratic convex estimation (CQCE) and piecewise linear regression (PLR) work well. AMP is solved by appropriate models depending on the estimation types. Finally, an extension of our proposed method is applied to this problem. According to our experiments, it is possible to obtain 0.74% shear stress reduction. Although this number seems quite small, any attempt to eliminate turbulence is valuable since this may result in significant operational cost savings in areas such as airline industry. Also note that we are only able to work with the data generated by LEGI's simulator. It may be the case that we have reached the limitations of the physical model the simulator is based on.

There are some promising research directions for the solution of AMP. Better estimations can be achieved by either enhancing the ones presented in this thesis or using new methods. For instance, all the data accumulated through this study can be incorporated into estimation procedures in order to obtain more reliable parameter values or the number of line segments in PLR can be increased to fit better functions. Recall that we always use static estimations for a dynamic system. This can be cor-

rected by including the flow values of the previous time periods to each estimation model. However, in this case, the neighbourhood structure should be redefined due to possible delays in the physical process.

As another possible future work, cross terms can be added to the neural network constructed for AMP. We observe that this approach is beneficial for the test instances and therefore, can be applied here as well. However, note that including all the cross terms may potentially make it impossible to train such a network due to the extensive number of input neurons. Hence, only a subset of the cross terms should be introduced.

The reduction of the number of active holes over the membrane can be an entirely different future research topic. Since locating each actuator is costly, we may want to limit the number of actuators but still decrease the shear stress on the grid as much as possible. In this case, points without actuators should be considered separately.

Finally, Support Vector Machines can be used to obtain better function estimations. This methodology should be first applied to the test problems and if the results turn out to be successful, then it can be also used for AMP as well.

APPENDIX A: PIECEWISE LINEAR APPROXIMATION MODELS

We have developed seven different types of piecewise linear approximation models in order to solve AMP using PLR by adapting the formulations in [12] to the univariate case. Two of them, namely Log and CC, are already given in Section 4.2.7.3. The remaining five models are presented in the upcoming sections. Then, we discuss why CC is the most appropriate among all the models.

A.1. Disaggregated Convex Combination Models

In these formulations, following substitutions are done:

$$f_{it} = \sum_{k \in \mathbf{K}} (f_{it}^{k-1} \lambda_{it}^{k-1,l} + f_{it}^k \lambda_{it}^{k-1,r}) \quad (\text{A.1})$$

$$S_{it}(f_{it}) = \sum_{k \in \mathbf{K}} \left[S_{it}(f_{it}^{k-1}) \lambda_{it}^{k-1,l} + S_{it}(f_{it}^k) \lambda_{it}^{k-1,r} \right] \quad (\text{A.2})$$

$$e_{it}(f_{it}) = \sum_{k \in \mathbf{K}} \left[(f_{it}^{k-1})^2 \lambda_{it}^{k-1,l} + (f_{it}^k)^2 \lambda_{it}^{k-1,r} \right] \quad (\text{A.3})$$

Using the above substitutions, it is possible to rewrite AMP as follows:

$$\min \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \left[a_t + \sum_{k \in \mathbf{K}} \left[S_{it}(f_{it}^{k-1}) \lambda_{it}^{k-1,l} + S_{it}(f_{it}^k) \lambda_{it}^{k-1,r} \right] \right] \quad (\text{A.4})$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{K}} \left[(f_{ijt}^{k-1})^2 \lambda_{it}^{k-1,l} + (f_{it}^k)^2 \lambda_{it}^{k-1,r} \right] \leq E \quad (\text{A.5})$$

$$\sum_{i \in \mathbf{I}} \sum_{k \in \mathbf{K}} (f_{it}^{k-1} \lambda_{it}^{k-1,l} + f_{it}^k \lambda_{it}^{k-1,r}) = 0 \quad t \in \mathbf{T} \quad (\text{A.6})$$

$$l_{it} \leq \sum_{k \in \mathbf{K}} (f_{it}^{k-1} \lambda_{it}^{k-1,l} + f_{it}^k \lambda_{it}^{k-1,r}) \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.7})$$

$$\lambda_{it}^{kl} \geq 0; \lambda_{it}^{kr} \geq 0 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.8})$$

Since we know that the total shear stress effect functions are not all convex, we again need some additional variables and constraints. Similar to convex combination models, the basic model, abbreviated as DCC, has K additional binary variables for each nonconvex S_{it} function in the formulation whereas the logarithmic model, abbreviated as DLog, has only $\log_2 K = D$ binary variables.

The additional formulation for DCC is as follows:

$$\lambda_{it}^{kl} + \lambda_{it}^{kr} = y_{it}^k \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.9})$$

$$\sum_{k \in \mathbf{K}} y_{it}^k = 1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.10})$$

$$y_{it}^k = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.11})$$

The additional formulation for DLog is given as below:

$$\sum_{k \in \mathbf{K}} \psi_{dk} (\lambda_{it}^{kl} + \lambda_{it}^{kr}) \leq y_{it}^d \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (\text{A.12})$$

$$\sum_{k \in \mathbf{K}} (1 - \psi_{dk}) (\lambda_{it}^{kl} + \lambda_{it}^{kr}) \leq 1 - y_{it}^d \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (\text{A.13})$$

$$\sum_{k \in \mathbf{K}} (\lambda_{it}^{kl} + \lambda_{it}^{kr}) = 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.14})$$

$$y_{it}^d = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; d \in \mathbf{D} \quad (\text{A.15})$$

Here, ψ_{dk} is the entry of a $D \times K$ matrix whose columns are constructed by choosing K distinct binary vectors of size D . Such a construction can be done using the following rule

$$\psi_{dk} = \text{Rem} \left(\left\lfloor \frac{k-1}{2^{D-d}} \right\rfloor, 2 \right) \quad (\text{A.16})$$

where Rem is as defined in Section 4.2.7.3.

Note that this simple formula is valid since S_{it} is univariate. For the general case, refer to [12].

A.2. Multiple Choice Model

In this formulation, following substitutions are done:

$$f_{it} = \sum_{k \in \mathbf{K}} \gamma_{it}^k \quad (\text{A.17})$$

$$S_{it}(f_{it}) = \sum_{k \in \mathbf{K}} (ms_{it}^k \gamma_{it}^k + cs_{it}^k y_{it}^k) \quad (\text{A.18})$$

$$e_{it}(f_{it}) = \sum_{k \in \mathbf{K}} (me_{it}^k \gamma_{it}^k + ce_{it}^k y_{it}^k) \quad (\text{A.19})$$

where

$$\begin{aligned} ms_{it}^k &= \frac{S_{it}(f_{it}^k) - S_{it}(f_{it}^{k-1})}{f_{it}^k - f_{it}^{k-1}} & me_{it}^k &= \frac{(f_{it}^k)^2 - (f_{it}^{k-1})^2}{f_{it}^k - f_{it}^{k-1}} = f_{it}^k + f_{it}^{k-1} \\ cs_{it}^k &= \frac{f_{it}^k S_{it}(f_{it}^{k-1}) - f_{it}^{k-1} S_{it}(f_{it}^k)}{f_{it}^k - f_{it}^{k-1}} & ce_{it}^k &= \frac{f_{it}^k (f_{it}^{k-1})^2 - f_{it}^{k-1} (f_{it}^k)^2}{f_{it}^k - f_{it}^{k-1}} \end{aligned} \quad \text{and} \quad (\text{A.20})$$

for $i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K}$.

It is possible to rewrite AMP using the above substitutions. The resulting model, abbreviated as MC, is given as follows:

$$\min \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \left[a_t + \sum_{k \in \mathbf{K}} (ms_{it}^k \gamma_{it}^k + cs_{it}^k y_{it}^k) \right] \quad (\text{A.21})$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \sum_{k \in \mathbf{K}} (me_{it}^k \gamma_{it}^k + ce_{it}^k y_{it}^k) \leq E \quad (\text{A.22})$$

$$\sum_{i \in \mathbf{I}} \sum_{k \in \mathbf{K}} \gamma_{it}^k = 0 \quad t \in \mathbf{T} \quad (\text{A.23})$$

$$l_{it} \leq \sum_{k \in \mathbf{K}} \gamma_{it}^k \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.24})$$

$$\gamma_{it}^k \in \mathbb{R}; y_{it}^k = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.25})$$

We again need some additional variables and constraints. This time, we need K additional binary variables for each (i, j) pair in the formulation. The additional formulation is as follows:

$$f_{it}^{k-1} y_{it}^k \leq \gamma_{it}^k \leq f_{it}^k y_{it}^k \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.26})$$

$$\sum_{k \in \mathbf{K}} y_{it}^k = 1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.27})$$

A.3. Incremental Model

In this formulation, following substitutions are done:

$$f_{it} = f_{it}^0 + \sum_{k \in \mathbf{K}} (f_{it}^k - f_{it}^{k-1}) \delta_{it}^k \quad (\text{A.28})$$

$$S_{it}(f_{it}) = S_{it}(f_{it}^0) + \sum_{k \in \mathbf{K}} [S_{it}(f_{it}^k) - S_{it}(f_{it}^{k-1})] \delta_{it}^k \quad (\text{A.29})$$

$$e_{it}(f_{it}) = (f_{it}^0)^2 + \sum_{k \in \mathbf{K}} [(f_{it}^k)^2 - (f_{it}^{k-1})^2] \delta_{it}^k \quad (\text{A.30})$$

It is possible to rewrite AMP using the above substitutions. The resulting model, abbreviated as Inc, is given as follows:

$$\min \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \left[a_t + \left[S_{it}(f_{it}^0) + \sum_{k \in \mathbf{K}} [S_{it}(f_{it}^k) - S_{it}(f_{it}^{k-1})] \delta_{it}^k \right] \right] \quad (\text{A.31})$$

$$\text{s.t.} \quad \sum_{i \in \mathbf{I}} \sum_{t \in \mathbf{T}} \left[(f_{it}^0)^2 + \sum_{k \in \mathbf{K}} [(f_{it}^k)^2 - (f_{it}^{k-1})^2] \delta_{it}^k \right] \leq E \quad (\text{A.32})$$

$$\sum_{i \in \mathbf{I}} \left[f_{it}^0 + \sum_{k \in \mathbf{K}} (f_{it}^k - f_{it}^{k-1}) \right] \delta_{it}^k = 0 \quad t \in \mathbf{T} \quad (\text{A.33})$$

$$l_{it} \leq f_{it}^0 + \sum_{k \in \mathbf{K}} (f_{it}^k - f_{it}^{k-1}) \delta_{it}^k \leq u_{it} \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.34})$$

$$\delta_{it}^k \geq 0 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \quad (\text{A.35})$$

We again need some additional variables and constraints. This time, we need $K - 1$ additional binary variables for each nonconvex S_{it} function in the formulation. The additional formulation is as follows:

$$\delta_{it}^1 \leq 1 \quad i \in \mathbf{I}; t \in \mathbf{T} \quad (\text{A.36})$$

$$\delta_{it}^{k+1} \leq y_{it}^k \leq \delta_{it}^k \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \setminus \{K\} \quad (\text{A.37})$$

$$y_{it}^k = 0, 1 \quad i \in \mathbf{I}; t \in \mathbf{T}; k \in \mathbf{K} \setminus \{K\} \quad (\text{A.38})$$

A.4. Specially Ordered Sets of Type 2 Model

We have already given six piecewise linear approximation models in which four of them have linear number of additional binary variables in the number of line segments and two of them have logarithmic number of additional binary variables in the number of line segments. An alternative approach is to have no additional binary variables at all. This can be done by using Specially Ordered Sets of Type 2 (SOS2). In this case, however, the solution procedure inherent in the software that is going to be used should support this feature. Since CPLEX 12.2 [42] has this option, we can easily take advantage of this modelling strategy. The resulting model is straightforward. In addition to the common constraints of CC and Log, we just declare λ_{it}^k as an SOS2 set for each (i, t) pair. Solution procedure internally forces at most two variables to have positive value and if that occurs, then they must be consecutive.

A.5. Selection of the Most Suitable Model

We need to choose the most appropriate model out of the seven piecewise linear approximation models introduced here. For this purpose, we generated 12 random AMP instances which do not necessarily reflect the real experimentation environment. As pointed out in [12], since Log and DLog have logarithmic number of binary variables, they are expected to perform better than the others. It turns out that this is mostly the case according to our tests summarized in Table A.1. Each row gives a different model

with the number of line segments K and each column gives the number of holes I and the number of time periods T . Each entry gives the solution time in seconds limited² by 50000 iterations. The experiment is carried out on a computer with Microsoft Windows 7 x64 Edition operating system and Intel i7 2.00 GHz processor with 8.00 GB RAM.

Table A.1. Comparison of piecewise linear approximation models for 12 problems in

		CPU seconds.					
		$T = 1$			$T = 3$		
	Model	$I = 512$	$I = 1024$	$I = 2048$	$I = 512$	$I = 1024$	$I = 2048$
$K = 8$	CC	0.92	1.59	3.22	2.25	4.58	11.47
	Log	0.73	1.31	2.82	2.08	4.21	11.64
	DCC	1.26	2.80	6.47	4.30	11.18	33.38
	DLog	1.12	2.19	4.55	3.38	5.66	16.53
	Inc	1.06	2.00	4.45	2.98	3.00	19.58
	MC	2.09	3.11	8.32	5.21	13.85	117.79
	SOS2	0.88	2.06	7.03	4.01	4.32	52.99
$K = 16$	CC	1.50	2.81	5.63	3.84	8.05	20.33
	Log	1.19	2.20	4.66	3.11	6.24	18.17
	DCC	2.67	6.33	17.71	10.99	33.46	112.34
	DLog	2.04	4.21	8.44	6.32	9.84	28.25
	Inc	2.19	4.55	11.57	7.39	5.79	87.23
	MC	4.94	7.30	21.99	14.08	40.39	123.18
	SOS2	1.24	3.03	9.19	5.14	7.72	67.00

The average performance of piecewise linear approximation models is given in Table A.2. As we may expect, Log performs the best on the average. However, CC outperforms DLog unexpectedly. This is mainly due to the choice of K , which is comparatively smaller than the ones used in [12]. Despite statistically insufficient data, we can still say that the results here are consistent with the literature.

²No solution is found for MC, 16, 2048, 3 due to this limitation.

Table A.2. Average performance comparison of piecewise linear approximation models for 12 problems in CPU seconds.

Model	$\overline{\text{CPU}}$
CC	5.52
Log	4.86
DCC	20.24
DLog	7.71
MC	21.73
Inc	12.65
SOS2	13.72

We can deduce that Log is the most advantageous among the seven models. However, we did not prefer to use it for the solution of AMP since its binary variables have no physical meaning. This is important because AMP is a large-scale MILP and the second simplification mentioned in Section 4.2.7.3, namely preprocessing the model and setting the binary variables to zero which correspond to the line segments not included between lower and upper bounds, cannot be applied to Log. Therefore, we pick the second best model CC since we can take advantage of this simplification.

APPENDIX B: OPTIMAL SOLUTIONS OF THE TEST PROBLEMS

For the first five problems, the optimal solutions are found by using CPLEX 12.2. For the others, the optimal solutions are taken from the literature as cited unless otherwise stated. Note that functions through (i)-(viii) are convex and (ix)-(xi) have a single global minimum although they are nonconvex. The rest of the functions are nonconvex and have more than one local minima except for (xiv). It has a single local minimum which is not global.

- (i) QP(3, 5) has a global minimum at (0.940, -0.722, 1.254, 0.657, 0.171) with objective value 1.968.
- (ii) QP(6, 8) has a global minimum at (1.060, 1.927, -0.429, 0.068, 0.923, 1.221, -0.036, 0.050) with objective value -8.976.
- (iii) QP(8, 12) has a global minimum at (0.768, 0.078, -0.547, -0.058, 0.445, -0.365, 0.030, 0.038, -0.780, 0.878, -0.070, 0.032) with objective value -13.758.
- (iv) QP(9, 15) has a global minimum at (-0.533, 0.008, -0.954, 0.634, -0.152, 0.406, -1.102, -0.102, 0.032, -0.650, 0.333, -0.043, 0.134, -0.559, 0.789) with objective value -11.888.
- (v) QP(12, 20) has a global minimum at (0.867, 0.453, -0.114, -0.287, -0.138, -0.070, -1.241, 0.066, 1.289, 0.045, 0.265, -0.388, -0.231, 0.640, -0.650, -0.558, 0.329, 0.009, 0.483, -0.489) with objective value -6.059.
- (vi) Matyas Function [47] has a global minimum at (0, 0) with objective value 0.
- (vii) Trid Function [47] has a global minimum at (6, 10, 12, 12, 10, 6) with objective value -50.
- (viii) Zakharov Function [47] has a global minimum at (0, 0, 0) with objective value 0.
- (ix) Colville Function [47] has a global minimum at (1, 1, 1, 1) with objective value 0.
- (x) Beale Function [47] has a global minimum at (3, 0.5) with objective value 0.
- (xi) Rosenbrock Function [47] has a global minimum at (1, 1) with objective value 0.
- (xii) Dixon & Price Function [47] has two global minima at (1.000, 0.707, 0.595, 0.545,

- 0.522) and (1.000, 0.707, 0.595, 0.545, 0.522) with objective value 0.
- (xiii) Ravindran Function [48] has two local minima at (-1.514, -0.859) and (1.514, 0.859) with objective value -3.844.
- (xiv) Winston Function [49] has a local minimum at (0.4, 0.447) with objective value -0.072.
- (xv) Himmelblau Function [50] has four global minima at (-2.805, 3.131), (-3.779, -3.283), (3, 2) and (3.584, -1.848) with objective value 0.
- (xvi) Branin Function [47] has three global minima at $(-\pi, 12.275)$, $(\pi, 2.275)$ and (9.425, 2.475) with objective value 0.398.
- (xvii) Six-Hump Function [51] has two global minima at (0.090, -0.713) and (-0.090, 0.713) with objective value -1.032. There are two local minima at (-1.703, 0.796) and (1.703, -0.796) with objective value -0.215 and two other local minima at (-1.607, -0.569) and (1.607, 0.569) with objective value 2.104.
- (xviii) Goldstein & Price Function [52] has four local minima at (0, -1), (-0.6, -0.4), (1.8, 0.2) and (1.2, 0.8) with respective objective function values 3, 30, 84 and 840.
- (xix) Broyden Function [53] has a global minimum at (-0.571, -0.682, -0.702, -0.706, -0.705, -0.701, -0.692, -0.666, -0.596, -0.416) with objective value 0 found by BARON using GAMS. Also, there are six more local minima at (1.831, -0.107, -0.588, -0.673, -0.669, -0.612, -0.454, -0.055, 0.758, 1.413), (0.896, 1.437, -0.088, -0.516, -0.563, -0.427, -0.035, 0.738, 1.312, 0.233), (1.573, 0.375, 0.168, 0.605, 1.052, 0.586, 0.420, 0.767, 1.172, 0.267), (1.764, 0.031, -0.324, -0.069, 0.702, 1.498, -0.066, -0.506, -0.550, -0.401), (1.739, 0.083, -0.247, 0.046, 0.749, 1.171, 0.394, 0.285, 0.792, 1.312) and (1.830, -0.102, -0.578, -0.650, -0.607, -0.450, -0.055, 0.723, 1.320, 0.235) with respective objective function values 1.360, 1.029, 1.051, 1.051, 0.713 and 0.397 as given in [54].

REFERENCES

1. Fu, M. C., “Feature Article: Optimization for Simulation: Theory vs. Practice”, *INFORMS Journal on Computing*, Vol. 14, No. 3, pp. 192–215, 2002.
2. Andradóttir, S., “Simulation Optimization”, J. Banks (Editor), *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, pp. 307–333, John Wiley & Sons, 1998.
3. Barton, R. R., “Simulation Optimization Using Metamodels”, *Winter Simulation Conference*, pp. 230–238, 2009.
4. Kabirian, A. and S. Ólafsson, “Continuous Optimization via Simulation Using Golden Region Search”, *European Journal of Operational Research*, Vol. 208, No. 1, pp. 19–27, 2011.
5. Liu, H. and S. Maghsoodloo, “Simulation Optimization Based on Taylor Kriging and Evolutionary Algorithm”, *Applied Soft Computing*, Vol. 11, No. 4, pp. 3451–3462, 2011.
6. Wang, L., “A Hybrid Genetic Algorithm-Neural Network Strategy for Simulation Optimization”, *Applied Mathematics and Computation*, Vol. 170, No. 2, pp. 1329–1343, 2005.
7. Duvinneau, R. and M. Visonneau, “Hybrid Genetic Algorithms and Artificial Neural Networks for Complex Design Optimization in CFD”, *International Journal for Numerical Methods in Fluids*, Vol. 44, No. 11, pp. 1257–1278, 2004.
8. Poloni, C., A. Giurgevich, L. Onesti and V. Pediroda, “Hybrid Genetic Algorithms and Artificial Neural Networks for Complex Design Optimization in CFD”, *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, p. 403–420, 2000.

9. Angün, E. and J. Kleijnen, “An Asymptotic Test of Optimality Conditions in Multiresponse Simulation Optimization”, *INFORMS Journal on Computing*, Vol. 24, No. 1, pp. 53–65, 2012.
10. Andradóttir, S. and A. A. Prudius, “Balanced Explorative and Exploitative Search with Estimation for Simulation Optimization”, *INFORMS Journal on Computing*, Vol. 21, No. 2, pp. 193–208, 2009.
11. Keha, A. B., I. R. de Farias Jr. and G. L. Nemhauser, “Models for Representing Piecewise Linear Cost Functions”, *Operations Research Letters*, Vol. 32, No. 1, pp. 44–48, 2004.
12. Vielma, J. P., S. Ahmed and G. L. Nemhauser, “Mixed Integer Models for Non-separable Piecewise Linear Optimization: Unifying Framework and Extensions”, *Operations Research*, Vol. 58, No. 2, pp. 303–315, 2010.
13. Keha, A. B., *A Polyhedral Study of Nonconvex Piecewise Linear Optimization*, PhD dissertation, Georgia Institute of Technology, 2003.
14. Kameshwaran, S. and Y. Narahari, “Nonconvex Piecewise Linear Knapsack Problems”, *European Journal of Operational Research*, Vol. 192, No. 1, pp. 56–68, 2009.
15. Gans, P., *Data Fitting in the Chemical Sciences by the Method of Least Squares*, John Wiley & Sons, NY, USA, 1992.
16. Kleinbaum, D. G., L. L. Kupper and K. E. Muller, *Applied Regression Analysis and Other Multivariable Methods*, Thomson Brooks/Cole, CA, USA, 2008.
17. Toriello, A. and J. P. Vielma, “Fitting Piecewise Linear Continuous Functions”, *Industrial Engineering*, pp. 1–30, 2010.
18. Liew, C. K., “Inequality Constrained Least Squares Estimation”, *Journal of the American Statistical Association*, Vol. 71, No. 355, pp. 746–751, 1976.

19. Robertson, D. G. and J. H. Lee, “On the Use of Constraints in Least Squares Estimation and Control”, *Automatica*, Vol. 38, No. 7, pp. 1113–1123, 2002.
20. Groeneboom, P., G. Jongbloed and J. A. Wellner, “Estimation of a Convex Function: Characterizations and Asymptotic Theory”, *Annals of Statistics*, Vol. 29, No. 6, pp. 1653–1698, 2001.
21. Kailath, T., A. Sayed and B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, USA, 2000.
22. Squires, T. M. and S. R. Quake, “Microfluidics: Fluid Physics at the Nanoliter Scale”, *Reviews of Modern Physics*, Vol. 77, pp. 977–1026, 2005.
23. Yager, P., T. Edwards, E. Fu, K. Helton, K. Nelson, M. R. Tam and B. H. Weigl, “Microfluidic Diagnostic Technologies for Global Public Health”, *NATURE*, Vol. 442, pp. 412–418, 2006.
24. Xu, T. and K. Chakrabarty, “Integrated Droplet Routing and Defect Tolerance in the Synthesis of Digital Microfluidic Biochips.”, *ACM Journal on Emerging Technologies in Computing Systems*, Vol. 4, No. 3, 2008.
25. Zhao, Y., R. Sturmer, K. Chakrabarty and V. K. Pamula, “Synchronization of Concurrently-Implemented Fluidic Operations in Pin-Constrained Digital Microfluidic Biochips”, *VLSI Design*, pp. 69–74, IEEE, 2010.
26. Pasaniuc, B., R. Garfinkel, I. Mandoiu and A. Zelikovsky, “Optimal Testing of Digital Microfluidic Biochips”, *INFORMS Journal on Computing*, Vol. 23, No. 4, pp. 518–529, 2011.
27. Koklu, M. and O. Baysal, “Modelling, Parameterisation and Simulation-based Optimisation of a Microflow-Control Actuator”, *International Journal of Computational Fluid Dynamics*, Vol. 22, pp. 367–381, July 2008.

28. Fukagata, K. and N. Kasagi, “Suboptimal Control for Drag Reduction via Suppression of Near-wall Reynolds Shear Stress”, *International Journal of Heat and Fluid Flow*, Vol. 25, p. 341–350, 2004.
29. Tardu, S. F. and O. Doche, “One-information Suboptimal Control Repercussion on the Fine Structure of Wall Turbulence”, *Computers and Fluids*, Vol. 38, pp. 637–647, 2009.
30. Kim, J., “Control of Turbulent Boundary Layers”, *Physics of Fluids*, Vol. 15, No. 5, pp. 1093–1105, 2003.
31. Lee, C., J. Kim, D. Babcock and R. Goodman, “Application of Neural Networks to Turbulence Control for Drag Reduction”, *Physics of Fluids*, Vol. 9, No. 6, pp. 1740–1747, 1997.
32. Morimoto, K., K. Iwamoto, Y. Suzuki and N. Kasagi, “Genetic Algorithm-Based Optimization of Feedback Control Scheme for Wall Turbulence”, *Proceedings of 3rd Symposium on Smart Control of Turbulence*, pp. 107–113, 2002.
33. Yoshino, T., Y. Suzuki and N. Kasagi, “Drag Reduction of Turbulence Air Channel Flow with Distributed Micro Sensors and Actuators”, *Journal of Fluid Science and Technology*, Vol. 3, No. 1, pp. 137–148, 2008.
34. Bazaraa, M. S., H. D. Sherali and C. Shetty, *Nonlinear Programming : Theory and Algorithms*, John Wiley & Sons, NJ, USA, third edn., 2006.
35. Hertz, J., A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, CA, USA, 1991.
36. Cybenko, G. V., “Approximation by Superpositions of a Sigmoidal Function”, *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, pp. 303–314, 1989.
37. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, MA, USA, second

edn., 2010.

38. Herstein, I. N., *Abstract Algebra*, John Wiley & Sons, NJ, USA, third edn., 1999.
39. Tardu, S. F., 2011, personal communication.
40. Ipsen, I. C., *Numerical Matrix Analysis : Linear Systems and Least Squares*, Society for Industrial and Applied Mathematics, PA, USA, 2009.
41. *Math.NET Numerics Package*, 2011, <http://numerics.mathdotnet.com/>, accessed at June 2012.
42. *User's Manual for CPLEX Version 12.2*, IBM, 2010.
43. *Knitro User's Manual Version 6.0*, Ziena Optimization Inc., 2009.
44. Hartman, P., "On Functions Representable as a Difference of Convex Functions", *Pacific Journal of Mathematics*, Vol. 9, No. 3, pp. 707–713, 1959.
45. Horst, R. and N. V. Thoai, "DC Programming: Overview", *Journal of Optimization Theory and Applications*, Vol. 103, No. 1, pp. 1–43, 1999.
46. Vielma, J. P. and G. L. Nemhauser, "Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints", *Mathematical Programming*, Vol. 128, No. 1-2, pp. 49–72, 2011.
47. Hedar, A.-R., *Test Functions for Unconstrained Global Optimization*, 2005, http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm, accessed at June 2012.
48. Ravindran, A., K. M. Ragsdell and G. V. Reklaitis, *Engineering Optimization: Methods and Applications*, John Wiley & Sons, NJ, USA, second edn., 2006.
49. Winston, W. L., *Operations Research: Applications and Algorithms*, Thomson

Brooks/Cole, CA, USA, fourth edn., 2004.

50. Himmelblau, D., *Applied Nonlinear Programming*, McGraw Hill, NY, USA, fourth edn., 1972.
51. Tsoulos, I. G. and I. E. Lagaris, “MinFinder: Locating All the Local Minima of a Function”, *Computer Physics Communications*, Vol. 174, pp. 166–179, 2006.
52. Goldstein, A. A. and J. F. Price, “On Descent from Local Minima”, *Mathematics of Computation*, Vol. 25, No. 115, pp. 569–574, 1971.
53. Moré, J. J., B. S. Garbow and K. E. Hillstom, “Testing Unconstrained Optimization Software”, *ACM Transactions on Mathematical Software*, Vol. 7, No. 1, pp. 17–41, 1981.
54. Kuntsevich, A. and F. Kappel, *Moré Set of Test Functions*, 1997, <http://www.uni-graz.at/imawww/kuntsevich/solvopt/results/moreset.html>, accessed at June 2012.