PERFORMANCES OF SHAPEFREE RBF-NEURAL NETWORKS IN PATTERN RECOGNITION APPLICATION WITH LARGE SCATTERED DATA SETS

SUNISA TAVAEN

มหาวิทยาลัยเทคโนโลยีสุรนารี

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Applied Mathematics
Suranaree University of Technology
Academic Year 2021

สมรรถนะของโครงข่ายประสาทเทียมด้วยฟังก์ชันรัศมีฐานหลักด้วยพารามิเตอร์ที่ปราศจากพจน์ปรับค่า สำหรับประยุกต์ในการจดจำแบบแผนของข้อมูลขนาดใหญ่ที่กระจัดกระจาย

นางสาวสุนิสา ตาแว่น

มหาวิทยาลัยเทคโนโลยีสุรนารี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2564

PERFORMANCES OF SHAPEFREE RBF-NEURAL NETWORKS IN PATTERN RECOGNITION APPLICATION WITH LARGE SCATTERED DATA SETS

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree.

Thesis Examining Committee

Benjawan Rodjanadid
(Asst. Prof. Dr. Benjawan Rodjanadid)
Chairperson

(Assoc. Prof. Dr. Sayan Kaennakham)
Member (Thesis Advisor)

(Assoc. Prof. Dr. Ratchada Viriyapong)
Member

Panu Yimmuang
(Asst. Prof. Dr. Panu Yimmuang)
Member

(Asst. Prof. Dr. Poj Lertchoosakul)
Member

(Assoc. Prof. Dr. Chatchai Jothityangkoon)
Vice Rector for Academic Affairs
and Quality Assurance

(Prof. Dr. Santi Maensiri)
Dean of Institute of Science

มหาวิทยาลัยเทคโนโลยีสุรนารี

สุนิสา ตาแว่น : สมรรถนะของโครงข่ายประสาทเทียมด้วยฟังก์ชันรัศมีฐานหลักด้วยพารามิเตอร์ที่ปราศจากพจน์ปรับค่า สำหรับประยุกต์ในการจดจำแบบแผนของข้อมูลขนาดใหญ่ที่กระจัดกระจาย (PERFORMANCES OF SHAPEFREE RBF-NEURAL NETWORKS IN PATTERN RECOGNITION APPLICATION WITH LARGE SCATTERED DATA SETS) อาจารย์ที่ปรึกษา : รองศาสตราจารย์ ดร.สายันต์ แก่นนาคำ, 181 หน้า.

คำสำคัญ : โครงข่ายประสาทเทียม/ฟังก์ชันรัศมีฐานหลัก/การจดจำแบบแผน/ข้อมูลที่กระจัดกระจาย

งานวิจัยนี้มีวัตถุประสงค์หลัก คือ การศึกษาเชิงเปรียบเทียบสมรรถนะของฟังก์ชันรัศมีฐานหลักประเภทที่ไม่มีตัวปรับค่า โดยได้มีการรวบรวมฟังก์ชันรัศมีฐานหลักในลักษณะนี้มาทั้งสิ้น 16 แบบ เพื่อนำมาประยุกต์ใช้ผ่านโครงข่ายประสาทเทียมและความสามารถในการเป็นตัวแทน (ซึ่งถือเป็นเครื่องมือหนึ่งในการลดมิติของข้อมูลในระบบวิเคราะห์) ปัญหาที่สนใจได้แก่ ปัญหาการจดจำแบบแผนโดยได้ทดสอบกับขนาดข้อมูลที่มีขนาดใหญ่และมีการกระจัดกระจายในรูปแบบที่แตกต่างกัน ข้อมูลเริ่มต้นได้มีการถูกรบกวนด้วยสัญญาณรบกวนและแบ่งออกเป็น 2 ชุดสำหรับ ‘ชุดข้อมูลฝึกฝน/ชุดข้อมูลทดสอบ’ ในการทดลองเชิงตัวเลขทั้งหมดในการวิจัยนี้ ผลการทดลองที่ได้จากการใช้ฟังก์ชันแต่ละรูป ถูกตัดสินประสิทธิภาพโดยใช้เกณฑ์หลายประการ ได้แก่ ความแม่นยำ ตัวเลขเงื่อนไข (ของเมทริกซ์การประมาณค่า) เวลาที่ซีพียูใช้ดำเนินการ หน่วยเก็บข้อมูลที่ซีพียูใช้ ลักษณะโมเดลที่จำเพาะมากเกินไปและโมเดลที่จำเพาะน้อยเกินไป และจำนวนจุดศูนย์กลางที่ถูกสร้างขึ้น และนอกจากนี้ เพื่อเป็นการเปรียบเทียบกับฟังก์ชันรัศมีฐานหลักประเภทปรกติ (คือประเภทที่มีตัวปรับค่า) การทดลองยังได้ศึกษาฟังก์ชันประเภทมัลติควอดริกไปด้วย และผลการทดลองที่ได้ในแต่ละกรณีได้มีการนำผลมาเปรียบเทียบกันอย่างระมัดระวัง ผลการทดลองโดยรวมทั้งหมดที่ได้จากงานวิจัยนี้ บ่งชี้อย่างชัดเจนว่าฟังก์ชันรัศมีฐานหลักบางรูปสามารถให้ค่าความแม่นยำที่สูงและมีประสิทธิภาพที่จะรับมือกับปัญหาในรูปแบบของการจดจำแบบแผนได้เทียบเท่า (หรือดีกว่า) ฟังก์ชันประเภทดั้งเดิม การค้นพบนี้เป็นประโยชน์อย่างยิ่งต่อการใช้งานของฟังก์ชันรัศมีฐานหลัก ที่จะไม่ต้องประสบกับปัญหาการหาค่าพารามิเตอร์อีกต่อไป

มหาวิทยาลัยเทคโนโลยีสุรนารี

สาขาวิชาคณิตศาสตร์ ลายมือชื่อนักศึกษา สุนิสา ตาแว่น

ปีการศึกษา 2564 ลายมือชื่ออาจารย์ที่ปรึกษา S.

SUNISA TAVAEN : PERFORMANCES OF SHAPEFREE RBF-NEURAL NETWORKS IN PATTERN RECOGNITION APPLICATION WITH LARGE SCATTERED DATA SETS : THESIS ADVISOR : ASSOCIATE PROFESSOR SAYAN KAENNAKHAM, Ph.D. 181 PP.

Keyword : NEURAL NETWORKS/RADIAL BASIS FUNCTIONS/PATTERN RECOGNITION/SCATTERED DATA SETS

This work focuses on radial basis functions containing no parameters. The main objective is to comparatively explore more of their effectiveness. For this, a total of sixteen forms of shapeless radial basis functions are gathered and investigated under the context of the pattern recognition problem through the structure of radial basis function neural networks, with the use of a dimensionality reduction algorithm namely the Representational Capability (RC) algorithm. Different sizes of datasets are disturbed with noise before being imported into the algorithm as 'training/testing' datasets. Each shapeless radial basis function is monitored carefully with effectiveness criteria including accuracy, condition number (of the interpolation matrix), CPU time, CPU-storage requirement, underfitting and overfitting aspects, and the number of centres being generated. For the sake of comparison, the well-known Multiquadric-radial basis function (MQ-RBF) is included as a representative of shape-contained radial basis functions. The numerical results have revealed that some forms of shapeless radial basis functions show good potential and are even better than Multiquadric itself indicating strongly that the future use of radial basis function may no longer face the pain of choosing a proper shape when shapeless forms may be equally (or even better) effective.

วิทยาลัยเทคโนโลยีสุร

School of Mathematics
Academic Year 2021

Student's Signature สุนิสา ตาแว่น
Advisor's Signature

ACKNOWLEDGEMENT

First and foremost, I would like to acknowledge all the kind support and understanding I have been receiving from my parents and family. I would also like to express my deepest gratitude to my advisor Associate Professor Dr. Sayan Kaennakham for giving me the motivation to do the research, encouragement, continuous support, caring and patience for doing this thesis.

Additionally, I would like to give a million thanks to Assistant Professor Dr. Benjawan Rodjanadid, Assistant Professor Dr. Panu Yimmuang, Assistant Professor Dr. Poj Lertchoosakul, and Assistant Professor Dr. Ratchada Viriyapong for their kindness and useful comments and suggestions. Moreover, many thanks to the School of Mathematics, Institute of Science, Suranaree University of Technology (SUT) and all of my friends at SUT for their support.

Finally, I would like to thank the Development and Promotion of Science and Technology Talents Project (DPST) for all of the financial support.

Sunisa Tavaen

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS

Page

ABSTRACT IN THAI I
ABSTRACT IN ENGLISH II
ACKNOWLEDGEMENTS III
CONTENTS IV
LIST OF TABLES VII
LIST OF FIGURES IX

CHAPTER

I INTRODUCTION 1
1.1 Neural Networks (NNs) 1
1.2 Radial Basis Functions (RBFs) 2
1.2.1 Globally-Supported RBFs 4
1.2.2 Compactly-Supported RBFs 9
1.3 Radial Basis Function Neural Networks (RBF-NNs) 9
1.4 Shape Parameter of Radial Basis Function 11
1.4.1 Hardy Method 11
1.4.2 Franke Method 11
1.4.3 Carlson Method 11
1.4.4 Shape Parameter Strategy 12
1.5 Pattern Recognition 12
1.6 Interpolation 14
1.7 Research Objective 15
1.8 Scope and Limitations 16
1.9 Research Procedure 16
1.10 Expected Results 19
II MATHEMATICAL BACKGROUND 20

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS (Continued)

Page

2.1 Singular Value Decomposition (SVD) 20
2.2 The Pseudoinverse (or Moore-Penrose Inverse) of a Matrix 21
2.3 Interpolation by RBF-NNs........ 21
2.4 Shapeless Parameter of Radial Basis Function........ 22
2.5 The Representational Capability (RC)........ 24
2.6 Additive White Gaussian Noise (AWGN) 25
III COMPUTATIONAL COMPONENTS........ 27
3.1 The Main Algorithm 27
3.2 Node Distribution Manner 28
3.2.1 The Training Datasets........ 28
3.2.2 The Testing Datasets 30
3.3 Performance Criteria 30
IV PRELIMINARY EXPERIMENTS........ 32
4.1 Experiment 1 : A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs 32
4.1.1 The Project Objective 32
4.1.2 Globally-Supported RBFs 32
4.1.3 Compactly-Supported RBFs........ 33
4.1.4 Test Case 1: Interpolation Problem 33
4.1.5 Test Case 2: Poisson Problem with Non-rectangular domain 38
4.1.6 Test Case 3: Nonlinear Problem 40
4.1.7 Main Conclusion........ 43
4.2 Experiment 2: Performances of Non-Parameterised Radial Basis Functions in Pattern Recognition Applications........ 44
4.2.1 The Project Objective 44
4.2.2 The Non-Parameterised Radial Basis Functions 44
4.2.3 Test Case 1: Linear interpolation........ 44

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS (Continued)

Page

4.2.4 Test Case 2: Parabola function 50

4.2.5 Test Case 3: Sine interpolation 55

4.2.6 Main Conclusions 60

4.3 Experiment 3: A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks 60

4.3.1 The Project Objective 60

4.3.2 RBFs and Their Shape Parameter's Selection Strategy 60

4.3.3 Test Case 1: The Sine Function 62

4.3.4 Test Case 2: The Famous Franke's Function 65

4.3.5 Main Conclusion 72

4.4 Experiment 4: Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery 72

4.4.1 The Project Objective 72

4.4.2 Variable Shape Parameters 72

4.4.3 Test Case 1: One-Dimensional Case 75

4.4.4 Test Case 2: Two-Dimensional Case 76

4.4.5 Main Conclusions 80

V THE MAIN NUMERICAL EXPERIMENT 81

5.1 The Project Objective 81

5.2 Numerical Experiments and General Discussion 86

5.2.1 Experiment 1: Franke's function 86

5.2.2 Experiment 2: F7 92

5.3 Main Conclusions 98

VI CONCLUSIONS 100

REFERENCES 101

APPENDICES

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS (Continued)

Page

APPENDIX A MAIN CODES 107

A.1 Main Code 1: Experiment 108

A.2 Main Code 2: Experiment 3 Selected Main Codes 148

A.3 Main Code 3: The Main Numerical Experiment 153

APPENDIX B PUBLICATIONS (SCOPUS-INDEXED) 176

B.1 Experiment 1 177

B.2 Experiment 2 178

B.3 Experiment 3 179

B.4 Experiment 4 180

CURRICULUM VITAE 181

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF TABLES

Table Page

1.1 Type of Globally-Supported Radial Basis Function. 5

2.1 The popular Wendland's CS-RBFs....23

2.2 The popular Wu's CS-RBFs. 24

2.3 The popular Buhmann's CS-RBFs. 24

3.1 Error Norms adopted in this work. 30

4.1 L_{Abs} measured at different locations over the computational domain with 15×15 interpolation nodes/centres....36

4.2 Errors produced by each RBF when using different levels of nodes density 39

4.3 Numerical solutions comparison with the exacts 41

4.4 The locations over the computational domain with 100 points for linear trend case. 45

4.5 Training and validation errors for candidate models for linear trend case....47

4.6 Results comparison when using different numbers of centres and RBFs for linear trend case....48

4.7 Listing of basis function centres for linear trend case. 48

4.8 The locations over the computational domain with 100 points for parabola trend case. 50

4.9 Training and validation errors for candidate models for parabolas trend case 52

4.10 Results comparison when using different numbers of centres and RBFs for parabolas trend case 53

4.11 Listing of basis function centres for parabolas trend case. 53

4.12 The locations over the computational domain with 100 points for sine trend case. 55

4.13 Training and validation errors for candidate models for sine trend case....57

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF TABLES (Continued)

Table Page

4.14 Results comparison when using different numbers of centres and RBFs for sine trend case 58
4.15 Listing of basis function centres for sine trend case. 58
4.16 Error comparison for some chosen different ***TVN*** cases for sine function 63
4.17 Error comparison for some chosen different ***TVN*** cases for Frank's function 68
4.18 Mean absolute errors measured for approximation of the function and its derivative for all forms of generalized MQ when using different variable shape forms. 76
5.1 Type of radial basis function. 82
5.2 Training and testing errors measured for Franke's function at $SNR = 30$ 88
5.3 Training and testing errors measured for Franke's function at $SNR = 15$ 90
5.4 The number of basis functions (m) and the condition number $(Cond_{\infty}(\cdot))$ of all radial basis function candidates for the Franke's function test case 91
5.5 Training and testing errors measured for F7 at $SNR = 30$ 94
5.6 Training and testing errors measured for F7 at $SNR = 15$ 96
5.7 The number of basis functions (m) and the condition number $(Cond_{\infty}(\cdot))$ of all radial basis function candidates for the F7 test case. 97

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF FIGURES

Figure Page

1.1 The mapping of typical neural networks structure 1

1.2 The mapping of a typical radial basis function structure 3

1.3 Two dimensional curves for Multiquadric RBFs under this investigation at $c = 10.00$ 5

1.4 Two dimensional curves for Inverse Multiquadric RBFs under this investigation at $c = 10.00$. 6

1.5 Two dimensional curves for Gaussian RBFs under this investigation at $c = 10.00$. 6

1.6 Two dimensional curves for Thin Plate Splines RBFs. 7

1.7 One dimensional curve for type of Multiquadric RBFs with different shape values. 7

1.8 One dimensional curve for type of Inverse Multiquadric RBFs with different shape values 8

1.9 Radial basis function (RBF) neural networks structure. 10

1.10 Examples of handwritten digits 13

3.1 $(x, y)-$Node uniformly distribution manner for training datasets using 50×50 nodes. 29

3.2 $(x, y)-$Node non-uniformly distribution manner for training datasets using 50×50 nodes. 29

4.1 L_{RMS} measured at different values of shape parameters with 10×10 interpolation nodes/centres. 34

4.2 L_{RMS} measured at different values of shape parameters with 30×30 interpolation nodes/centres. 34

4.3 Solution approximation at selected locations obtained from each RBF type, compared again the exact solution. 37

4.4 Solution profile comparison between that produced by CS1-RBF and that of the exact one 37

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF FIGURES (Continued)

Figure Page

4.5 Density distribution at 64 nodes....39

4.6 Density distribution at 100 nodes 40

4.7 Node solution comparison between that of CS-RBF and the exact one. 40

4.8 Node being uniformly-distributed over the computational domain, together with supported nodes needed for nonlinear computing process....41

4.9 Solution comparisons by using MQ-RBF 42

4.10 Solution comparisons by using GA-RBF....42

4.11 Solution comparisons by using CS-RBF 43

4.12 Data with 100 points for linear trend case. 47

4.13 Predicted training trend produced by using 4 centres for linear trend case 49

4.14 Predicted validation trend produced by using 4 centres for linear trend case....49

4.15 Data with 100 points for parabolas trend case. 52

4.16 Predicted training trend produced by using 4 centres for parabolas trend case....54

4.17 Predicted validation trend produced by using 4 centres for parabolas trend case 54

4.18 Data with 100 points for sine trend case....57

4.19 Predicted training trend produced by using 4 centres for sine trend case 59

4.20 Predicted validation trend produced by using 4 centres for sine trend case....59

4.21 Pattern reconstruction obtained from using Gaussian RBF with all three choices of shape parameter compared with that produced by the shapeless CS1-RBF where training phase, calculated using *TVN* = (100:900) 65

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF FIGURES (Continued)

Figure Page

4.22 Pattern reconstruction obtained from using Gaussian RBF with all three choices of shape parameter compared with that produced by the shapeless CS1-RBF where validation phase, calculated using *TVN* = (100:900) 65

4.23 Node-distribution, projected on xy-plane, in the case with 121 training data points (depicted with blue dots) and 3600 validation data points depicted with black dots). 66

4.24 Surface plot of $f(x,y)$ 67

4.25 Pattern reconstruction yielded by using GA-***CS*** with the training phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 69

4.26 Pattern reconstruction yielded by using GA-***CS*** for the validation phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 69

4.27 Pattern reconstruction yielded by using MQ-***CS*** with the training phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 70

4.28 Pattern reconstruction yielded by using MQ-***CS*** for the validation phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 70

4.29 Pattern reconstruction yielded by using ***CS1*** with the training phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 71

4.30 Pattern reconstruction yielded by using ***CS1*** for the validation phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red) 71

4.31 Generalized MQ with six values of β's 73

4.32 Generalized distribution of centres test function 73

4.33 Generalized two-dimensional test function 73

4.34 Multiquadrics’ shape determining ***Stg-1*** by using $c_{\min} = 1/\sqrt{N}$, $c_{\max} = 3/\sqrt{N}$, and $N = 30$ 74

4.35 Multiquadrics’ shape determining ***Stg-2*** by using $c_{\min} = 1/\sqrt{N}$, $c_{\max} = 3/\sqrt{N}$, and $N = 30$ 75

LIST OF FIGURES (Continued)

Figure Page

4.36 Mean absolute errors measured at different numbers of centres for approximation of $f(x,y)$ by using ***Stg-1*** 77

4.37 Mean absolute errors measured at different numbers of centres for approximation of $f(x,y)$ by using ***Stg-2*** 77

4.38 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial x$ by using ***Stg-1***. 78

4.39 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial x$ by using ***Stg-2*** 78

4.40 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial y$ by using ***Stg-1*** 79

4.41 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial y$ by using ***Stg-2*** 79

5.1 One-dimensional curve for some type of RBFs under this investigation of those containing no shapes (or shapeless) and $2 \leq \phi\ r\ \leq 16$ 84

5.2 One-dimensional curve for some type of RBFs under this investigation of those containing no shapes (or shapeless) and $0.2 \leq \phi\ r\ \leq 2$ 84

5.3 One-dimensional curve for some types of RBFs under this investigation of those containing no shapes (or shapefree) and $0 \leq \phi\ r\ \leq 0.2$ 85

5.4 One-dimensional curve for the famous MQ with different shape values under this investigation. 85

5.5 Noised-$z(x,y)$ training datasets with $SNR = 30$ based on uniformly distribution of $\{(x,y)\}$ 87

5.6 Noised-$z(x,y)$ training datasets with $SNR = 30$ based on non-uniformly (randomly) distribution of $\{(x,y)\}$. 87

5.7 CPU-storage measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR = 30$ 88

5.8 CPU-time measurement observed at three different sizes of testing datasets as uniform nodes distribution using $SNR = 30$ 89

LIST OF FIGURES (Continued)

Figure **Page**

5.9 Noised-$z(x,y)$ training datasets with $SNR=15$ based on uniformly distribution of $\{(x,y)\}$ 89

5.10 Noised-$z(x,y)$ training datasets with $SNR=15$ based on non-uniformly (randomly) distribution of $\{(x,y)\}$. 90

5.11 CPU-storage measurement observed at three different sizes of testing datasets with random nodes distribution using $SNR=15$. 91

5.12 CPU-time measurement observed at three different sizes of testing datasets with random nodes distribution using $SNR=15$. 91

5.13 Noised-$z(x,y)$ training datasets based on non-uniformly (randomly) distribution of $\{(x,y)\}$ 93

5.14 Noised-$z(x,y)$ training datasets based on non-uniformly (randomly) distribution of $\{(x,y)\}$ 93

5.15 CPU-storage measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR=30$ 94

5.16 CPU-time measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR=30$ 95

5.17 Noised-$z(x,y)$ training datasets based on uniformly distribution of $\{(x,y)\}$ 95

5.18 Noised-$z(x,y)$ training datasets based non-uniformly (randomly) distribution of $\{(x,y)\}$ 96

B.1 A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs. 177

B.2 Performances of Non-Parameterised Radial Basis Functions in Pattern Recognition Applications. 178

B.3 A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks. 179

B.4 Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery. 180

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER I

INTRODUCTION

1.1 Neural Networks (NNs)

Neural networks are widely interesting in the computer science and technology industry. In current years, neural networks have been applied to search for the best solution to many problems, for instance, image recognition, facial recognition, voice recognition, speech recognition, optical character recognition (OCR) and natural language processing (NLP), etc. The basic definition of neural networks emulates the brain computational unit, which is a "neuron". The function of the brain is connected with neurons and then multiple connected neurons become neural networks.

The machine mimics the work of the brain, the basic unit of computation in neural networks, or the neuron and connected cell of the brain or in this is called a node or unit. This means that machines can learn and process data in the same way humans do. A typical architecture of a neural network contains three layers which are input layer, hidden layer and output layer as shown Figure 1.1.

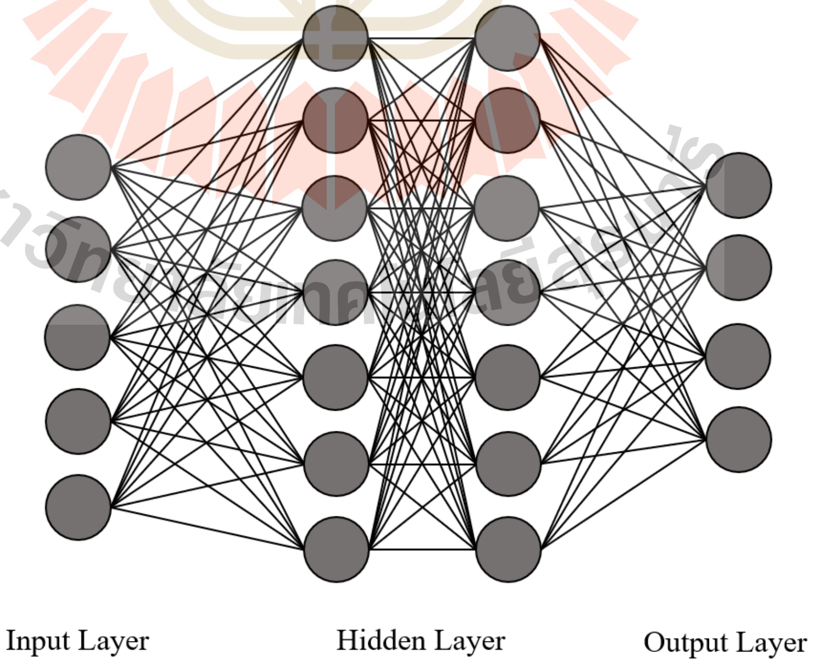


Figure 1.1 The mapping of typical neural networks structure.

มหาวิทยาลัยเทคโนโลยีสุรนารี

In these layers, there will always be input and output layers and it has zero or more hidden layers. The entire learning process of the neural networks is done with layers.

Input Layer: The input layer is the very beginning of the workflow for the neural network which contains input nodes. There are no calculations in this layer. In other words, this layer only passes input data from some external sources to the next layer. Each of the input data has to be numerical and if it is non-numerical, find a way to make it numerical first. The process of manipulating data before inputting it into the neural network is called data processing and oftentimes will be the most time-consuming part of making machine learning models.

Hidden Layer: The hidden layers are composed of most of the neurons in the neural network as superior to machine learning algorithms and are the heart of manipulating the data to get the desired output. They perform computations and transfer information from the input nodes to the output nodes. Then data will pass through the hidden layers and be manipulated by many weights and biases. It is called the "hidden" layer because developers of neural networks will not directly work with these layers, as opposed to input and output layers. There should be zero or more than zero hidden layers in the neural networks. For the large majority of problems one hidden layer is sufficient. In general, the more hidden layers in the neural network, the longer it takes to process for the output. Complex problems can be solved in the hidden layers of the neural network.

Output Layer: The output layer is the final product of manipulating the data in the neural network and can represent different things. There must be always an output layer in the neural networks. The output layer takes the inputs which are passed in from the layers before and performs the calculations through its neurons and then the output is computed. And the output layer has one node per one class label of our model.

มหาวิทยาลัยเทคโนโลยีสุรนารี

1.2 Radial Basis Functions (RBFs)

In the recent years, the application of radial basis functions has been greatly increasing in terms of classification and prediction tasks for examples of interesting contexts included in daily life (Chen, Li, Wang and Deng, 2019; Wang, Li and Miao, 2020; Dawson, 2020), agriculture (Hemageetha and Nasir, 2013; Shastry, Sanjay and Deexith, 2017), finance (Rashedi, Ismail, Hamadneh, Wadi, Jaber and Tahir, 2021), and medicine (Fragopoulos, Pouliakis, Meristoudis, Mastorakis and Margari, 2020).

Radial Basis Functions are commonly found as multivariate functions whose values are dependent only on the distance from the centre, i.e. $\phi\ \mathbf{x} = \phi\ r \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^d$ and $r \in \mathbb{R}$ or on the distance from centres point at j-location, $\mathbf{x}_j \in \mathbb{R}^d$ as $\phi\ \mathbf{x} - \mathbf{x}_j = \phi\ r_j \in \mathbb{R}$. The input vector $\mathbf{x}_i = x_1, x_2, \cdots, x_d \in \mathbb{R}^d$ of matrix $\mathbf{X}$ are input data and any function $\phi\ \mathbf{x} = \phi\ \|\mathbf{x}\|_2$ is the Euclidean distance. And then radial basis functions are completely described by specifying the number of basis functions, basis function parameters and the weights of the basis function outputs to produce the output data.

From a modeling perspective, it can be viewed as generating a sequence of two mappings. The first is a nonlinear mapping between input data and the basis function and the second is a linear mapping between the basis function and weight was generated from the model outputs.

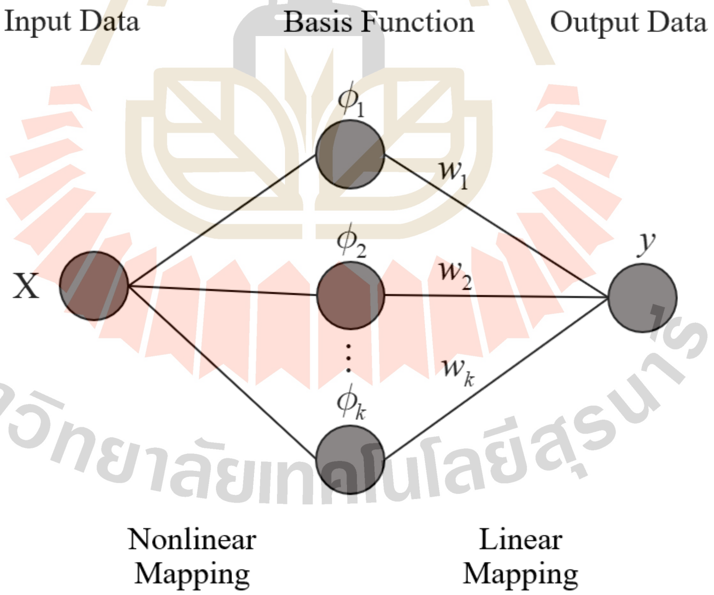


Figure 1.2 The mapping of a typical radial basis function structure.

มหาวิทยาลัยเทคโนโลยีสุรนารี

A typical RBF structure is illustrated in Figure 1.2 that shows the connectivity mapping of input data, basis function and output data. It is clear to understand that an input vector in d-dimensional space is transformed by the basis function into a k-dimensional vector. From the input vector $\mathbf{x} = x_1, x_2, \cdots, x_d$ pass the k basis functions and becomes the basis function output $\phi = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_k(\mathbf{x}))$ where $\phi_j(\cdot)$ is the output from the j-th basis function. The output y corresponding to the input vector $\mathbf{x}$ is obtained by the summation of the weighted basis function outputs, which are

$$y = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x}) \tag{1.1}$$

Formally, for a mapping $f : \mathbb{R}^d \to \mathbb{R}$ the radial basis function can be described mathematically as

$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi\left(\frac{\left\|\mathbf{x} - \boldsymbol{\mu}_j\right\|_2}{\sigma_j}\right), k \le N \tag{1.2}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $\boldsymbol{\mu}_j \in \mathbb{R}^d$ is the j-th basis function centres, $\phi(\cdot)$ is the basis function that is the nonlinear function from $\mathbb{R}$ to $\mathbb{R}$ and w_j is the weight associated with the j-th basis function output. Also, σ_j is the parameter to control the receptive field width of the j-th basis function.

The basis functions $\phi(\cdot)$ in RBF networks play the role of transfer functions in traditional neural networks. However, the basis functions have the unique feature that their responses to the input vectors are not only radially symmetric but also monotonically decreasing or increasing with distance r from the center μ_j. When $r = \|\mathbf{x} - \mu\|/\sigma$, several forms of these functions $\phi(r)$ can be used.

1.2.1 Globally-Supported RBFs

The interest of this study is the RBF interpolation of a continuous multivariate function, $f(\mathbf{x}) = \mathbf{x} \in \Omega \subset \mathbb{R}^d$, Ω is a bounded domain. Given N basis functions of output values $\{y_i\}_{i=1}^{N}$, when $y_i \in \mathbb{R}$ at data location $\{\mathbf{x}_i\}_{i=1}^{N} \in \Omega$. Then the traditional RBFs is Globally-Supported RBFs that defined with N- th unknown basis functions, which related

$$f(\mathbf{x}) \approx \sum_{j=1}^{N} w_j \phi_j\left(\left\|\mathbf{x} - \mathbf{x}_j\right\|_2\right) \tag{1.3}$$

where $\{w_j\}_{j=1}^{N}$ are the unknown coefficients to be determined.

Some popular choices and their expression are given in Table 1.1, where c is a positive constant (Powell, 1987). Amongst these, the Gaussian is probably the most popular basis function for two reasons that it has attractive mathematical properties and its hill-like shape is easy to control with a parameter σ (Winston, 1993).

มหาวิทยาลัยเทคโนโลยีสุรนารี

The basis functions are radially symmetric with distance r from each input vector $\mathbf{x}_j$ when $r_j = \left\|\mathbf{x}-\mathbf{x}_j\right\|_2$.

Table 1.1 Type of Globally-Supported Radial Basis Function.

RBFs	$\phi \; r$
Polynomial Spline (PS)	$\begin{cases} r^{2k-1}, k\in\mathbb{N} \\ r^{2k}\ln \; r \;, k\in\mathbb{N}\end{cases}$
Thin Plate Splines (TPS)	$r^2 \ln r$
Multiquadric (MQ)	$r^2+c^2 \; ^{\beta}, \beta\in\mathbb{R}$ such $\beta>0$
Inverse Multiquadric	$r^2+c^2 \; ^{-\beta}, \beta\in\mathbb{R}$ such $\beta>0$
Gaussian	$\exp \; - \; cr \; ^2$

Figure 1.3 – Figure 1.6 illustrate their surface profiles of at $c=10.00$. Differences in the curve profiles for each parameter value can be seen and noticed in Figure 1.7 and Figure 1.8.

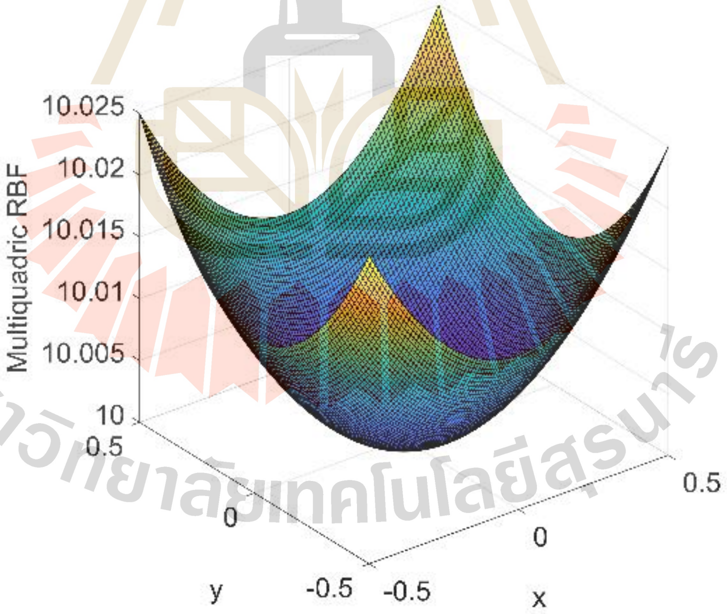


Figure 1.3 Two dimensional curves for Multiquadric RBFs under this investigation at $c=10.00$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

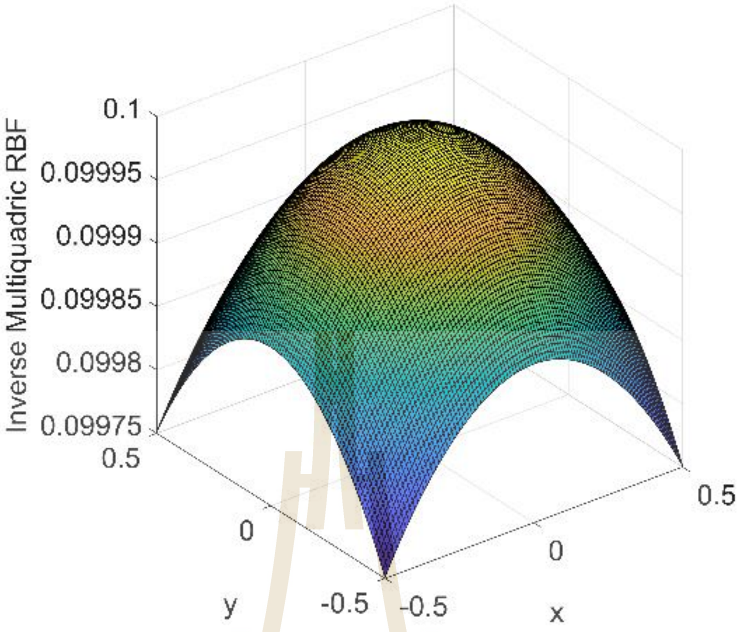


Figure 1.4 Two dimensional curves for Inverse Multiquadric RBFs under this investigation at $c = 10.00$.

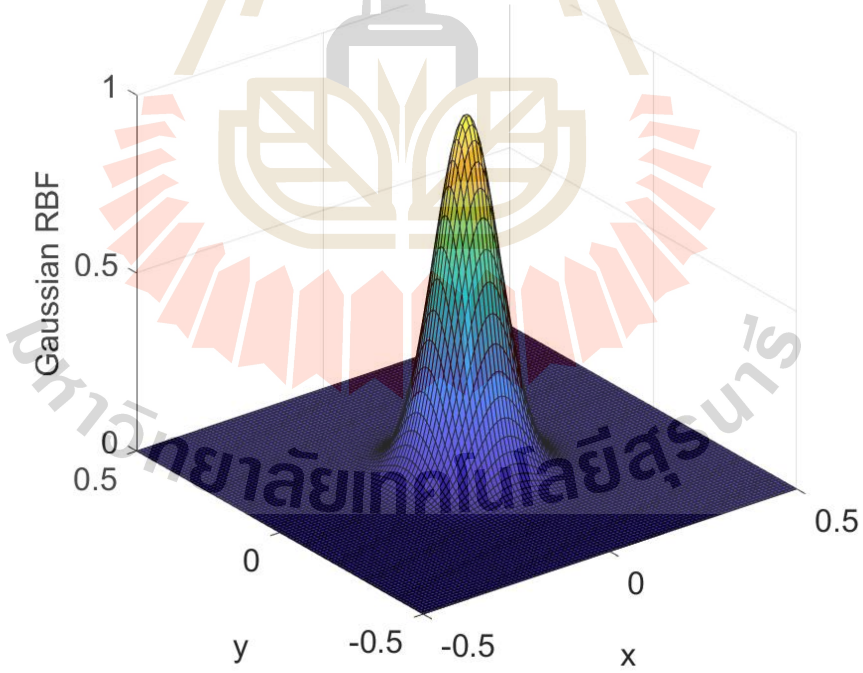


Figure 1.5 Two dimensional curves for Gaussian RBFs under this investigation at $c = 10.00$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

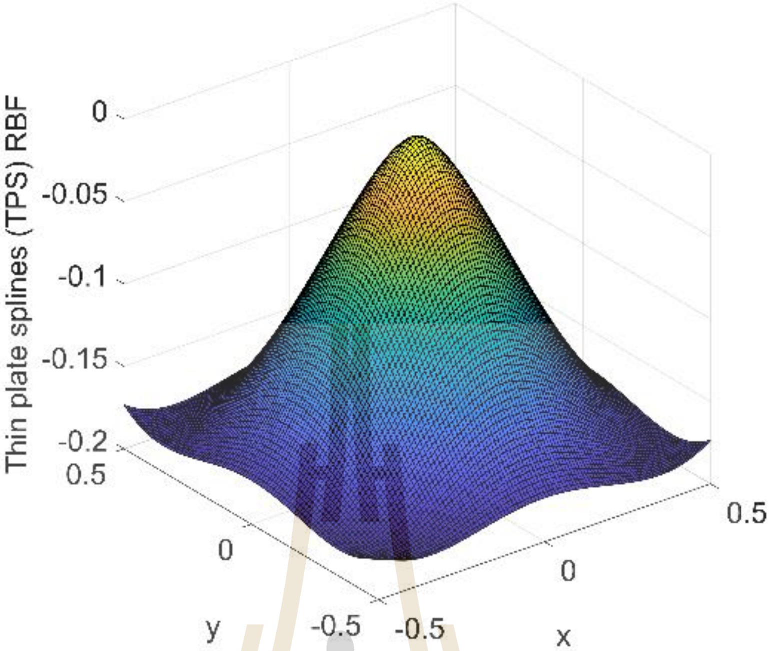


Figure 1.6 Two dimensional curves for Thin Plate Splines RBFs.

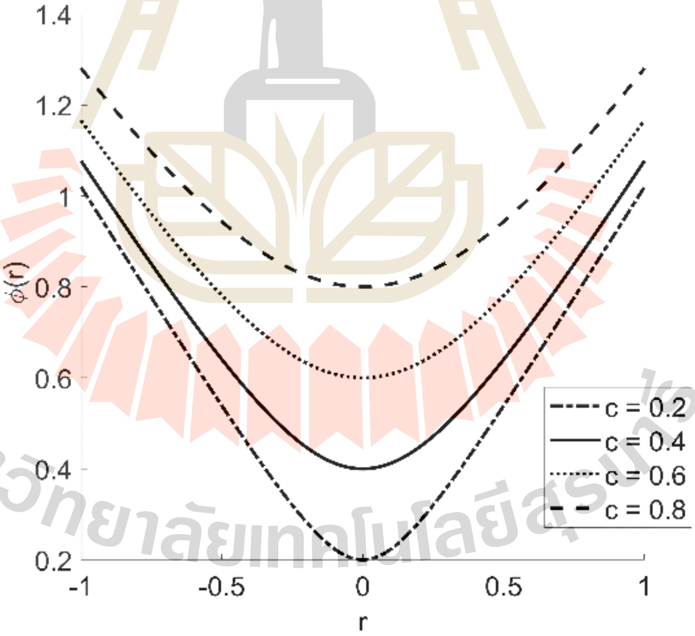


Figure 1.7 One dimensional curve for type of Multiquadric RBFs with different shape values.

มหาวิทยาลัยเทคโนโลยีสุรนารี

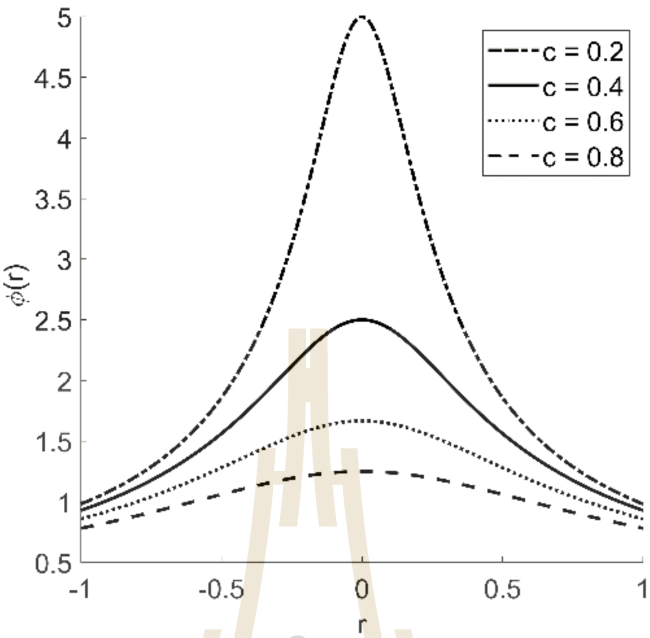


Figure 1.8 One dimensional curve for type of Inverse Multiquadric RBFs with different shape values.

From linear combination equation (1.3), we have

$$y_i = f(\mathbf{x}_i) = \sum_{j=1}^{N} w_j \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|_2\right), \quad i = 1, 2, \ldots, N. \tag{1.4}$$

The linear system of equations can be rewritten in the following matrix form,

$$\mathbf{Aw} = \mathbf{b}, \tag{1.5}$$

in which $\mathbf{w} = (w_1, w_2, \ldots, w_N)^T$ is an unknown coefficient vector to be determined, $\mathbf{b} = (y_1, y_2, \ldots, y_N)^T$ is the right-hand side vector, and the RBF matrix is given by

$$\mathbf{A} = \left[\phi_{ij}\right] = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \cdots & \phi_{1j} \\ \phi_{21} & \phi_{22} & \phi_{23} & \cdots & \phi_{2j} \\ \phi_{31} & \phi_{32} & \phi_{33} & \cdots & \phi_{3j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{i1} & \phi_{i2} & \phi_{i3} & \cdots & \phi_{ij} \end{bmatrix} \tag{1.6}$$

where $\phi_{ij} = \phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|_2\right), i, j = 1, 2, \ldots, N$.

In recent years, the optimal parameter c is witnessed by the continued efforts of many to establish the theory of evaluating. Since the condition number of the interpolation matrix dramatical grows as the parameter c increases, the optimal parameter c is the largest value at which it can be utilized before the instability of matrix calculation occurs due to the machine precision.

มหาวิทยาลัยเทคโนโลยีสุรนารี

1.2.2 Compactly-Supported RBFs

The Globally-Supported RBFs took a lot of computing space and time, so Wu and Wendland have introduced a new method that has been adjusted to be more local. Therefore, defined as compactly supported positive definite RBFs (CS-RBFs). From the linear combination equation (1.4) which are;

$$y_i = f(\mathbf{x}_i) = \sum_{j=1}^{N} w_j \phi(r_{ij}) \quad ,i = 1,2,\ldots,N \tag{1.7}$$

where $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ or written in general form as $\phi(r)$. For constructed radial basis function with compact support are of the form

$$\phi(r) = \begin{cases} p(r) & ,0 \le r \le 1 \\ 0 & ,otherwise. \end{cases} \tag{1.8}$$

with a univariate polynomial p.

Therefore, the equations can be rewritten in the following matrix form,

$$y_i = f(\mathbf{x}_i) = \begin{cases} \sum_{j=1}^{N} w_j \phi_j(r) & ,0 \le r \le 1 \\ 0 & ,otherwise \end{cases} \tag{1.9}$$

1.3 Radial Basis Function Neural Networks (RBF-NNs)

In this section, first of all is introduced the basic neural networks structure of RBF model with some basis function $\phi(r)$ and give expression for the mappings which converts the input data vectors to model inputs that divided into three layers as shown in Figure 1. and can be described as follow:

มหาวิทยาลัยเทคโนโลยีสุรนารี

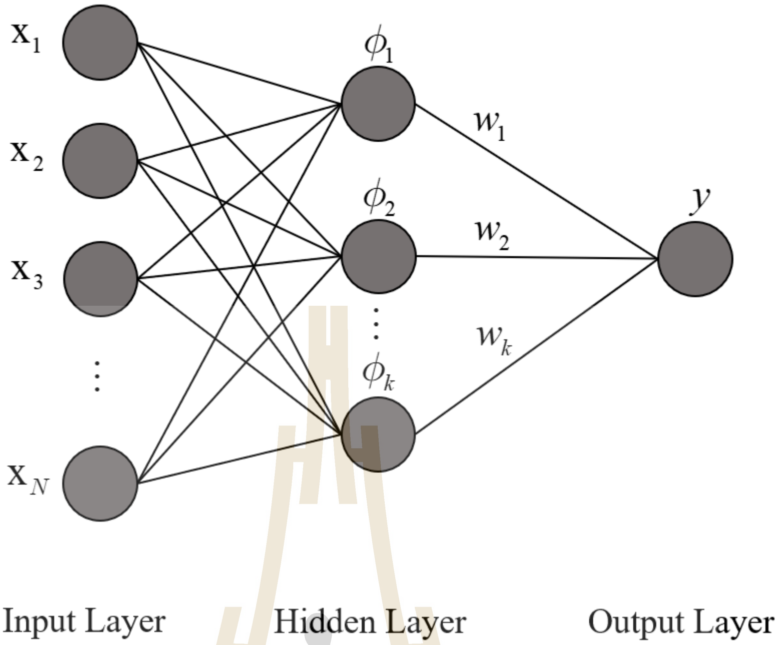


Figure 1.9 Radial basis function (RBF) neural networks structure.

Input Layer: The input layer forwards the data to the hidden layer. The number of neurons in the input layer should be equal to the dimensions of the data. In the input layer, there are no calculations the same as a standard neural network. The input neuron is completely connected to the hidden neuron.

Hidden Layer: The hidden layer takes the input layer to the radial basis function which might not be a linear function and transform it into a new space that is a more linear function. The computations in the hidden layers are based on comparisons with prototype vectors of radial basis function which is a vectors from the training set.

Output Layer: The output layer is a linear function for both classification or prediction tasks. The computations in the output layer are performed in a linear combination between the input vector and the weight vector with calculated in hidden layer by the radial basis function.

มหาวิทยาลัยเทคโนโลยีสุรนารี

1.4 Shape Parameter of Radial Basis Function

In the Globally-Supported RBFs contains shape parameter that is very important problem. Nowadays, the optimal shape parameter is a challenging problem. This research introduces some popular methods how to choose the shape parameter.

1.4.1 Hardy Method

The first method introduced by Hardy (Hardy, 1971), The Hardy method was originally used with the Multiquadric basis function and (Hardy, 1977) indicating the best value for

$$\varepsilon = 0.815d \tag{1.10}$$

where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$ (the mean distance of each data point to its nearest neighbor) and d_i is the distance from each data point x_i, y_i to the nearest neighbor.

1.4.2 Franke Method

The method was first proposed by Franke (Franke, 1979) which is developed from Hardy's method and changed nominal values. Nevertheless, this method considers the boundary cycle of the data set that it is diameter D and defined as:

$$\varepsilon = \frac{1.25D}{\sqrt{N}} \tag{1.11}$$

where D is the diameter of the smallest circle containing all data points and N is the number of data point. The beginning used for quadratic basis function.

1.4.3 Carlson Method

This method was invested by Carlson (Carlson and Foley, 1991). The second method previously computed only domain however Carlson's method includes range to compute. Start with calculating the minimum and maximum values of the data points x_i, y_i, z_i and next compute the least squares bivariate quadratic polynomial fit to the data $\bar{x}_i, \bar{y}_i, \bar{z}_i$. Defined as:

$$\bar{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{1.12}$$

$$\bar{y}_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}, \tag{1.13}$$

$$\bar{z}_i = \frac{z_i - z_{\min}}{z_{\max} - z_{\min}}. \tag{1.14}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Next, denoting this quadratic by $q(\bar{x},\bar{y})$ and compute

$$V=\sum_{i=1}^{N}\frac{\left(\bar{z}_i-q(\bar{x}_i,\bar{y}_i)\right)^2}{N}. \tag{1.15}$$

Finally, find form of shape parameter as

$$\varepsilon=\frac{1}{1+120V}. \tag{1.16}$$

And the next variables of shape parameter are two nonlinear-variable shape strategies following.

1.4.4 Shape Parameter Strategy

This method was introduced by Nojavan (Nojavan, Abbasbandy and Allahviranloo, 2017). The variable shape parameter strategy which is based on a variably scaled radial kernel that can be defined as:

$$K_c(\mathbf{x}-\mathbf{x}_j)=\phi\left((r_j/c_j)^2\right), r_j=\|\mathbf{x}-\mathbf{x}_j\|_2, j=1,2,\ldots,N \tag{1.17}$$

where c_j is the shape parameter corresponding to the j-th center.

In this paper, we have proposed some strategies to choose, defined as

$$\varepsilon_j=\left(c_{\min}+(c_{\max}-c_{\min})\exp(-j)\right)^{-1}, j=1,2,\ldots,N \tag{1.18}$$

and

$$\varepsilon_j=\left(c_{\min}+(c_{\max}-c_{\min})\sin(j)\right), j=1,2,\ldots,N \tag{1.19}$$

where $c_{\max}=3/\sqrt{N}$, and $c_{\min}=1/\sqrt{N}$, are positive parameters.

1.5 Pattern Recognition

Pattern recognition has a long history of being used which has originated in engineering, whereas in the recent year that was used in machine learning and grew out of computer science. Human ability is the recognition and classification from observation. Pattern recognition is the science of recognizing patterns by machines. Nowadays, this is a big wide research area in Artificial Intelligence Science (Parasher, Sharma, Sharma and Gupta, 2011). There are several useful areas, such as biomedical and biology, social medial intelligence (SMI), video surveillance, intelligent retail environment, and digital cultural heritage (Paolanti and Frontoni, 2020).

The task of pattern recognition is to construct a model that captures an unknown input-output mapping on the basis of limited evidence about the nature of this mapping, called training. The evidence available is a set of labelled training data, also called training samples. The goal, however, is not to learn an exact representation of

มหาวิทยาลัยเทคโนโลยีสุรนารี

the training data itself but to build a model that captures the underlying relationship in the training data, so that it can be used to predict the unknown output at some future observations of the input. In the literature, this latter ability is called generalization capability, a term borrowed from psychology.

The process of pattern recognition is successful to adapt and learn from examples which are underlying idea for generating system the predictive learning. The problem of pattern recognition is searching for patterns in the data. For instance, recognizing handwritten digits from the MNIST data set [http://yann.lecun.com/exdb/mnist/] illustrated in Figure 1.10, each digit contains a 28x28 pixel image.

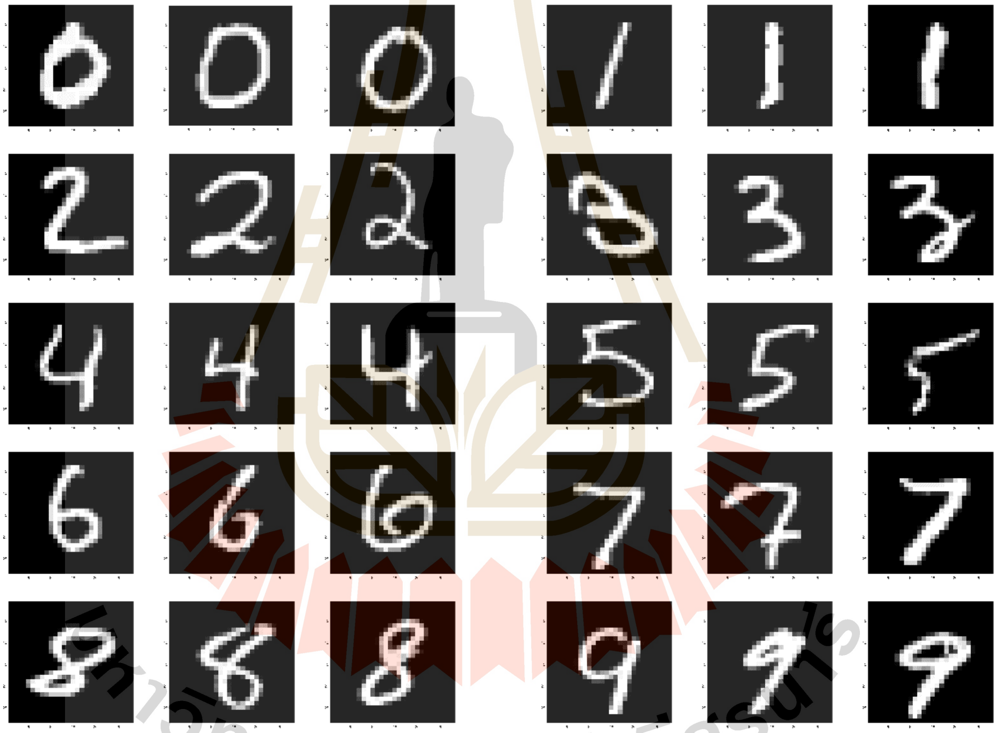


Figure 1.10 Examples of handwritten digits.

มหาวิทยาลัยเทคโนโลยีสุรนารี

The goal of this problem is to build a produce of machine system to identify of the digit 0,1,...,9 as the output from the $\mathbf{X}$ vectors input. That is the nonminor problem due to the fact that specific characteristic of handwriting that can be tackled by handcrafted rules or behavior analysis the digits based on the shapes or format of the strokes.

Pattern recognition is the process of differentiating and dividing the data according to certain criteria or by general components, which are performed by special algorithms. The task of pattern recognition is to construct the model with unknown input-output mapping pattern. In other words, it is to construct the best model, if any, from the train data with some mapping functions and expect this model to best represent the rest of the data, called 'training data'. Both sets of the data can be of the following form;

$$D=\left\{(\mathbf{x}_i,y_i)\mid \mathbf{x}_i\in\mathbb{R}^d, y_i\in\mathbb{R}, i=1,2,\cdots,N\right\} \tag{1.20}$$

where $\mathbf{x}_i$ are inputs with the corresponding y_i are outputs. The main task is to find a mapping D from the d- dimensional input space to 1-dimensional output space.

Model structure of a typical radial basis function network is depicted in Figure 1., when a data set $\left\{(\mathbf{x}_i,y_i)\right\}_{i=1}^{n}$ is given and the model searches for the corresponding output estimate $\hat{y}$ for input vector $\mathbf{x}$, represented by functional form:

$$\hat{y}=f(\mathbf{x})=\sum_{j=1}^{m}w_j\phi_j(\mathbf{x})=\sum_{j=1}^{m}w_j\phi_j\left(\left\|\mathbf{x}-\mu_j\right\|_2/\sigma\right) \tag{1.21}$$

where $\|\cdot\|_2$ is the Euclidean distance norm, $\phi(\cdot)$ is a basis function and m is the number of centres, σ is the parameter to control the receptive field width of the j-th basis functions, μ_j are the centres of j-th basis functions, and w_j are the weight associated with the j-th basis functions (note that $m\le n$). Thus, the RBF model is fully determined by $P=(m,\mu,\sigma,w)$.

1.6 Interpolation

Interpolation is a subset of statistical methods to estimate an unknown function by using other established values located by a given discrete data set that the function passes through the provided data points. This work focuses on the radial basis function that the original raised to be used for the interpolation problem. The RBF maps input data vectors $\mathbf{x}$ in $\mathbb{R}^d$ to output value y in $\mathbb{R}$. Then the domain of the RBF is $\mathbb{R}^d$ and the range is $\mathbb{R}$. Suppose that given a set of N input-output data points $\left\{\mathbf{x}_i,y_i\right\}_{i=1}^{N}$ is obtained, in which $\mathbf{x}_i\in\mathbb{R}^d$, $y_i\in\mathbb{R}$ and all data vectors are distinct. Therefore, the functional relationship between input vectors $\mathbf{x}_i$ and output vector y_i is

$$y_i=h(\mathbf{x}_i),\ i=1,2,\ldots,N. \tag{1.22}$$

where y_i are the exact output and the interpolation problem is to find the RBF $h(\mathbf{x}_i)$ that estimate at the vectors $\mathbf{x}_i, i=1,2,\ldots,N.$ The radial basis function model is contained with N basis function and N is the number of data vectors and all distinct. When used for interpolation, the number of centres equals the number of data points,

มหาวิทยาลัยเทคโนโลยีสุรนารี

i.e., $k = N.$ On the other hand, when the RBF model is employed for function approximation, usually the number of centres is much smaller than N The interpolation matrix which is related to the interpolation problem is analyzed at some length.

It should be noted that in general, the output could consist of several variables $y^1, y^2, \ldots, y^p$. Though this generalization is straight forward, we discuss only the one output variable case in this work.

Then the RBF model can be written in the form

$$y_i = f \;\; \mathbf{x}_i \;\; = \sum_{j=1}^{N} w_j \phi \;\; \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2 \;\; , i = 1, 2, \ldots, N \tag{1.23}$$

where the number of centres is equal to the number of data vectors which are $k = N$. Let us denote the new matrix by $\mathbf{G}$ and its columns by $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_N$ for the Gaussian radial basis functions and their shape parameter are $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N$ respectively, therefore the interpolation matrix is

$$\mathbf{G} = \left[\boldsymbol{g}_1 \quad \boldsymbol{g}_2 \quad \cdots \quad \boldsymbol{g}_j \quad \cdots \quad \boldsymbol{g}_N \right] \tag{1.24}$$

where $\boldsymbol{g}_j = \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{x}_j \right\|^2}{2\varepsilon_j^2} \right)$. We get the interpolation matrix $\mathbf{G}$ is a square matrix of size $N \times N.$

Moreover, the N basis functions have the centres at $\mathbf{x}_1, \mathbf{x}_2, \cdots,$ and $\mathbf{x}_N$ and the output of the RBF mapping given in the form of the interpolation matrix as

$$f \;\; \mathbf{x} \qquad {}^{N}_{j} \; w_j \exp \;\; \frac{\left\| \mathbf{x} \quad \mathbf{x}_j \right\|^2}{2\varepsilon_j^2} \; . \tag{1.25}$$

[redacted]

Furthermore, we can be written in matrix form based on the interpolation matrix as following

$$\mathbf{G}\mathbf{w} = \mathbf{y}. \tag{1.26}$$

1.7 Research Objective

The main purpose of this study is to investigate how well radial basis functions containing no shape parameters can perform when being utilized in pattern recognition applications under the structure of neural networks. Its ability is tested with large scattered data sets where the performances are carefully monitored and assessed via several criteria; accuracy, sensitivity to parameters, CPU-time, storage requirement, and ease of implementation.

มหาวิทยาลัยเทคโนโลยีสุรนารี

1.8 Scope and Limitations

Below are scope and limitations of this study.

1. Shapeless radial basis functions being considered are only those stated in Section 2.4

2. Input data is mainly in 2-dimension format.

3. Programming language used is mainly MATLAB.

4. For result validation, the mean square error norms (MSE) is employed and it is of the following form;

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \tilde{y}_i\right)^2, \tag{1.27}$$

with N being the number of data points involved in each case, y_i, $\tilde{y}_i$ are respectively the actual and approximated value of the output at a node i^{th}.

5. All computational processes are executed in serial computation manner, one task will be done after another, with the following information;

- Model: Swift SF514-54GT
- Processor: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
- Installed memory(RAM): 16.0 GB(15.8 GB usable)
- System type: 64-bit Operating System, x64-based processor.

1.9 Research Procedure

Before reaching the main experiment of the thesis, five investigations have successfully been carried out, all of which have already been published in a Scopus-indexed journal, as details listed below (See also the APPENDICES).

Project-1:

Title: A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs.

Abstract: It is known that all Radial Basis Function-based meshfree methods suffer from a lack of reliable judgement on the choice of shape parameter, appearing in most of the RBFs. While the popularity of meshfree/meshless numerical methods is growing fast over the past decade, the great challenge is still to find an optimal RBF form with its optimal shape parameter. In this work, the main focus is on one type of RBF namely ‘Compactly Supported (CS-RBF)’ that contains no parameter, and yet has

มหาวิทยาลัยเทคโนโลยีสุรนารี

not been explored numerically as much in the past, particularly under the context of data interpolation/approximation and solving partial differential equations (PDEs). To compare the potential advantages of CS-RBF, two most popular choices of RBF widely used; Multiquadric (MQ), and Gaussian (GA) were studied parallelly. The information gathered and presented in this work shall be useful for the future users in making decision on RBF.

Publication status: IOP Conf. Series: Journal of Physics: Conf. Series **1489** (2020) 012020. (Scopus)

Project-2 :

Title: Performances of non-parameterised radial basis functions in pattern recognition applications.

Abstract: Pattern recognition appears in many applications with most popular scheme are those involved the so-called 'Radial Basis Function (RBF)'. It is known that the shape parameter contained in some RBFs used has great influence on the final quality of prediction. This study focusses on RBFs which contains no parameters where three data patterns are used for performance validation. With a good choice of number of centres, it is clearly possible to obtain satisfactory results with no burden on choosing the suitable or optimal shape. This can well shed more light into applications with more complexity with less user's judgment and be more automatic in the process.

Publication status: IOP Conf. Series: Journal of Physics: Conf. Series **1706** (2020) 012165. (Scopus)

Project-3:

Title: A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks.

Abstract: This study investigates three choices of shape parameter selection when the so-called Radial Basis Function (RBF) is used. Under the problem of pattern recognition via RBF-Neural Network using RC-algorithm, three RBFs are focussed on; Gaussian (GA), Multiquadric (MQ), and Compactly-Supported (CS1). Two pattern recognition cases are tested and the best choice of shape parameter is validated using Model-Selection Criteria (MSC).

มหาวิทยาลัยเทคโนโลยีสุรนารี

Publication status: IOP Conf. Series: Journal of Physics: Conf. Series (Scopus)

Project-4 :

Title: Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery.

Abstract: After being introduced to approximate two-dimensional geographical surfaces in 1971, the multivariate radial basis functions (RBFs) have been receiving a great amount of attention from scientists and engineers. In 1987 the idea was extended into the construction of neural networks corresponding to the beginning of the era of artificial intelligence, forming what is now called 'Radial Basis Function Neural Networks (RBFNs)'. Ever since, RBFNs have been developed and applied to a wide variety of problems; approximation, interpolation, classification, prediction, in nowadays science, engineering, and medicine. This also includes numerically solving partial differential equations (PDEs), another essential branch of RBFNs under the name of the 'Meshfree/Meshless' method. Amongst many, the so-called 'Multiquadric (MQ)' is known as one of the mostly-used forms of RBFs and yet only a couple of its versions have been extensively studied. This study aims to extend the idea toward more general forms of MQ. At the same time, the key factor playing a very crucial role for MQ called 'shape parameter' (where selecting a reliable one remains an open problem until now) is also under investigation. The scheme was applied to tackle the problem of function recovery as well as an approximation of its derivatives using six forms of MQ with two choices of the variable shape parameter. The numerical results obtained in this study shall provide useful information on selecting both a suitable form of MQ and a reliable choice of MQ-shape for further applications in general.

Publication status: IOS Press Ebooks: Fuzzy Systems and Data Mining VII Series: Frontiers in Artificial Intelligence and Applications (Scopus)

Project-5 :

Title: A modified local distance-weighted (MLD) method of interpolation and its numerical performances for scattered large datasets.

Abstract: The purpose of this study is to propose an interpolation scheme that is designed to remedy shortcomings encountered in two popular interpolation

มหาวิทยาลัยเทคโนโลยีสุรนารี

methods; the triangle-based blending (TBB) and the inverse distance weighed (IDW). At the same time, the proposed method combines their derisible aspects; the local nature and free of quadratic surface construction, making it comparatively less time-consuming and more independent of the global effect. For these properties, it is named as 'Modified Local Distance-weighted (MLD)' method and is being tested in detail with different sizes of datasets. Datasets involved in this investigation are both synthetic datasets and real datasets (from some public datasets available) using both uniform and non-uniform node distributions. The performances are being carefully monitored and assessed via several criteria; accuracy, sensitivity to parameters, CPU-time, storage requirement, and ease of implementation. For comparative purposes, three alternative methods are also involved; TBB, IDW, and Radial Basis Function-based (RBF) method. The anticipated outcome is an effective mathematical tool for interpolating data in large scales with acceptable accuracy while requiring less computational efforts. The findings would definitely be useful for a wide range of applications; industry, economy, education, healthcare, environment, and many more.

Publication status: Current Applied Science and Technology Via. the 25th Annual Meeting in Mathematics 2021 (AMM 2021).

Further steps of investigation shall involve more forms of shapefree radial basis functions which will be mentioned in Section 2.4 and larger data sets (both synthetic datasets and real datasets) in scattered manner shall be considered.

1.10 Expected Results

Some insights of performances of some meshfree RBFs are discovered in terms of accuracy, sensitivity to parameters, CPU-time, storage requirement, and ease of implementation. This shall lead to alternative choices for practical applications in many areas; engineering, industry, economy, education, healthcare, environment, and many more.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER II

MATHEMATICAL BACKGROUND

2.1 Singular Value Decomposition (SVD)

This is a one of widely technique used to factorization matrix into three matrices that often used to find the rank of matrix. In this work is an important technique used by theorem.

Theorem 2.1 (Singular Value Decomposition) If $\mathbf{G}$ is a real $N \times N$ interpolation

$$\mathbf{G} = U \sum V^T \tag{2.1}$$

matrix, then there decompose into the product of three matrices which are:

where U and V are an orthogonal $N \times N$ matrices with the property that $UU^T = I$ and $VV^T = I$, and $\sum$ is a diagonal matrix with positive numbers $s_1, s_2, \cdots, s_N$ on the diagonal.

Therefore, the orthogonal matrices $U = \mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N \in \mathbb{R}^{N \times N}$ and $V = \mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N \in \mathbb{R}^{N \times N}$ such that $\sum = diag\ s_1, s_2, \cdots, s_N \in \mathbb{R}^{N \times N}$ where $s_1 \geq s_2 \geq \cdots \geq s_N \geq 0$. The s_i are called i-th singular values of matrix $\mathbf{G}$ and the column vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are called i-th left singular vector and i-th right singular vector, respectively.

The importance of this section to interest in rank of the matrix that the rank of $\mathbf{G}$ is defined as the number of non-zero singular values defined as r by

$$s_1 \geq s_2 \geq \cdots \geq s_r > s_{r+1} = \cdots = s_N = 0, \tag{2.2}$$

Then

$$\text{rank}(G) = r, \tag{2.3}$$

And from the rank of matrix which is important to show related matrices to vectors.

มหาวิทยาลัยเทคโนโลยีสุรนารี

2.2 The Pseudoinverse (or Moore-Penrose Inverse) of a Matrix

The pseudoinverse or Moore-Penrose Inverse defined as:

$$pinv(\phi) = (\phi^T\phi)^{-1}\phi^T \tag{2.4}$$

for a system of linear equations $\phi\mathbf{w}=\mathbf{y}$ where ϕ is the interpolation matrix of the basis function where ϕ is an $N\times m$ matrix that can be explained the least squares solution to any system of linear equations.

2.3 Interpolation by RBF-NNs

From radial basis function model is a function which maps input data vectors $\mathbf{x}\in\mathbb{R}^d$ to output values $y\in\mathbb{R}$. Considered, the domain of RBF model $\mathbb{R}^d$ and the range is $\mathbb{R}$. Additionally, the RBF models was used for interpolation problem. In truth, in the past few years, this model is used in neural networks that are very famous computing systems today. Therefore, the interpolation is carried out according to the structure of the neural networks as follows:

Input Layer: This layer consists the input data $\mathbf{x}_i=(x_1,x_2,\cdots,x_d)\in\mathbb{R}^d$ of matrix $\mathbf{X}$, i.e.,

$$\mathbf{X}=[\mathbf{x}_1\quad \mathbf{x}_2\quad \cdots\quad \mathbf{x}_N]^T\in\mathbb{R}^{N\times d} \tag{2.5}$$

where $\mathbf{x}_i$ are d-dimensional input data vectors, i.e., features of input data and N is the number of input vectors, i.e., size of input data.

Hidden Layer: Suppose that the hidden layer consists of k basis function ϕ_j · $j=1,2,\cdots,k$. These functions $\phi_j(\mathbf{x})=\phi\left(\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|_2/\sigma_j\right)$, $j=1,2,\ldots,k$ transform the input data matrix via linear and nonlinear mapping based on the Euclidean distance between the input vector $\mathbf{x}$ and prototype vector $\boldsymbol{\mu}_j$. The $N\times d$ input matrix is transformed by the k basis function into the following $N\times k$ matrix ϕ whose j-th column represents the outputs from the j-th basis function, $j=1,2,\cdots,k$.

$$\phi=\begin{array}{c}
\begin{array}{ccccc} \phi_1 & \cdots & \phi_j & \cdots & \phi_k \end{array}\\
\begin{bmatrix}
\phi_1(\mathbf{x}_1) & \cdots & \phi_j(\mathbf{x}_1) & \cdots & \phi_k(\mathbf{x}_1)\\
\phi_1(\mathbf{x}_2) & \cdots & \phi_j(\mathbf{x}_2) & \cdots & \phi_k(\mathbf{x}_2)\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
\phi_1(\mathbf{x}_N) & \cdots & \phi_j(\mathbf{x}_N) & \cdots & \phi_k(\mathbf{x}_N)
\end{bmatrix}_{N\times k}
\end{array} \tag{2.6}$$

For ϕ is the basis function from Section 1.2. As the Gaussian RBF, we have $\phi(r)=\exp(-r^2/2)$ so that the expression for the j-th function mapping can be explicitly written as

มหาวิทยาลัยเทคโนโลยีสุรนารี

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|^2}{2\sigma_j^2}\right), \tag{2.8}$$

where $\boldsymbol{\mu}_j$ is the center and σ_j is the width of the j-th basis function. On substituting for $\phi_j(\cdot)$ in Equation (2.7), we get the following expression for ϕ.

$$\phi = \begin{bmatrix} \exp\left(-\frac{\left\|\mathbf{x}_1-\boldsymbol{\mu}_1\right\|^2}{2\sigma_1^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_1-\boldsymbol{\mu}_j\right\|^2}{2\sigma_j^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_1-\boldsymbol{\mu}_k\right\|^2}{2\sigma_k^2}\right) \\ \exp\left(-\frac{\left\|\mathbf{x}_2-\boldsymbol{\mu}_1\right\|^2}{2\sigma_1^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_2-\boldsymbol{\mu}_j\right\|^2}{2\sigma_j^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_2-\boldsymbol{\mu}_k\right\|^2}{2\sigma_k^2}\right) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \exp\left(-\frac{\left\|\mathbf{x}_n-\boldsymbol{\mu}_1\right\|^2}{2\sigma_1^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_n-\boldsymbol{\mu}_j\right\|^2}{2\sigma_j^2}\right) & \cdots & \exp\left(-\frac{\left\|\mathbf{x}_n-\boldsymbol{\mu}_k\right\|^2}{2\sigma_k^2}\right) \end{bmatrix} \tag{2.9}$$

Output Layer: From mapping of the input data was produced by k basis function in the hidden layer. Imports in this matrix are combined linearly according to weights $w_1, w_2, \cdots, w_k$ associated with the k basis functions $\phi_1(\cdot), \phi_2(\cdot), \cdots, \phi_k(\cdot)$, respectively. The output values are given by

$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi_j(\mathbf{x}) \tag{2.10}$$

where $f(\mathbf{x})$ represents the RBF output for input vector $\mathbf{x}$. Therefore, the output layer of the N input vectors $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)^T$ are generated by estimated output vector $\hat{\mathbf{y}}$ that corresponding with

$$\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^T = (f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_N))^T. \tag{2.11}$$

2.4 Shapeless Parameter of Radial Basis Function

As in the previous, the most popular of the RBFs containing shape parameters, which caused the selection problem and have an effect on the accuracy of the model is not good accuracy or not comparable to other methods.

Following the compactly supported radial basis function, the first introduction of this type by Wendland's CS-RBFs (Wendland, 1995). The usual basis function still some problems exist, caused by a large number of centres N That has an effect on methods

มหาวิทยาลัยเทคโนโลยีสุรนารี

of computation and evaluation which was developed to remedy this defect. These problems could be avoided if the radial basis function is compactly supported. If the radial basis function has compact support the interpolation matrices are sparse and for evaluation of the interpolants, only few terms have to be considered. Wendland's CS-RBFs are listed in the Table 2.1.

Table 1.1 The popular Wendland's CS-RBFs.

RBFs	$\phi(r)$
Wendland's	$(1-r)_+$
	$(1-r)_+^3(3r+1)$
	$(1-r)_+^5(8r^2+5r+1)$
	$(1-r)_+^2$
	$(1-r)_+^4(4r+1)$
	$(1-r)_+^6(35r^2+18r+3)$
	$(1-r)_+^8(32r^3+25r^2+8r+1)$

Note that the cut-off function, $(r)_+$ is defined to be r if $0<r<1$ and to be zero elsewhere.

Wu considered interpolation matrices generally to have large matrices. Therefore, to improve importantly if the radial basis function of compact support is implemented, Wu provided a criterion that produces a series of compactly supported radial basis functions (Wu, 1995). Wu employs convolution to construct another kind of CS-RBFs as shown in Table 2.2. Wu's function can be derived by the following formula

$$\phi_{k,s} = D^k(\phi_s), \ d \le 2k+1, \tag{2.1}$$

where d is the dimension of input space, $\mathbb{R}^d$ and the differential operator D is defined as

$$D_{\varphi}(r) = -\phi'(r)/r, r \ge 0. \tag{2.2}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 2.2 The popular Wu's CS-RBFs.

RBFs	$\phi\ r$
Wu	$1-r\ _{+}^{7}\ \ 5r^6+35r^5+101r^4+147r^3+101r^2+35r+5$
	$1-r\ _{+}^{6}\ \ 6r^5+30r^4+72r^3+82r^2+36r+6$
	$1-r\ _{+}^{5}\ \ 5r^4+25r^3+48r^2+40r+8$
	$1-r\ _{+}^{4}\ \ 5r^3+20r^2+29r+16$

Another class of CS-RBFs constructed by Buhmann (Buhmann, 1998) is reminiscent of the popular thin plate splines. Buhmann was interested in studying approximation spaces generated by a novel type of compactly supported radial basis functions as opposed to most previously reviewed globally supported functions. Buhmann has appreciating the advantage of compact support for radial function methods that the linear systems resulting from interpolation from these spaces are easy to solve and on the other hand, that the resulting interpolants can be evaluated very fast. The popular of these CS-RBFs are given in Table 2.3.

Table 2.3 The popular Buhmann's CS-RBFs.

RBFs	$\phi\ r$
Buhmann	$\left(2r^4\log\ r\ -\dfrac{7r^4}{2}+\dfrac{16r^3}{3}-2r^2+\dfrac{1}{6}\right)_+,\mathrm{x}\in\mathbb{R}^3$
	$\left(\dfrac{112r^{\frac{9}{2}}}{45}+\dfrac{16r^{\frac{7}{2}}}{3}-7r^4-\dfrac{14r^2}{15}+\dfrac{1}{9}\right)_+,\mathrm{x}\in\mathbb{R}^2$
	$\left(\dfrac{1}{18}-r^2+\dfrac{4r^3}{9}+\dfrac{r^4}{2}-\dfrac{4r^3\log\ r}{3}\right)_+,\mathrm{x}\in\mathbb{R}^2$

มหาวิทยาลัยเทคโนโลยีสุรนารี

2.5 The Representational Capability (RC)

A matrix consists of a set of column vectors and spans a subspace based on these column vectors. Particularly, if a matrix $\mathbf{G}\in\mathbb{R}^{l\times k}$ and its column partitioning is $\boldsymbol{g}_1\ \ \boldsymbol{g}_2\ \ \dots\ \ \boldsymbol{g}_k$ then this matrix spans a subspace based on its k vectors in an l dimensional space. For our focus to the matrix $\mathbf{G}\in\mathbb{R}^{N\times N}$, we recall that $\mathbf{G}$ has N columns which are $\boldsymbol{g}_1\ \ \boldsymbol{g}_2\ \ \dots\ \ \boldsymbol{g}_N$ and $\mathbf{G}$ spans a subspace of $\mathbb{R}^N$. Also, the dimensionality of this subspace is equal to the rank of $\mathbf{G}$. Therefore, the minimum number of column vectors for representing an interpolation matrix equals its rank. Now we introduce a matric which we recall *Representational Capability (RC).*

Definition 2.1 (Representational Capability of $\mathbf{G}_m$)

Let $\mathbf{G}$ be an $N \times N$ interpolation matrix and the SVD of $\mathbf{G}_m$ be given by Theorem 2.1. If $m \leq N$ and $\mathbf{G}_m$ is

$$\mathbf{G}_m = \sum_{i=1}^{m} s_i \mathbf{u}_i \mathbf{v}_i^T \qquad (2.3)$$

Then RC of $\mathbf{G}_m$ is defined as

$$RC \;\; \mathbf{G}_m \;\; = 1 - \frac{\|\mathbf{G} - \mathbf{G}_m\|_2}{\|\mathbf{G}\|_2} \qquad (2.4)$$

From Corollary 2.2 (Shin, 1998) defined equation (2.14) for $m < N$ as

$$RC \;\; \mathbf{G}_m \;\; = 1 - \frac{s_{m+1}}{s_m} \qquad (2.5)$$

To determination of the number of centres is typically less than the number of input vectors and set centers which provide the best representation of the input space. Note that the interpolation matrix based on N-centres has overfitting problem. Then, the reduction of interpolation based on a number of centres as m with $m \leq N$ and m should not be so small because of leads to under fitting problem.

Since rank of matrix reveals important information about structure of an interpolation matrix. In addition, association of RC-Algorithm with interpolation matrix is its rank which is assumed to be a specific number.

2.6 Additive White Gaussian Noise (AWGN)

The model of noise is the simple and powerful model as Additive White Gaussian Noise (AWGN) from the name have meaning itself:

Additive means the sum of two components which are input term x_k, output term y_k and noise term w_k at the $k-$th position. We can write

$$y_k = x_k + w_k$$

White from a property of the frequency of the spectra in the signal in which white has a frequency of light that is stabilized by the arrangement. Therefore, the white symbol is taken to represent the constant frequency or in other words is zero-mean.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Gaussian from the noise term w_k is random by a Gaussian random variable. The reason Gaussian distribution is suitable because interference results from a combination of different and independent random variables. Therefore, it is considered a good estimation. Since output term y_k have a noise term w_k obtained from *probability density function* especially at Gaussian distribution:

$$F(w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w-\mu)^2}{2\sigma^2}} \tag{2.6}$$

This equation is dependent on mean μ and variance σ^2. From zero-mean Gaussian noise and the standard deviation may be seen as an expected measurement tool for the amplitude of the noise, which it associates with the power of noise. From the AWGN is usually use in signal then the power of noise is importance. Such the power of noise dependent with signal to noise ratio (SNR). Defined as

$$SNR(w) = \frac{input}{noise} \tag{2.7}$$

where $input$ is input term x_k and $noise$ is noise term w_k, this easier to understand SNR is inversely proportional to noise.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER III

COMPUTATIONAL COMPONENTS

3.1 The Main Algorithm

RC algorithm as proposed by Shin and Park (2000) (Shin and Park, 2000) is a method to drag out information of interpolation matrix when RBF is in use. For given input data $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, the algorithm contains the following steps.

Step 1: Select a value to control the receptive field width of the basis function (or the shape parameter) (ε) and effect of noise (δ) which δ is usually taken to be 0.1% to 1.0% and construct the interpolation matrix $\mathbf{G}$. For example, if Gaussian basis function is used, so that

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N1} & g_{N2} & \cdots & g_{NN} \end{bmatrix} \tag{3.1}$$

where $g_{ij} = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\varepsilon^2\right)$ for $i, j = 1, 2, \ldots, N$.

Step 2: Determine the number of centres (m) by applying singular value decomposition of the interpolation matrix $\mathbf{G}$. This yields a diagonal matrix of singular values $s_1 \geq s_2 \geq \cdots \geq s_N \geq 0$. From these, m can be determined from the following:

$$m = \max_{0 \leq i \leq N} \left\{ i \mid s_{i+1} \leq s_1 \times \frac{\delta}{100} \right\}. \tag{3.2}$$

Step 3: Determine the centres of basis function $(\boldsymbol{\mu})$. Partition matrix V from singular value decomposition of the interpolation matrix $\mathbf{G}$ as:

$$V = \begin{array}{c} \left[\begin{array}{cc} v_{11} & v_{12} \\ v_{21} & v_{22} \end{array}\right] \begin{array}{l} m \\ N-m. \end{array} \\ \begin{array}{cc} m & N-m \end{array} \end{array} \tag{3.3}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Next, generate matrix $V' = \begin{bmatrix} v_{11}^T & v_{21}^T \end{bmatrix}$ and apply QR factorization with column pivoting of matrix V'. And then compute $\mathbf{X}^T P$ and choose the first m elements in $\mathbf{X}^T P$ be the centres of basis function which are:

$$\boldsymbol{\mu} = \left\{ \mu_j \right\}_{j=1}^{m}. \tag{3.4}$$

Step 4: Compute the weight parameters from the basis function (w). Consider,

$$\phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nm} \end{bmatrix} \tag{3.5}$$

For $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, m$. Compute the m weights with

$$w = \phi^{+} y \tag{3.6}$$

where ϕ^{+} denoted the pseudo inverse of ϕ.

3.2 Node Distribution Manner

For data-partitioning process, suggestion provided by Damiana Lazzaro and Laura B. Montefusco (2002) (Lazzaro and Montefusco, 2002) is followed here with the information described as follows.

3.2.1 The Training Datasets

Two datasets with the size of 50×50 nodes are generated within a [0,1]× [0,1] domain; uniformly and randomly, shown in Figure 3.1 and Figure 3.2, respectively. These sets of data are to be used as ‘Training Dataset’ for all numerical experiments in this study.

มหาวิทยาลัยเทคโนโลยีสุรนาร

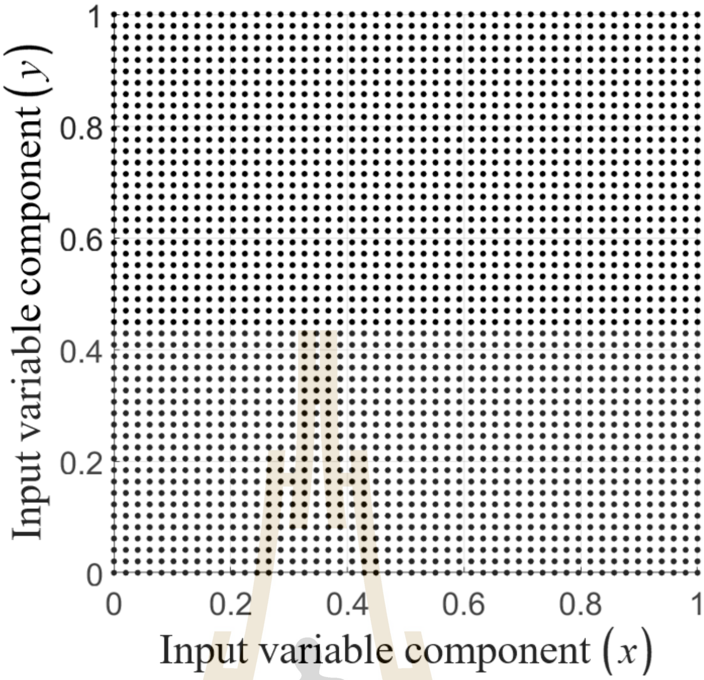


Figure 3.1 $(x, y)-$Node uniformly distribution manner for training datasets using 50×50 nodes.

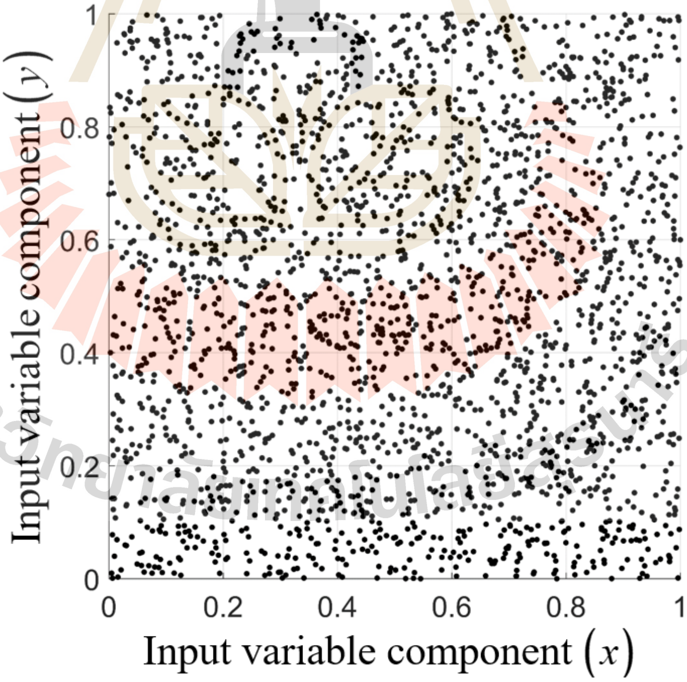


Figure 3.2 $(x, y)-$Node non-uniformly distribution manner for training datasets using 50×50 nodes.

มหาวิทยาลัยเทคโนโลยีสุรนารี

3.2.2 The Testing Datasets

To monitor and record the effectiveness of each RBF type, three large datasets are generated within the same domain as the training ones in both manners; uniformly (Ufm.) and randomly (Rdm.), and they contain 10000, 20164 and 30276 nodes (Lazzaro and Montefusco, 2002).

3.3 Performance Criteria

As previously mentioned, all fifteen shapeless RBFs are comparatively investigated. Therefore, proper and all-around effectiveness criteria are required and they are listed below.

- *Accuracy:* This process is carried out using the following three error norms;

Table 3.1 Error Norms adopted in this work.

Error Norm	Symbol Defined	Mathematical Formula
Maximum	L_∞	$\max\limits_{1\le i\le N}\left\|u^{ext.}(\mathbf{x}_i)-u^{appx.}(\mathbf{x}_i)\right\|$
Root-Mean-Square	L_{RMS}	$\left(\frac{1}{N}\sum_{j=1}^{N}\left(u^{ext.}(\mathbf{x}_i)-u^{appx.}(\mathbf{x}_i)\right)^2\right)^{1/2}$
Absolute	L_{Abs}	$\left\|u^{ext.}(\mathbf{x}_i)-u^{appx.}(\mathbf{x}_i)\right\|$

- *Condition number:* The system can be solvable if the collocation matrix, $\boldsymbol{\varphi}$, has an inverse and this can be indicated by the means of its condition number ($Cond_\delta(\cdot)$) expressed as;

$$Cond_\delta(\boldsymbol{\varphi})=\|\boldsymbol{\varphi}\|_\delta\left\|\boldsymbol{\varphi}^{-1}\right\|_\delta . \tag{3.7}$$

The trending behaviour of this number is also recorded throughout this experiment, providing information on the solvability, (Lazzaro and Montefusco, 2002) , of the collocation matrix for future uses.

- *CPU-time:* With less amount of time required for the computational process, a method would be more desirable. The ‘tic-toc’ command in MATLAB is employed for this task.

- *Storage requirement:* Each computation step involved in the algorithm should ideally take as least amount of storage space as possible. For the sake of fairness, all experiments are carried out on the same computer; Intel(R) Core(TM) i7-1065G7 CPU

มหาวิทยาลัยเทคโนโลยีสุรนารี

@ 1.30GHz 1.50 GHz with RAM 16.00 GB and 64-bit Operating System.

- *User's Interference:* A good algorithm should be able to process entirely by itself, no human interruption or judgments should be involved. This is all for future practical uses.
- *Sensitivity to parameters:* A small change in parameters embedded in the process should have as little effect as possible on the overall performance. Methods containing no parameters would be best in practice.
- *Ease of implementation:* Once the desirable method has been proven with small test models, it is then anticipated to successfully be implemented to larger problems with no difficulties (in terms of both mathematical structures and programming/coding).

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER IV

PRELIMINARY EXPERIMENTS

4.1 Experiment 1: A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs

4.1.1 The Project Objective

The main focus is on one type of RBF, ' Compactly-Supported (CS-RBF)’, which contains no parameter and has not been explored numerically as much in the past, particularly under the context of data interpolation/approximation and solving partial differential equations (PDEs). To compare the potential advantages of CS-RBF, the two most popular choices of RBF widely used; Multiquadric (MQ), and Gaussian (GA) were studied parallelly.

4.1.2 Globally-Supported RBFs

For the method of collocation meshfree, there are several forms of well-known radial basis function and some are listed below, (Kaennakham and Chuathong, 2017);

- Gaussian (GS), defined as;

$$\phi(r)=e^{-(\varepsilon r)^2}$$

- Inverse Multiquadric (IMQ), defined as;

$$\phi(r)=1/\sqrt{1+(\varepsilon r)^2}$$

- Multiquadric (MQ), defined as;

$$\phi(r)=\sqrt{1+(\varepsilon r)^2}$$

- Inverse quadratic (IQ), defined as;

$$\phi(r)=1/\left(1+(\varepsilon r)^2\right)$$

Here, ε is called shape parameter, determined by the user and this exactly remains the open topic for researchers to investigate the possible optimal choice. A lot of attempts have been made to alleviate this problem but by far no universal

มหาวิทยาลัยเทคโนโลยีสุรนารี

choice has been found (Hardy, 1977; Franke, 1979; Carlson and Foley, 1991). The attempt to get rid of this burden by turning attention away from these groups of RBFs to those containing no parameter is now being made.

4.1.3 Compactly-Supported RBFs

For this purpose, this work focusses on a compactly-supported form of RBF proposed by Buhmann (Buhmann, 1998) and the effectiveness one can obtained when applied this type of RBF with the collocation-based method.

- Noted as CS1 and defined as;

$$\phi\ r = 119r^{9/2}/45+16r^{7/2}/3-7r^4-14r^2/15+1/9\ _{+}$$

- Noted as CS2 and defined as;

$$\phi\ r = -4r^3\log(r)/3+r^4/2+4r^3/9-r^2+1/18\ _{+}$$

- Noted as CS3 and defined as;

$$\phi\ r = 2r^4\log(r)-7r^4/2+16r^3/3-2r^2+1/6\ _{+}$$

when $+$ indicates the cut-off function which is defined to be r, when $0 \le r \le 1$ and zero elsewhere.

4.1.4 Test Case 1: Interpolation Problem

The interpolation problem starts with a set of discrete data $\mathbf{X}=\{\mathbf{x}_i\}_{i=1}^{N}, \mathbf{x}_i \in \mathbb{R}^d$ where for each $\mathbf{x}_i$ there is its corresponding real value $y_i \in \mathbb{R}$, then the task is to construct a continuous function $\phi(\mathbf{x}):\mathbb{R}^d \to \mathbb{R}$ such that;

$$\phi(\mathbf{x}_i) = y_i\,. \tag{4.1}$$

For all $i = 1,2,\ldots,N$. The first validation of the scheme proposed in this work is done using a benchmark function defined on a square-domain, $[0,4\pi]\times[0,4\pi]$, as follows:

$$f(x,y) = \sin(x)\cos(y). \tag{4.2}$$

As regarded as one of the most influential factors, the shape parameter is firstly investigated. For this purpose simulations using the Multiquadric (MQ) and Gaussian (GA) were under the investigation. Figure 4.1 and Figure 4.2 clearly show the strong effects the shape parameter has for both RBFs. When using different numbers of nodes, it also can be seen that the errors norms can be affected as well.

มหาวิทยาลัยเทคโนโลยีสุรนารี

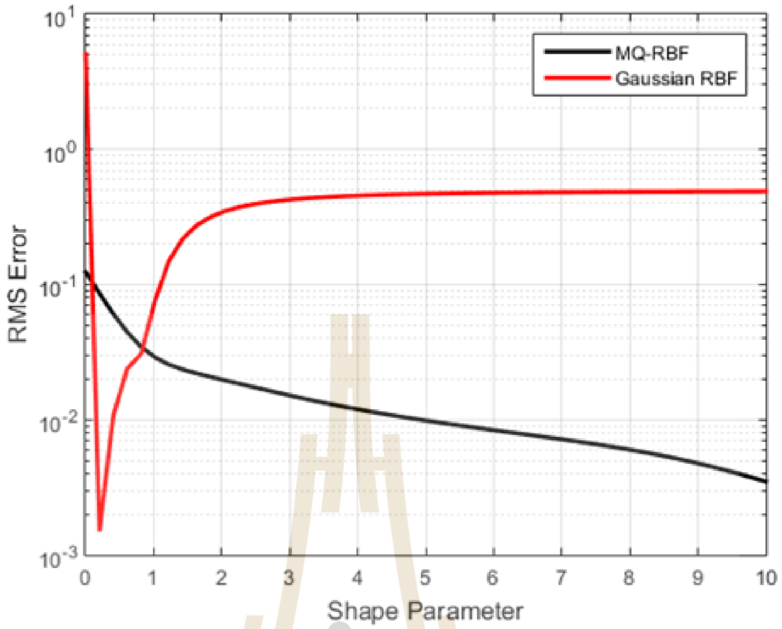


Figure 4.1 L_{RMS} measured at different values of shape parameters with 10 × 10 interpolation nodes/centres.

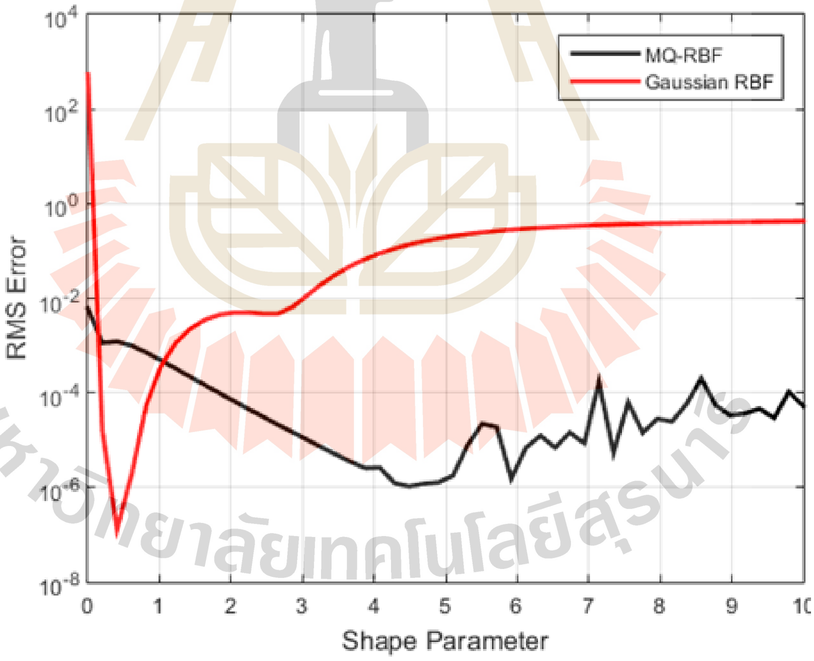


Figure 4.2 L_{RMS} measured at different values of shape parameters with 30 × 30 interpolation nodes/centres.

มหาวิทยาลัยเทคโนโลยีสุรนารี

At the node density of 15×15, Table 4.1 provides the absolute errors measured at different locations all over the domain. The information contained in this table shows significant accuracy obtained from all types of RBFs. Nevertheless, for MQ and GA, a caution has been previously taken on fining the optimal shape values and they were found to be in (4,6.5) for MQ, (0.1-0.8) and for GA. It must be noted that no extra treatment was needed for the cases of using compactly-supported RBFs. Solutions produced by all the cases were plotted against one another as shown in Figure 4.3 and Figure 4.4.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.1 L_{Abs} measured at different locations over the computational domain with 15×15 interpolation nodes/centres.

x	y	MQ ($\varepsilon=5.00$)	GA ($\varepsilon=0.50$)	CS1	CS2	CS3
0.000	0.000	8.59E-10	1.57E-12	1.46E-11	3.55E-11	1.46E-11
0.000	1.396	1.87E-06	2.73E-12	4.17E-04	1.09E-02	5.42E-04
0.000	12.566	1.32E-09	3.64E-13	5.46E-11	7.84E-12	6.37E-12
1.396	0.000	7.06E-06	1.60E-05	6.66E-04	7.37E-05	3.86E-03
2.793	0.000	2.56E-06	5.86E-06	3.49E-04	1.41E-03	1.07E-02
2.793	1.396	7.73E-06	1.13E-05	8.23E-04	5.66E-03	9.36E-04
2.793	12.566	2.55E-06	5.86E-06	3.49E-04	1.41E-03	1.07E-02
4.189	0.000	1.12E-06	2.43E-06	5.30E-04	1.21E-02	1.55E-02
4.189	12.566	1.12E-06	2.43E-06	5.30E-04	1.21E-02	1.55E-02
5.585	12.566	3.32E-07	6.88E-07	3.51E-04	6.51E-03	1.64E-02
6.981	0.000	3.32E-07	6.88E-07	3.51E-04	6.51E-03	1.64E-02
6.981	12.566	3.32E-07	6.88E-07	3.51E-04	6.51E-03	1.64E-02
8.378	0.000	1.12E-06	2.43E-06	5.30E-04	1.21E-02	1.55E-02
8.378	12.566	1.12E-06	2.43E-06	5.30E-04	1.21E-02	1.55E-02
9.774	0.000	2.55E-06	5.86E-06	3.49E-04	1.41E-03	1.07E-02
9.774	11.170	7.73E-06	1.13E-05	8.23E-04	5.66E-03	9.36E-04
9.774	12.566	2.56E-06	5.86E-06	3.49E-04	1.41E-03	1.07E-02
11.170	0.000	7.07E-06	1.60E-05	6.66E-04	7.37E-05	3.86E-03
11.170	12.566	7.06E-06	1.60E-05	6.66E-04	7.37E-05	3.86E-03
12.566	0.000	1.09E-09	3.76E-13	1.20E-10	6.24E-11	2.64E-11
12.566	12.566	2.26E-09	9.46E-13	5.91E-13	5.34E-11	4.06E-11

มหาวิทยาลัยเทคโนโลยีสุรนารี

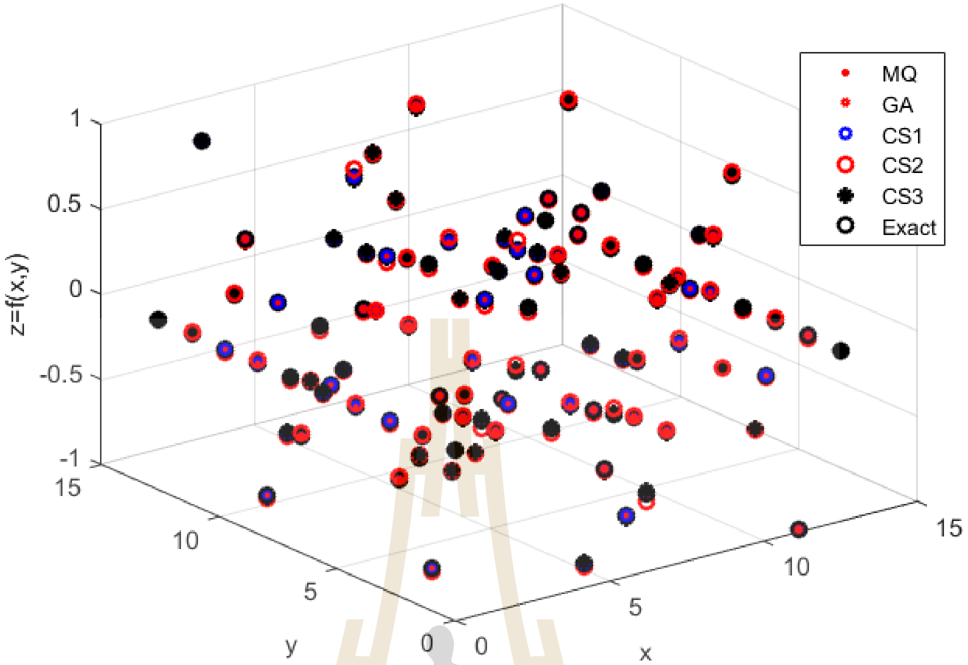


Figure 4.3 Solution approximation at selected locations obtained from each RBF type, compared to against the exact solution.

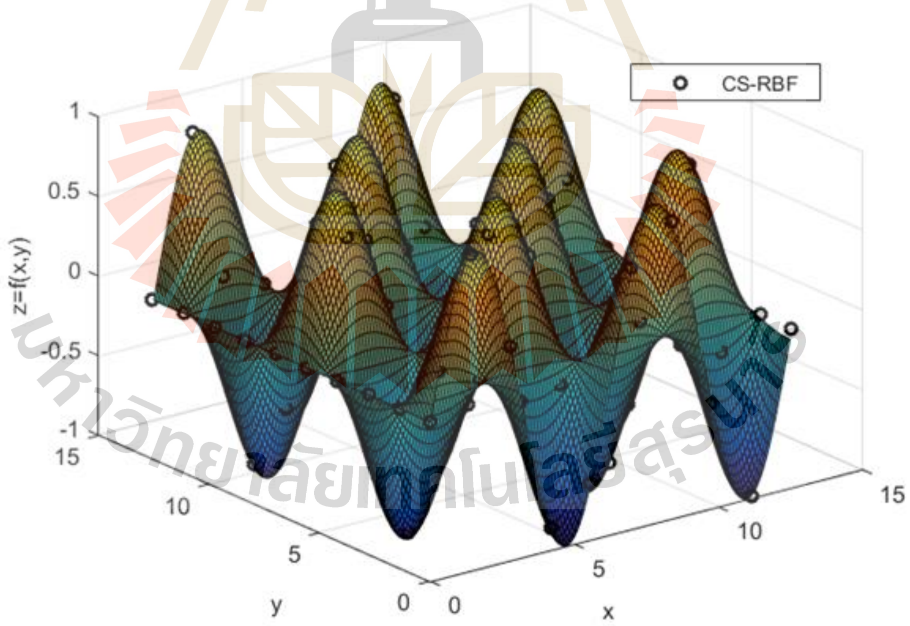


Figure 4.4 Solution profile comparison between that produced by CS1-RBF and that of the exact one.

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.1.5 Test Case 2: Poisson Problem with Non-rectangular domain

In this second test case, the Poisson equation shown below is numerically solved by the collocation meshfree method.

$$\nabla^2 u = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u = -x^2.$$

This is defined on the domain with an elliptical boundary expressed as $\left(x^2/4\right) + y^2 = 1$. Where the boundary condition is taken directly from the exact solution which is expressed as follows;

$$u(x,y) = -\frac{1}{246}\left(50x^2 - 8y^2 + 33.6\right)\left(\frac{x^4}{4} + y^2 - 1\right). \tag{4.3}$$

Based on the promising results obtained from the previous example, only CS1 is now presented here and it is marked as 'CS' from hereon.

For this example, L_∞ and L_{RMS} were utilized to measure the accuracy at different node density of 64, 144, 225, and 400. Table 4.2 clearly shows that the accuracy obtained from using MQ and GA is greatly influenced by the number of nodes involved in the system. This figure is more obvious when using 225 nodes with GA-RBF where a strong fluctuation of L_{RMS} occurs. This is not, however, the case when using CS where the solutions were found to remain almost intact with the change of node density. Figure 4.5, Figure 4.6, and Figure 4.7 depicts the node distributions and the corresponding solution profile. It should also be remarked here that the optimal values of shape parameters shown in the Table 4.2 were obtained from a series of simulations taking place beforehand.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.2 Errors produced by each RBF when using different levels of nodes density.

Number of Computational Nodes	Error Norms	MQ $(\varepsilon=0.1)$	GA $(\varepsilon=0.1)$	CS
64	L_∞	8.6331e-04	9.4614e-04	0.0157
	L_{RMS}	3.8422e-04	3.5081e-04	0.0063
144	L_∞	5.7818e-04	0.0036	0.0077
	L_{RMS}	2.0858e-04	0.0014	0.0039
225	L_∞	0.0057	0.0028	0.0048
	L_{RMS}	0.0024	9.4496e-04	0.0027
400	L_∞	0.0034	0.0039	0.0032
	L_{RMS}	0.0012	0.0014	0.0015

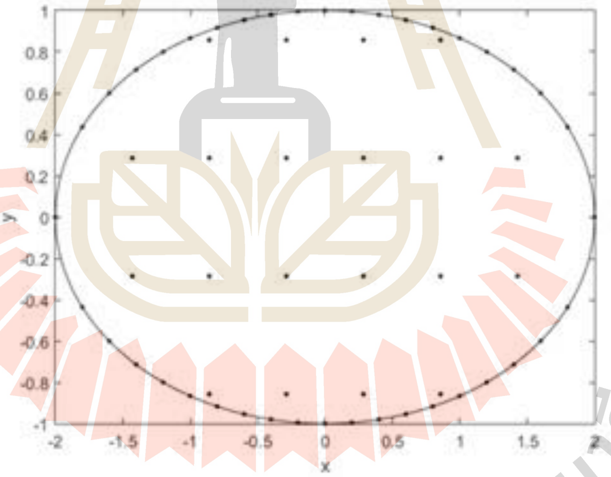


Figure 4.5 Density distribution at 64 nodes.

มหาวิทยาลัยเทคโนโลยีสุรนารี

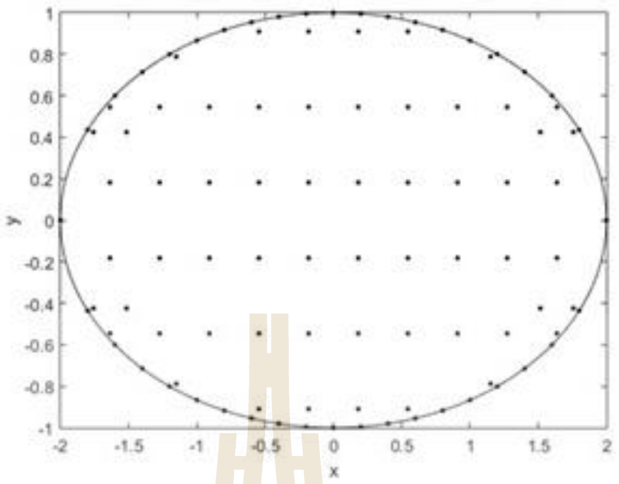


Figure 4.6 Density distribution at 144 nodes.

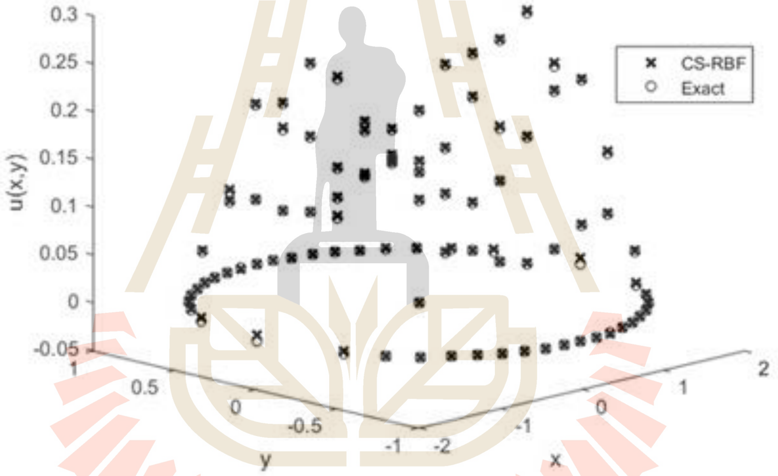


Figure 4.7 Node solution comparison between that of CS-RBF and the exact one.

4.1.6 Test Case 3: Nonlinear Problem

The nonlinear PDE as given in GE. Fasshauer[5] on a square domain $0<x<1,\ 0<y<1$ is taken a look at. The governing equation is as follows;

$$-\varepsilon^2\nabla^2 u-u+u^3=f \tag{4.4}$$

with the boundary condition $u=0$ on $\partial\Omega$ and the right-hand side of the equation is chosen from the analytical solution of form;

$$u(x,y)=\psi(x)\psi(y) \tag{4.5}$$

with $\psi(t)=1+e^{-1/\varepsilon}-e^{-t/\varepsilon}-e^{(t-1)/\varepsilon}$,and (x,y) denotes the Cartesian coordinate of $\mathbf{x}\in\mathbb{R}^2$.

The computational domain for this example is shown in Figure 4.8 where

มหาวิทยาลัยเทคโนโลยีสุรนารี

some extra nodes are required for the computing algorithm for nonlinear case. Table 4.3 provide strong evidence confirming that the solution quality produced by CS can be as good as those obtained from the two popular choices of RBFs. Solution profiles are plotted against the exact ones and shown in Figure 4.9, Figure 4.10 and Figure 4.11.

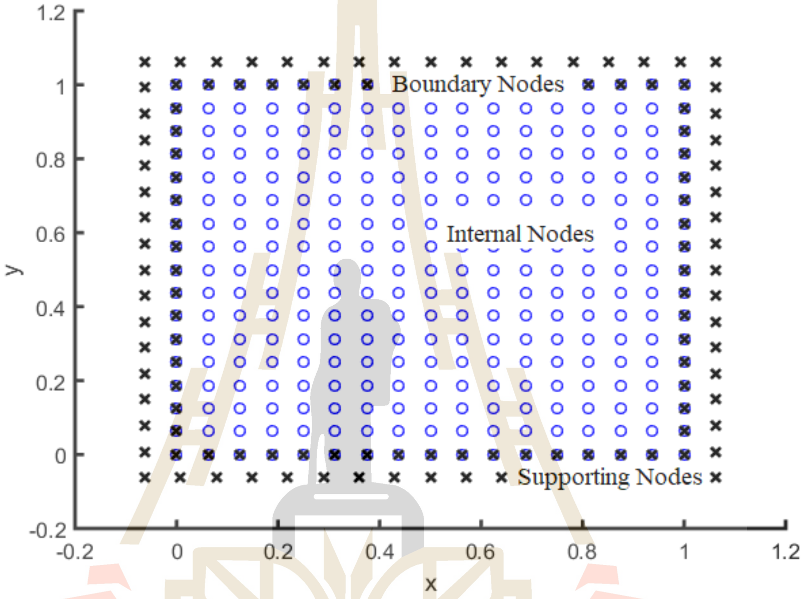


Figure 4.8 Node being uniformly-distributed over the computational domain, together with supported nodes needed for nonlinear computing process.

Table 4.3 Numerical solutions comparison with the exacts.

Points	MQ	GA	CS	Exact
(0.2,0.2)	0.750708	0.751436	0.753032	0.747144
(0.2,0.4)	0.848307	0.849172	0.849450	0.846440
(0.2,0.6)	0.848307	0.849172	0.849450	0.846440
(0.4,0.2)	0.848307	0.849172	0.849450	0.846440
(0.4,0.4)	0.959267	0.960383	0.959399	0.958933
(0.4,0.6)	0.959267	0.960383	0.959399	0.958933
(0.6,0.2)	0.848307	0.849172	0.849450	0.846440
(0.6,0.4)	0.959267	0.960383	0.959399	0.958933
(0.6,0.6)	0.959267	0.960383	0.959399	0.958933
(0.6,0.8)	0.848307	0.849172	0.849450	0.846440
(0.8,0.2)	0.750708	0.751436	0.753032	0.747144
(0.8,0.4)	0.848307	0.849172	0.849450	0.846440
(0.8,0.6)	0.750708	0.751436	0.753032	0.846440

มหาวิทยาลัยเทคโนโลยีสุรนารี

(0.8,0.8)	0.750708	0.751436	0.753032	0.747144

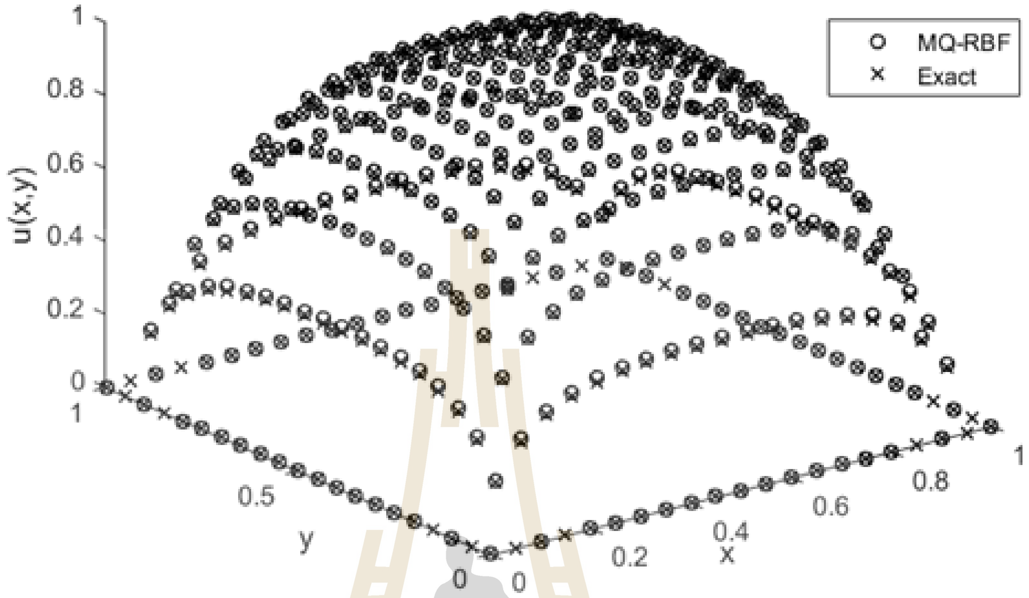


Figure 4.9 Solution comparisons by using MQ-RBF.

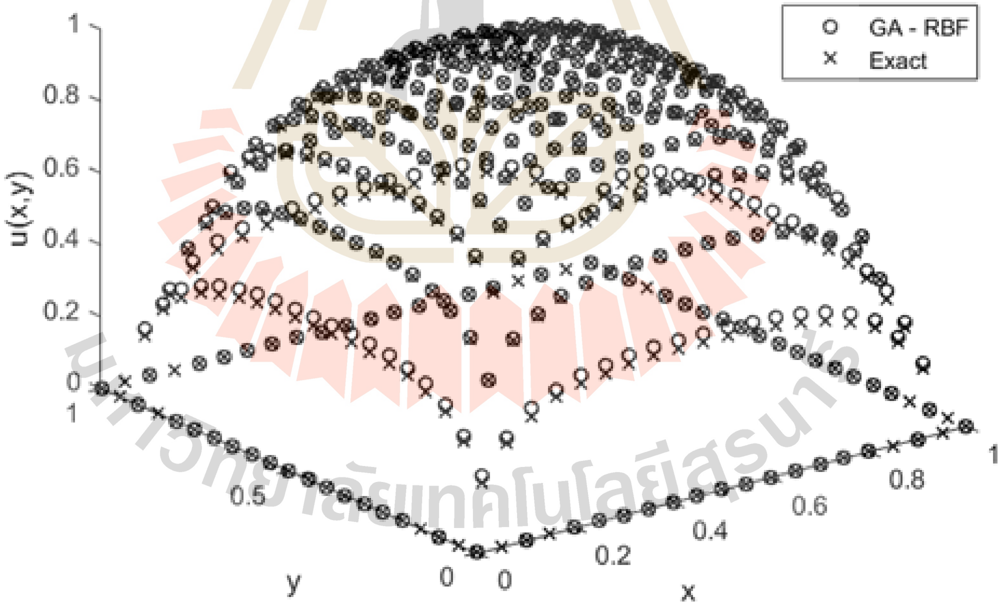


Figure 4.10 Solution comparisons by using GA-RBF.

มหาวิทยาลัยเทคโนโลยีสุรนารี

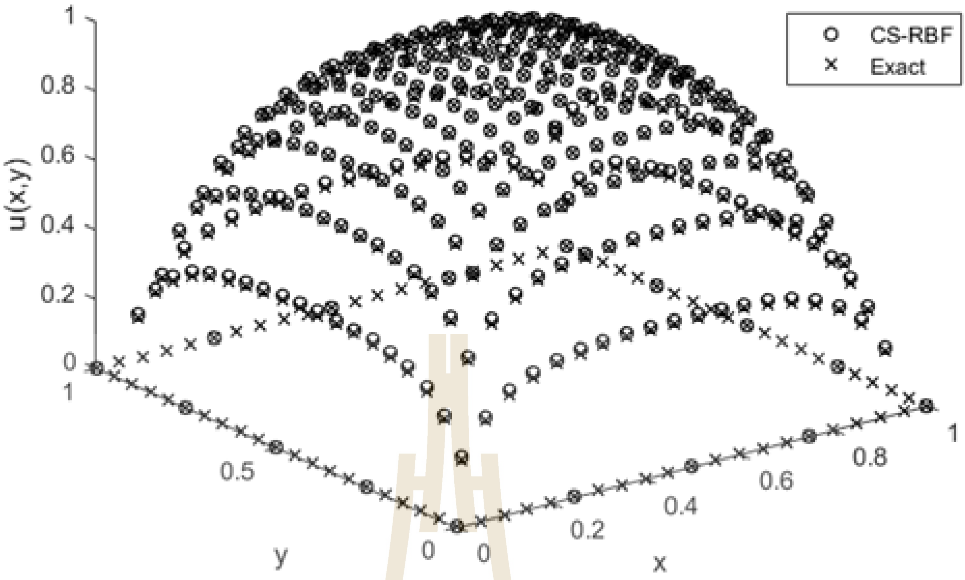


Figure 4.11 Solution comparisons by using CS-RBF.

4.1.7 Main Conclusion

The main objective is to numerically study the effectiveness of one type of radial basis functions that has not been explored as much and it is called 'compacted-supported, CS-RBF'. This is done under the context of a numerical method that requires no mesh or grid so it is called 'meshfree/meshless' with collocation on computational nodes. The CS-RBF is applied to several benchmark test cases and its potential uses have been monitored. The main findings obtained from all the experiments are as follows;

1. When compared to the use of two popular RBFs; Multiquadric (MQ) and Gaussian (GA), the burden of finding an optimal shape parameter normally encountered in MQ and GA is completely omitted while the solutions produced are still in a good agreement.
2. The numerical solutions are found to be significantly affected by the distance between centres in the domain. This is not the case for all the shape-parameter-based RBFs such as MQ and GA.
3. The mathematical structure of CS-RBF is much less complex and much simpler when it comes to finding its first and second order of derivatives for solving PDEs.

This all indicates its promising future for further applications in more complex problems and it remains as our future investigation.

4.2 Experiment 2: Performances of Non-Parameterised Radial Basis Functions in Pattern Recognition Applications

4.2.1 The Project Objective

This study focuses on RBFs which contain no parameters where three data patterns (Linear interpolation, Parabola function and Sine interpolation) are used for performance validation and comparison with the popular radial basis function which are Gaussian RBF that contains the large problem is finding the optimal shape parameter. Additionally, by studying the RBF by using RC-Algorithm for interpolation for Pattern Recognition problem and studying the behavior of the numbers of centres with a good choice of the number of centres, it is clearly possible to obtain satisfactory results with no burden on choosing the suitable or optimal shape.

4.2.2 The Non-Parameterised Radial Basis Functions

With the non-straight forward way to pinpoint the optimal choice for Gaussian RBF uses, the main attention has now turned to alternative forms of RBF. In this work, it focuses on non-parameter form of radial basis functions and with this purpose, those proposed by Buhmann are to be explored and they are;

- Noted as '**CS-RBF1**' and defined as: $\phi(r)=\frac{1}{3}+r^2-\frac{4}{3}r^3+2r^2\log(r)$.
- Noted as '**CS1**' and defined as: $\phi(r)=\frac{112}{45}r^{\frac{9}{2}}+\frac{16}{3}r^{\frac{7}{2}}-7r^4-\frac{14}{15}r^2+\frac{1}{9}$.
- Noted as '**CS2**' and defined as: $\phi(r)=\frac{1}{18}-r^2+\frac{4}{9}r^3+\frac{1}{2}r^4-\frac{4}{3}r^3\log(r)$.

What appears to be interesting about these forms is that they do not depend on any user's input information making it more convenient when in use.

4.2.3 Test Case 1: Linear interpolation

In this section, a linear case is studied and the simple line expressed as followed is considered:

$$y=2x+1 \tag{4.6}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

The total number of 100 data points from above function with x in $[0, 2\pi]$ are generated. The data points are generated by $x_i = 2\pi\left(\frac{i-1}{100}\right), i = 1, 2, \ldots, 100$ with node distribution are shown in Table 4.4 and its plot are in Figure 4.12.

Table 4.4 The locations over the computational domain with 100 points for linear trend case.

i	x_i	y_i	i	x_i	y_i
1	0	0.655842	51	3.141593	5.600409
2	0.062832	2.114886	52	3.204425	7.73681
3	0.125664	1.913245	53	3.267256	5.122997
4	0.188496	3.177257	54	3.330088	8.606219
5	0.251327	0.689014	55	3.39292	8.462995
6	0.314159	1.668217	56	3.455752	9.174039
7	0.376991	-1.3465	57	3.518584	7.321504
8	0.439823	1.61845	58	3.581416	6.916049
9	0.502655	1.532319	59	3.644247	9.373074
10	0.565487	2.71808	60	3.707079	11.14124
11	0.628319	2.020064	61	3.769911	9.624707
12	0.69115	1.312449	62	3.832743	8.687917
13	0.753982	2.001953	63	3.895575	9.33649
14	0.816814	3.82832	64	3.958407	7.912225
15	0.879646	1.935325	65	4.021239	9.111349
16	0.942478	3.008351	66	4.08407	7.770642
17	1.00531	3.669298	67	4.146902	8.799503
18	1.068142	4.632178	68	4.209734	12.25011
19	1.130973	5.154288	69	4.272566	11.29631
20	1.193805	3.174726	70	4.335398	10.34536
21	1.256637	6.069671	71	4.39823	8.374006
22	1.319469	3.162803	72	4.461062	13.23273
23	1.382301	2.696466	73	4.523893	10.0354
24	1.445133	3.421394	74	4.586725	10.7539

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.4 (continued).

i	x_i	y_i	i	x_i	y_i
25	1.507964	0.350422	75	4.649557	11.40374
26	1.570796	2.981907	76	4.712389	9.190378
27	1.633628	2.666889	77	4.775221	12.15359
28	1.69646	1.362263	78	4.838053	9.736684
29	1.759292	7.036787	79	4.900885	12.20263
30	1.822124	4.011233	80	4.963716	12.78446
31	1.884956	4.989338	81	5.026548	10.79367
32	1.947787	5.14384	82	5.08938	12.00101
33	2.010619	7.63181	83	5.152212	7.833796
34	2.073451	4.510752	84	5.215044	9.437101
35	2.136283	4.955316	85	5.277876	8.841867
36	2.199115	6.488184	86	5.340708	8.713244
37	2.261947	5.255902	87	5.403539	13.24431
38	2.324779	7.085093	88	5.466371	13.87933
39	2.38761	5.16598	89	5.529203	11.30799
40	2.450442	3.295977	90	5.592035	9.982219
41	2.513274	5.019097	91	5.654867	11.53432
42	2.576106	8.208935	92	5.717699	12.28004
43	2.638938	7.915806	93	5.78053	12.51945
44	2.70177	3.932963	94	5.843362	15.95529
45	2.764602	6.406319	95	5.906194	12.41553
46	2.827433	3.74281	96	5.969026	12.26261
47	2.890265	5.261321	97	6.031858	14.35
48	2.953097	8.177601	98	6.09469	11.41828
49	3.015929	6.559347	99	6.157522	11.73805
50	3.078761	7.892402	100	6.220353	12.72404

มหาวิทยาลัยเทคโนโลยีสุรนารี

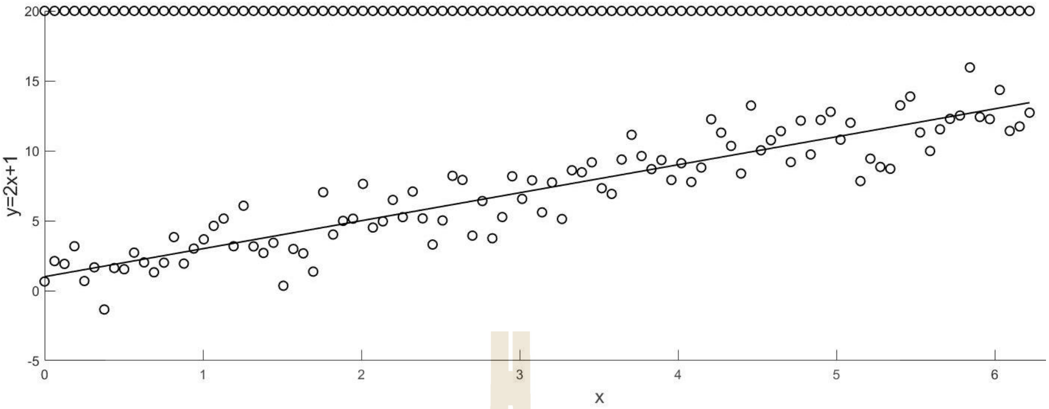


Figure 4.12 Data with 100 points for linear trend case.

With using RC Algorithm, the parameters σ and m are optioned for both training and validation cases are shown in Table 4.5.

Table 4.5 Training and validation errors for candidate models for linear trend case.

RBF	σ	m	MSE	
			Training	Validation
CS-RBF1	-	29	1.6659	3.1688
CS1	-	13	1.9633	2.849
CS2	-	22	1.853	3.0211
Gaussian I	0.2	9	2.2658	2.4408
Gaussian II	0.3	7	2.2941	2.4161
Gaussian III	0.4	5	2.3514	2.3643
Gaussian IV	0.5	5	2.3458	2.3603
Gaussian V	0.6	4	2.3444	2.3585
Gaussian VI	0.7	4	2.3438	2.3574

Thus, based on RC Algorithm, Gaussian III with $\sigma = 0.4$ and $m = 5$ for this data set can be a good model. When using the same numbers of centres, Table 4.6 and Table 4.7 provide the main results of this case. Figure 4.13 and Figure 4.14 illustrate the predicted training trend and predicted validation trend, respectively.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.6 Results comparison when using different numbers of centres and RBFs for linear trend case.

RBF	$m=9, \sigma=0.2$		$m=7, \sigma=0.3$		$m=5, \sigma=0.4$		$m=4, \sigma=0.7$	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
CS-RBF1	2.1988	2.5176	2.1796	2.4946	2.2539	2.6201	2.4811	2.5307
CS1	2.2469	2.4538	2.2802	2.4502	2.6874	2.6801	2.3129	2.5212
CS2	2.2698	2.4799	2.3149	2.4290	2.6218	2.6326	2.6478	2.8352
Gaussian	2.2658	2.4408	2.2941	2.4161	2.3514	2.3643	2.3438	2.3574

Table 4.7 Listing of basis function centres for linear trend case.

m	RBF	Weights								
		μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9
9	**CS-RBF1**	0.6768	0.9697	0.0303	0.1818	0.9293	0.8586	0.6465	0.2525	0.8990
	CS1	0.0707	0.6667	0.5859	0.9293	0.8485	0.2424	1.0000	0	0.4141
	CS2	0.7172	0.1919	0.2323	0.0404	0.6667	1.0000	0.9091	0.7677	0.1414
	Gaussian	0	1.0000	0.0808	0.9192	0.2121	0.7879	0.3535	0.6465	0.5051
7	**CS-RBF1**	0.2828	0.9697	0.5051	0.7475	0.9293	0.1818	0.8990		
	CS1	0.2424	0.4141	0.9293	0	0.3333	1.0000	0.5859		
	CS2	0.7172	0.5253	0.1919	0.6162	1.0000	0.0404	0.2828		
	Gaussian	0	1.0000	0.1010	0.8889	0.7071	0.2929	0.4949		
5	**CS-RBF1**	0.8586	0.4646	0.8182	0.9293	0.3939				
	CS1	0.5051	0.4141	0.5859	0.8485	0.2424				
	CS2	0.2828	0.9091	0.1919	0.7677	0.3333				
	Gaussian	1.0000	0	0.8081	0.2020	0.5051				
4	**CS-RBF1**	0.7172	0.9293	0.3535	0.1818					
	CS1	0.7576	0.8485	1.0000	0.0707					
	CS2	0.4242	0.5758	1.0000	0.7677					
	Gaussian	1.0000	0	0.2727	0.7071					

มหาวิทยาลัยเทคโนโลยีสุรนารี

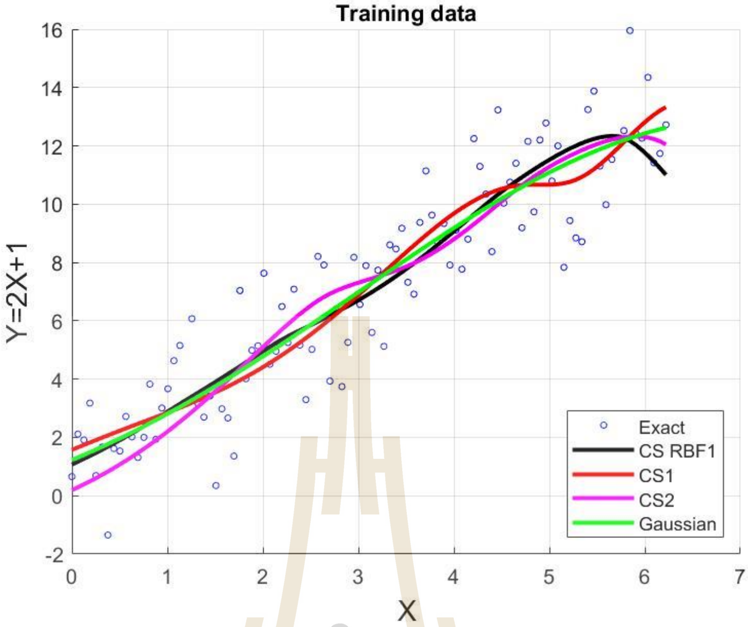


Figure 4.13 Predicted training trend produced by using 4 centres for linear trend case.

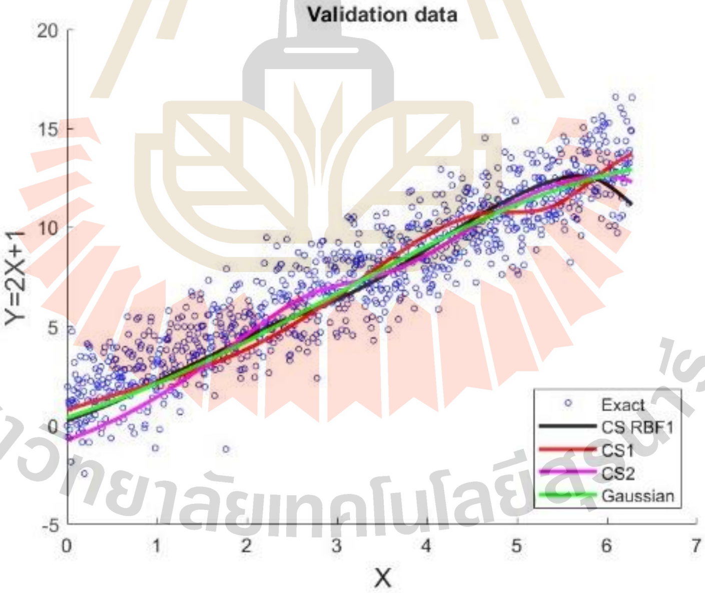


Figure 4.14 Predicted validation trend produced by using 4 centres for linear trend case.

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.2.4 Test Case 2: Parabola function

The second case to investigate is a parabola described as the following equation;

$$y = \frac{x^2}{6} - x + 4. \tag{4.7}$$

The data points are generated similarly to the previous example show in Table 4.8 and Figure 4.15 depicts the plot with nodes.

Table 4.8 The locations over the computational domain with 100 points for parabola trend case.

i	x_i	y_i	i	x_i	y_i
1	0	4.24814	51	3.141593	3.604046
2	0.062832	3.704129	52	3.204425	2.579202
3	0.125664	4.169598	53	3.267256	1.807343
4	0.188496	3.80927	54	3.330088	2.351547
5	0.251327	3.737286	55	3.39292	3.171114
6	0.314159	4.194421	56	3.455752	3.909455
7	0.376991	3.873224	57	3.518584	2.434338
8	0.439823	3.131685	58	3.581416	2.215887
9	0.502655	3.585788	59	3.644247	2.640374
10	0.565487	2.777425	60	3.707079	2.329124
11	0.628319	3.018628	61	3.769911	2.689632
12	0.69115	3.94122	62	3.832743	2.383895
13	0.753982	2.924287	63	3.895575	1.91565
14	0.816814	4.103699	64	3.958407	2.733152
15	0.879646	3.121798	65	4.021239	2.613712
16	0.942478	3.667874	66	4.08407	1.75167
17	1.00531	3.087243	67	4.146902	2.591541
18	1.068142	3.556016	68	4.209734	3.31661
19	1.130973	3.212318	69	4.272566	2.065257
20	1.193805	2.839454	70	4.335398	2.945395
21	1.256637	2.198888	71	4.39823	3.153849

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.8 (continued).

i	x_i	y_i	i	x_i	y_i
22	1.319469	2.854652	72	4.461062	3.36909
23	1.382301	1.320843	73	4.523893	3.693778
24	1.445133	2.485727	74	4.586725	3.221386
25	1.507964	2.629543	75	4.649557	2.778476
26	1.570796	2.603496	76	4.712389	3.352436
27	1.633628	3.497001	77	4.775221	3.480584
28	1.69646	3.166593	78	4.838053	3.345557
29	1.759292	2.94798	79	4.900885	2.639394
30	1.822124	2.863549	80	4.963716	3.038108
31	1.884956	1.377919	81	5.026548	3.265107
32	1.947787	3.41562	82	5.08938	3.36192
33	2.010619	2.518353	83	5.152212	3.401107
34	2.073451	2.415294	84	5.215044	2.460658
35	2.136283	2.273644	85	5.277876	3.513121
36	2.199115	3.257021	86	5.340708	3.020274
37	2.261947	2.406498	87	5.403539	3.86994
38	2.324779	3.192788	88	5.466371	3.490127
39	2.38761	2.733065	89	5.529203	2.096448
40	2.450442	2.78557	90	5.592035	3.480605
41	2.513274	2.50616	91	5.654867	3.053712
42	2.576106	2.203222	92	5.717699	3.385007
43	2.638938	2.669624	93	5.78053	4.334289
44	2.70177	2.295242	94	5.843362	4.214735
45	2.764602	2.688145	95	5.906194	4.293023
46	2.827433	2.931139	96	5.969026	4.202053
47	2.890265	2.774863	97	6.031858	4.068915
48	2.953097	2.429877	98	6.09469	4.440242
49	3.015929	1.973173	99	6.157522	3.394129
50	3.078761	2.430819	100	6.220353	3.690945

มหาวิทยาลัยเทคโนโลยีสุรนารี

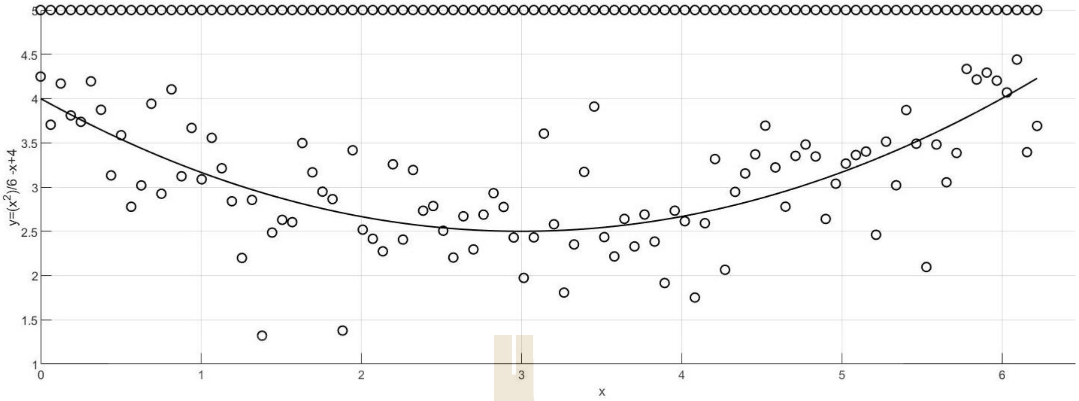


Figure 4.15 Data with 100 points for parabolas trend case.

With using RC Algorithm, the parameters σ and m are optioned for both training and validation cases are shown in Table 4.9.

Table 4.9 Training and validation errors for candidate models for parabolas trend case.

RBF	σ	m	MSE	
			Training	Validation
CS-RBF1	-	29	0.17637	0.52294
CS1	-	13	0.22069	0.44781
CS2	-	22	0.18575	0.51042
Gaussian I	0.2	9	0.24256	0.3985
Gaussian II	0.3	7	0.24649	0.3960
Gaussian III	0.4	5	0.24742	0.39345
Gaussian IV	0.5	5	0.24736	0.39365
Gaussian V	0.6	4	0.25371	0.39465
Gaussian VI	0.7	4	0.25203	0.39386

Thus, based on RC Algorithm, Gaussian V with $\sigma = 0.6$ and $m = 4$ for this data set can be a good model. When using the same numbers of centres, Table 4.10 and Table 4.11 provide the main results of this case. Figure 4.16 and 4.17 illustrates the predicted trend with all the testing data points.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.10 Results comparison when using different numbers of centres and RBFs for parabolas trend case.

RBF	$m=9,\sigma=0.2$		$m=7,\sigma=0.3$		$m=5,\sigma=0.5$		$m=4,\sigma=0.6$	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
CS-RBF1	0.2432	0.41758	0.24342	0.40239	0.26359	0.3919	0.25937	0.39388
CS1	0.22953	0.43281	0.24674	0.39482	0.26100	0.3992	0.25813	0.40699
CS2	0.24407	0.40677	0.25483	0.40454	0.25456	0.39304	0.24952	0.40052
Gaussian	0.24256	0.3985	0.24649	0.3960	0.24742	0.39345	0.25371	0.39465

Table 4.11 Listing of basis function centres for parabolas trend case.

m	RBF	Weights								
		μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9
9	**CS-RBF1**	0	0.9293	0.8182	0.1414	0.2525	0.5354	0.7172	0.0707	0.3232
	CS1	0.5051	0.1616	0.0707	0.8485	0.9293	0.7576	0.4141	0.6667	1.0000
	CS2	0.4747	1.0000	0.4242	0.5253	0.2323	0.6162	0.0909	0.7677	0.9596
	Gaussian	0	1.0000	0.0808	0.9192	0.2121	0.7879	0.3535	0.6465	0.5051
7	**CS-RBF1**	0	1.0000	0.4646	0.5354	0.1010	0.8586	0.7172		
	CS1	0.8485	0.2424	0.6667	1.0000	0	0.1616	0.9293		
	CS2	0	0.9091	0.5253	0.2828	0.8081	0.7172	0.0404		
	Gaussian	0	1.0000	0.1010	0.8889	0.7071	0.2929	0.4949		
5	**CS-RBF1**	0.4646	1.0000	0.7172	0.0303	0.9697				
	CS1	0	0.7576	0.0707	0.6667	0.8485				
	CS2	0	0.7172	1.0000	0.3838	0.4747				
	Gaussian	1.0000	0	0.8081	0.2020	0.5051				
4	**CS-RBF1**	0.9697	0.7879	0.6465	0					
	CS1	0.1616	0	1.0000	0.4141					
	CS2	1.0000	0.4747	0.0404	0					
	Gaussian	1.0000	0	0.7172	0.3030					

มหาวิทยาลัยเทคโนโลยีสุรนารี

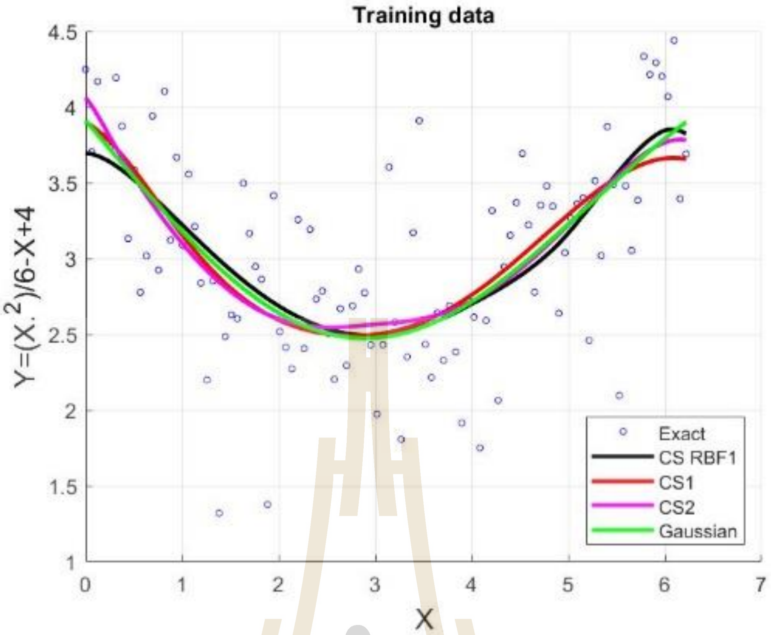


Figure 4.16 Predicted training trend produced by using 4 centres for parabolas trend case.

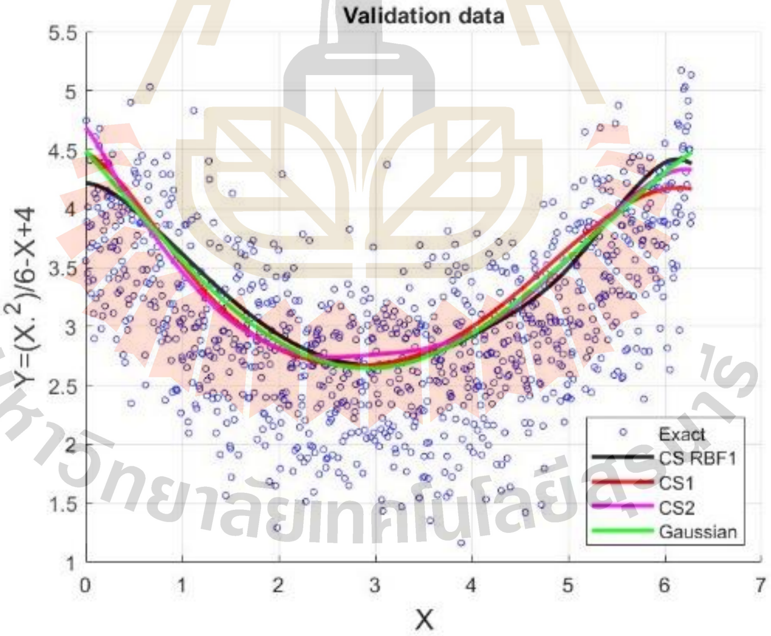


Figure 4.17 Predicted validation trend produced by using 4 centres for parabolas trend case.

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.2.5 Test Case 3: Sine interpolation

In the final case study, a non-linear sine function expressed below is studied;

$$y = \sin(x) + \varepsilon. \tag{4.8}$$

A set of 100 data on $[0, 2\pi]$ is generated and the noise is set to be a Gaussian distribution with mean zero and standard deviation 0.5, as depicted in Figure 4.18 and Table 4.12.

Table 4.12 The locations over the computational domain with 100 points for sine trend case.

i	x_i	y_i	i	x_i	y_i
1	0	0.358921	51	3.141593	-0.57653
2	0.062832	1.287004	52	3.204425	-0.01115
3	0.125664	-1.38256	53	3.267256	-0.93582
4	0.188496	0.762927	54	3.330088	-0.9307
5	0.251327	0.461482	55	3.39292	-0.25326
6	0.314159	-0.56393	56	3.455752	0.714093
7	0.376991	0.078679	57	3.518584	-0.88192
8	0.439823	0.654499	58	3.581416	-0.17786
9	0.502655	2.870519	59	3.644247	-0.63234
10	0.565487	2.38457	60	3.707079	0.210066
11	0.628319	-0.31333	61	3.769911	-1.31479
12	0.69115	2.663393	62	3.832743	-0.61569
13	0.753982	1.168792	63	3.895575	-0.31571
14	0.816814	0.686876	64	3.958407	0.005746
15	0.879646	1.247641	65	4.021239	0.260328
16	0.942478	0.672192	66	4.08407	-0.75165
17	1.00531	0.761455	67	4.146902	-1.84004
18	1.068142	1.870757	68	4.209734	-1.37183
19	1.130973	1.845431	69	4.272566	-1.61349
20	1.193805	1.875826	70	4.335398	0.639276
21	1.256637	1.399316	71	4.39823	-1.362
22	1.319469	0.162523	72	4.461062	-0.4692

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.12 (continued).

i	x_i	y_i	i	x_i	y_i
23	1.382301	1.461081	73	4.523893	-1.11074
24	1.445133	2.080381	74	4.586725	-0.39892
25	1.507964	1.324389	75	4.649557	-1.5086
26	1.570796	1.690711	76	4.712389	-1.93609
27	1.633628	1.48326	77	4.775221	-1.94754
28	1.69646	0.789552	78	4.838053	-0.66622
29	1.759292	1.178462	79	4.900885	-1.10069
30	1.822124	0.443031	80	4.963716	-1.09946
31	1.884956	1.544107	81	5.026548	-0.00359
32	1.947787	0.164048	82	5.08938	-0.73513
33	2.010619	0.191301	83	5.152212	-0.77278
34	2.073451	0.335924	84	5.215044	0.183565
35	2.136283	-1.12113	85	5.277876	-1.38135
36	2.199115	1.76921	86	5.340708	-0.34398
37	2.261947	0.987595	87	5.403539	-0.21305
38	2.324779	0.225015	88	5.466371	-0.89166
39	2.38761	1.599292	89	5.529203	-0.54058
40	2.450442	-0.5051	90	5.592035	-1.41569
41	2.513274	0.519533	91	5.654867	-1.3541
42	2.576106	0.374648	92	5.717699	-0.46582
43	2.638938	0.694841	93	5.78053	0.000388
44	2.70177	0.634629	94	5.843362	1.30017
45	2.764602	-0.20923	95	5.906194	-0.81331
46	2.827433	0.288956	96	5.969026	-0.18396
47	2.890265	0.138625	97	6.031858	-0.30376
48	2.953097	0.606408	98	6.09469	-1.47777
49	3.015929	0.855145	99	6.157522	-0.41837
50	3.078761	0.803288	100	6.220353	-1.26083

มหาวิทยาลัยเทคโนโลยีสุรนารี

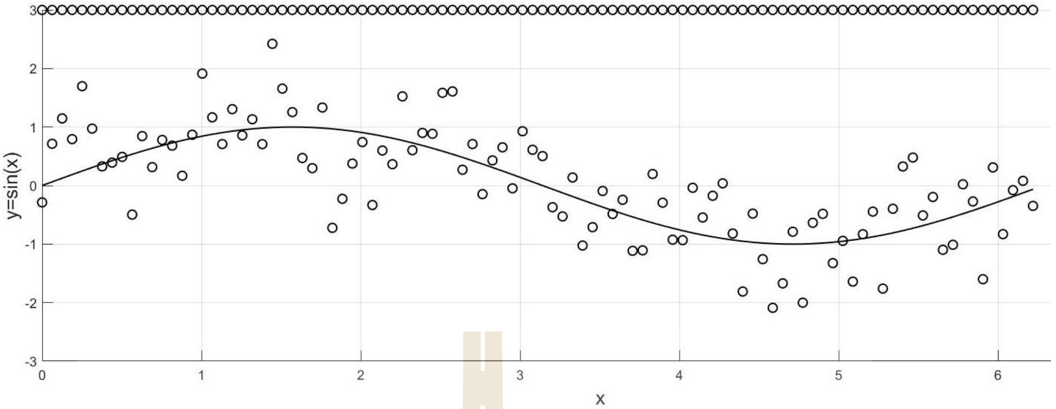


Figure 4.18 Data with 100 points for sine trend case.

With using RC Algorithm, the parameters σ and m are optioned for both training and validation cases are shown in Table 4.13.

Table 4.13 Training and validation errors for candidate models for sine trend case.

RBF	σ	m	MSE	
			Training	Validation
CS-RBF1	-	29	0.22723	0.78921
CS1	-	13	0.30183	0.64521
CS2	-	22	0.24992	0.73249
Gaussian I	0.2	9	0.36463	0.53588
Gaussian II	0.3	7	0.36596	0.53268
Gaussian III	0.4	5	0.37101	0.52852
Gaussian IV	0.5	5	0.37275	0.52933
Gaussian V	0.6	4	0.37927	0.52375
Gaussian VI	0.7	4	0.37985	0.52492

มหาวิทยาลัยเทคโนโลยีสุรนารี

Thus, based on RC Algorithm, Gaussian V with $\sigma = 0.6$ and $m = 4$ for this data set can be a good model. When using the same numbers of centres, Table 4.14 and Table 4.15 provide the main results of this case. Figure 4.19 and Figure 4.20 illustrates the predicted trend with all the testing data points.

Table 4.14 Results comparison when using different numbers of centres and RBFs for sine trend case.

RBF	$m=9, \sigma=0.2$		$m=7, \sigma=0.3$		$m=5, \sigma=0.5$		$m=4, \sigma=0.6$	
	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error	Training Error	Validation Error
CS-RBF1	0.34868	0.55688	0.37794	0.53255	0.45835	0.66754	0.46907	0.63571
CS1	0.35783	0.54659	0.34087	0.57013	0.47041	0.66921	0.38884	0.53472
CS2	0.33763	0.58504	0.36135	0.56636	0.41765	0.55269	0.52803	0.80572
Gaussian	0.36463	0.53588	0.36596	0.53268	0.37275	0.52933	0.37927	0.52375

Table 4.15 Listing of basis function centres for sine trend case.

m	RBF	Weights								
		μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9
9	**CS-RBF1**	0	0.4646	0.8182	0.8586	0.0707	0.5758	0.1414	0.7475	0.2525
	CS1	0.1616	0.5051	0.5859	0.7576	0.0707	0.4141	0	1.0000	0.6667
	CS2	0	0.9091	0.7172	0.5253	0.6162	0.0404	0.2323	0.6667	0.3333
	Gaussian	0	1.0000	0.0808	0.9192	0.2121	0.7879	0.3535	0.6465	0.5051
7	**CS-RBF1**	0.8586	0.0303	1.0000	0.1414	0	0.2828	0.5051		
	CS1	0.3333	0.5051	0.2424	0.7576	0.8485	0.6667	0.5859		
	CS2	0.5253	0.1919	0.2828	0.7172	0.3838	0.5758	0.8586		
	Gaussian	0	1.0000	0.1010	0.8889	0.7071	0.2929	0.4949		
5	**CS-RBF1**	0.6768	0.2828	0.7172	0.4242	0				
	CS1	0.7576	0.4141	0.2424	0.1616	0				
	CS2	0.6162	0.2323	0.2828	0.9091	0.1919				
	Gaussian	1.0000	0	0.1818	0.8081	0.4949				
4	**CS-RBF1**	0.7475	0.4242	0.6768	0.2121					
	CS1	0.7576	0	0.3333	0.8485					
	CS2	0.7172	0.1919	0.4747	0.0404					
	Gaussian	1.0000	0	0.7172	0.3030					

มหาวิทยาลัยเทคโนโลยีสุรนารี

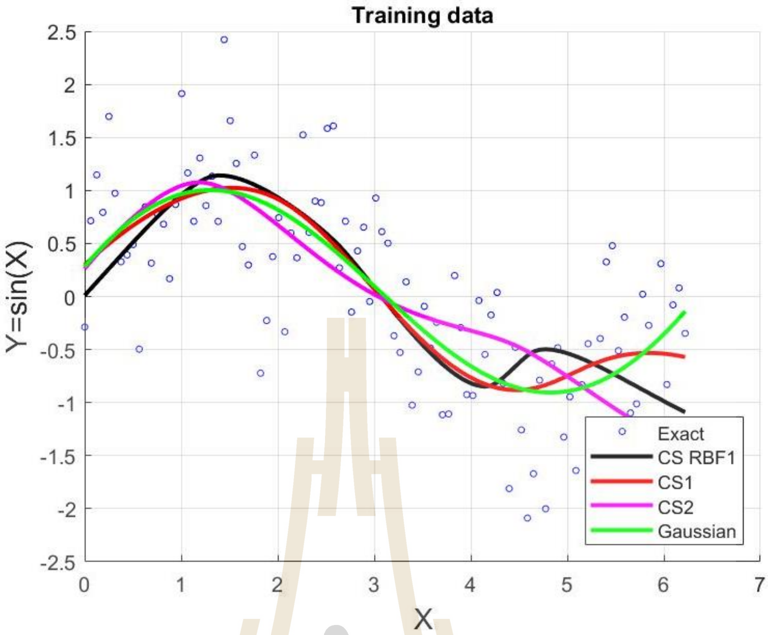


Figure 4.19 Predicted training trend produced by using 4 centres for sine trend case.

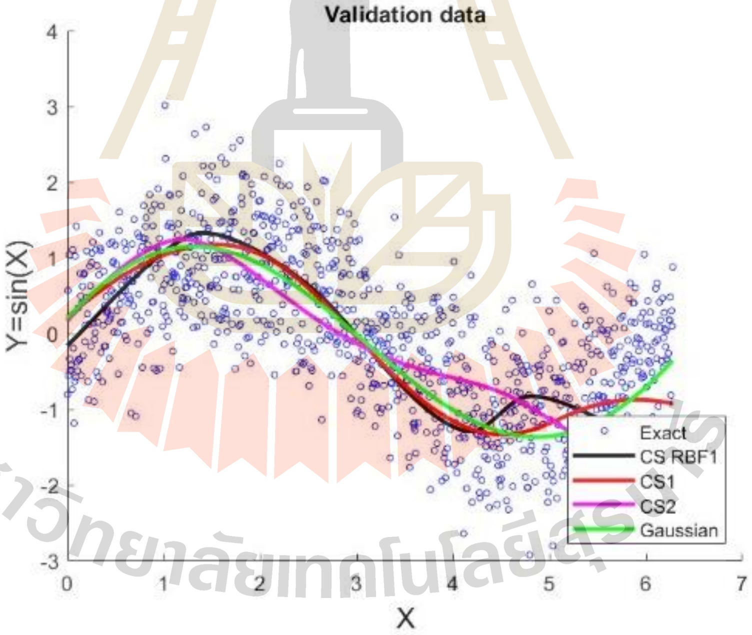


Figure 4.20 Predicted validation trend produced by using 4 centres for sine trend case.

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.2.6 Main Conclusions

The investigation begins with the observation that RBF-pattern recognition problem relies highly on the choice of what is called ‘shape parameter’ which requires a user’s pre-judgement. It then comes to attention an alternative way to avoid this difficulty by using RBFs that contain no-parameter. For this, three non-parameterized RBF have been explored numerically. For the comparison purpose, the RC algorithm normally used with Gaussian RBF is utilized for suggestion for the number of suitable centres. Three trends of data patterns are tested with the scheme and the main findings are;

- With RC-algorithm, it is found that CS-RBF1, CS1, and CS2 have appeared to be slightly overfitting. This figure is to be further investigated.
- With appropriate choice of number of centres, these selected non-parameterized RBFs are found to perform equally well when compared to the famous Gaussian.

The next step of this study is to tackle problems in more dimensions and with more complexity.

4.3 Experiment 3: A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks

4.3.1 The Project Objective

This study constructs the model with unknown input-output mapping pattern under the problem of pattern recognition via RBF-Neural Network using RC-algorithm. In this work, we focus on radial basis function and investigate the capability of other RBFs containing no shape parameter by comparison with 3 choices of shape parameters for the same problem of pattern recognition. For the purpose, to introduce three popular shape parameter acquisition algorithms commonly used with Multiquadric RBF are used as a guide for selecting the optimal shape parameter initially.

4.3.2 RBFs and Their Shape Parame

4.3.2.1 The Selected RBFs

For the purpose of comparison, three forms of radial basis functions are focused on and they are listed as follows;

มหาวิทยาลัยเทคโนโลยีสุรนารี

Gaussian RBF (GA):

$$\phi(r) = \exp - r^{2} / 2\varepsilon^{2} \tag{4.9}$$

Multiquadric RBF (MQ):

$$\phi(r) = \sqrt{\varepsilon^{2} + r^{2}} \tag{4.10}$$

Compactly-Supported RBF (CS1):

$$\phi(r) = \frac{112}{45} r^{\frac{9}{2}} + \frac{16}{3} r^{\frac{7}{2}} - 7r^{4} - \frac{14}{15} r^{2} + \frac{1}{9}. \tag{4.11}$$

The first two are known to be popular in many areas of research and are determined by the choice of the so-called ‘shape parameter, ε’, (Shin, 1998) . The third one is included in this study with the main aim to monitor the performance and compare with the other two RBFs driven by ε . It is also discovered in our previous study (Tavaen, Viriyapong and Kaennakham, 2020) that amongst three forms of Compactly-Supported RBFs, CS1 is more simply implemented with no significant differences in final results quality. This is the main reason why it is being further investigated in this work.

4.3.2.2 The Parameter’s Choices

In this work, the main attention is paid on the effect of choices of the shape parameter proposed in three well-known numerical researches (under the applications of interpolation) and they are briefly detailed below.

Choice I: Hardy Shape (HS)

This was proposed by Hardy (Hardy, 1977) and it is of the following form:

$$\varepsilon_{HS} = 0.815d \tag{4.12}$$

where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$ and d_i is the mean distance from each data point (x_i, y_i) to its nearest neighbor.

Choice II: Franke Shape (FS)

This was invented by Franke (Franke, 1979) and uses the following formula:

$$\varepsilon_{FS} = \frac{1.25D}{\sqrt{N}} \tag{4.13}$$

where D is the diameter of the smallest circle containing all data points.

Choice III: Carlson Shape (CS)

It was formed by Carlson (Carlson and Foley, 1991) and it starts by computing the least squares bivariate quadratic polynomial fit to the data $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ and denoting this quadratic by $q(\bar{x}_i, \bar{y}_i)$, i.e. compute $V = \sum_{i=1}^{N} \frac{\left(\bar{z}_i - q(\bar{x}_i, \bar{y}_i)\right)^2}{N}$ by

มหาวิทยาลัยเทคโนโลยีสุรนารี

setting $\bar{x}_i = \frac{(x_i - x_{\min})}{(x_{\max} - x_{\min})}$, $\bar{y}_i = \frac{(y_i - y_{\min})}{(y_{\max} - y_{\min})}$ and $\bar{z}_i = \frac{(z_i - z_{\min})}{(z_{\max} - z_{\min})}$. The proposed form of shape parameter is then of the following:

$$\varepsilon_{CS} = \frac{1}{1+120V}. \qquad (4.14)$$

These three interesting forms of ε are numerically tested with 1D and 2D pattern applications and the results are discussed in the following section.

4.3.2.3 Additional Criteria

The following are experiment's ingredients used in this study.

1. The whole dataset for each case is split into two parts namely 'training dataset (TD)' and 'validation dataset (VD)', and they are fixed throughout the investigation.
2. $TD \cap VD = \{\ \} = empty\ set$.
3. Define TVN represents the ratio of "$TD:VD$"
4. Defining 'The Model Selection Criteria (MSC)', the best model is ideally the one with the following properties;
    - 4.1. produces small validation error.
    - 4.2. requires small number of centres, i.e. small m in RC-algorithm.
    - 4.3. the number of centres, m, should be as less affected by TD as possible.

(This is all to compromise the problem of over- and under-fitting scenarios.)

4.3.3 Test Case 1: The Sine Function

In the first case study, a sine function expressed below is investigated.

$$y = \sin(x) + \omega \qquad (4.15)$$

A set of 100 data on $[0, 2\pi]$ is generated and the noise ω is set to be a Gaussian distribution with mean zero and standard deviation 0.5. The data points are generated uniformly over the domain with 1,000 points.

The experiments are conducted covering a wide range of TVN from 100:900 to 900:100 (only some are shown here, however). Table 4.16 provides information concerned in each case and it is clearly seen that the number of centres is significantly increasing with more training data being involved in the case of HS for

มหาวิทยาลัยเทคโนโลยีสุรนารี

both GA ($29 \le m \le 135$) and MQ ($22 \le m \le 49$). On the other hand, FS and CS lead to a much narrow range of m over the same interval of TD for both GA and MQ, indicating that FS and CS are comparatively less affected by TD. For all cases, it should be noted that the shapeless CS1 is not being at all affected by the increase in training dataset TD.

In terms of validation error trends, at every ratio of TVN, it can be seen from the Table 4.16 that HS produces the comparatively highest error for both GA and MQ. On the other hand, FS and CS are found to generate small error and stay very close to each other for all TVN shown. It is also found that errors produced by CS1 is smaller than that obtained from HS but higher than that of FS and CS. For the case of GA only, pattern simulations obtained from all the shape choices are illustrated in Figure 4.21 and in Figure 4.22.

To sum up for this sine-function experiment, as long as the MSE is concerned, the shape parameter choice proposed by Franke (Franke, 1979) can be considered as the best.

Table 4.16 Error comparison for some chosen different TVN cases for sine function.

TVN	RBF	Type of ε	ε	m	MSE	
					Training	Validation
100:900	GA	HS	0.0512	29	0.3721	0.9191
		FS	0.7775	4	0.4709	0.7535
		CS	0.2403	8	0.4362	0.8176
	MQ	HS	0.0512	22	0.3926	0.8906
		FS	0.7775	4	0.4701	0.7526
		CS	0.2403	9	0.4383	0.8132
	CS1	-	-	13	0.4259	0.8352
200:800	GA	HS	0.1322	12	0.4534	0.6915
		FS	0.5526	4	0.4677	0.6662
		CS	0.2403	8	0.4362	0.8176

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.16 (continued).

TVN	RBF	Type of ε	ε	m	MSE	
					Training	Validation
100:900	MQ	HS	0.0512	22	0.3926	0.8906
		FS	0.7775	4	0.4701	0.7526
		CS	0.2403	9	0.4383	0.8132
	CS1	-	-	13	0.4259	0.8352
200:800	GA	HS	0.1322	12	0.4534	0.6915
		FS	0.5526	4	0.4677	0.6662
		CS	0.2401	8	0.4582	0.6806
	MQ	HS	0.1322	13	0.4504	0.6981
		FS	0.5526	5	0.4644	0.6632
		CS	0.2401	9	0.4581	0.6800
	CS1	-	-	13	0.4519	0.6949
250:750	GA	HS	0.0205	69	0.3449	0.7334
		FS	0.4947	5	0.4587	0.6316
		CS	0.3326	6	0.4577	0.6340
	MQ	HS	0.0205	36	0.4093	0.6919
		FS	0.4947	5	0.4590	0.6328
		CS	0.3326	7	0.4579	0.6340
	CS1	-	-	13	0.4498	0.6428
500:500	GA	HS	0.0102	135	0.3415	0.6067
		FS	0.3505	6	0.4759	0.5014
		CS	0.3552	6	0.4758	0.5014
	MQ	HS	0.0102	49	0.4271	0.5533
		FS	0.3505	7	0.4740	0.5020
		CS	0.3552	7	0.4740	0.5020
	CS1	-	-	13	0.4720	0.5044

มหาวิทยาลัยเทคโนโลยีสุรนารี

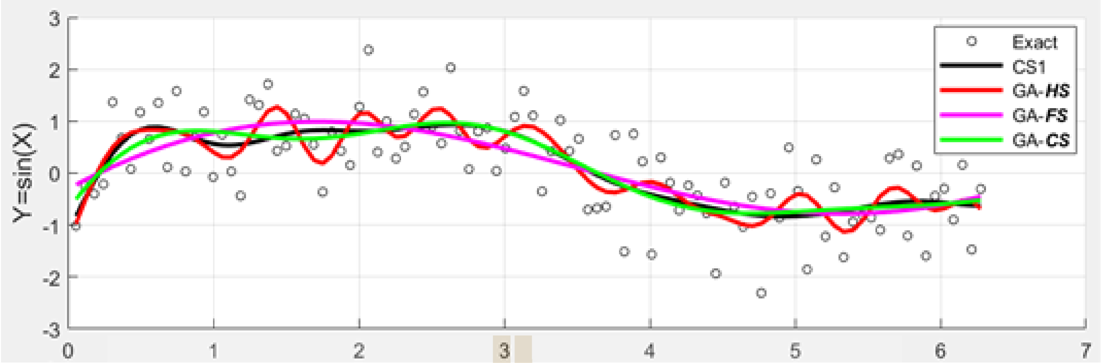


Figure 4.21 Pattern reconstruction obtained from using Gaussian RBF with all three choices of shape parameter compared with that produced by the shapeless CS1-RBF where training phase, calculated using TVN=(100:900).

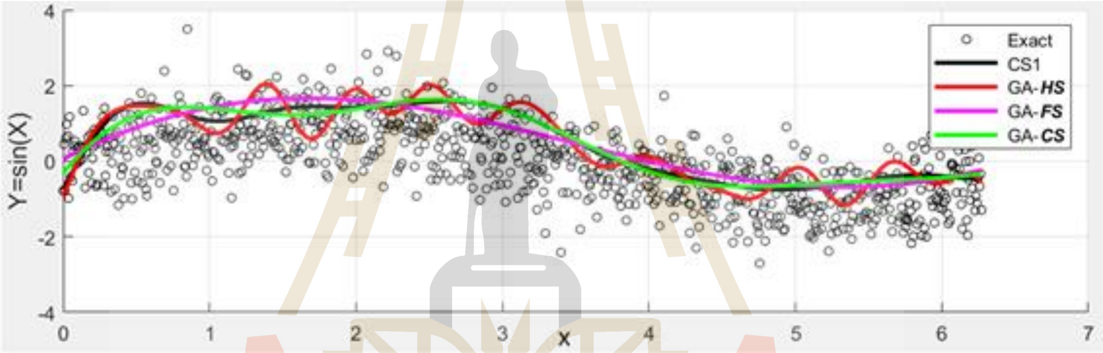


Figure 4.22 Pattern reconstruction obtained from using Gaussian RBF with all three choices of shape parameter compared with that produced by the shapeless CS1-RBF where validation phase, calculated using TVN= (100:900).

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.3.4 Test Case 2: The Famous Franke's Function

This experiment is concerned with one of the most well-known testing functions invented by Franke (Franke, 1982). The complication of the function's surface makes it a good test for validating any interpolation-based methods proposed over the past two decades. The expression of the function is as followed.

$$f(x,y) = 0.75\exp\left(-\frac{(9x-2)^2+(9y-2)^2}{4}\right)+0.75\exp\left(-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}\right) \tag{4.16}$$
$$+0.5\exp\left(-\frac{(9x-7)^2+(9y-3)^2}{4}\right)-0.2\exp(-(9x-4)^2-(9y-7)^2).$$

Like done in the previous experiment, this function is disturbed with Gaussian distribution with mean zero and standard deviation 15.0 and for simplicity the following function is denoted.

$$f_1(x, y) = f(x, y) + \omega. \tag{4.17}$$

Several ratios of training to validation data points have been carried out. The data points are generated uniformly over the domain with 61×61 points. Figure 4.23 shows the node distribution in the case of using TVN=(121:3600) and Figure 4.24 shows the corresponding exact surface (without noise).

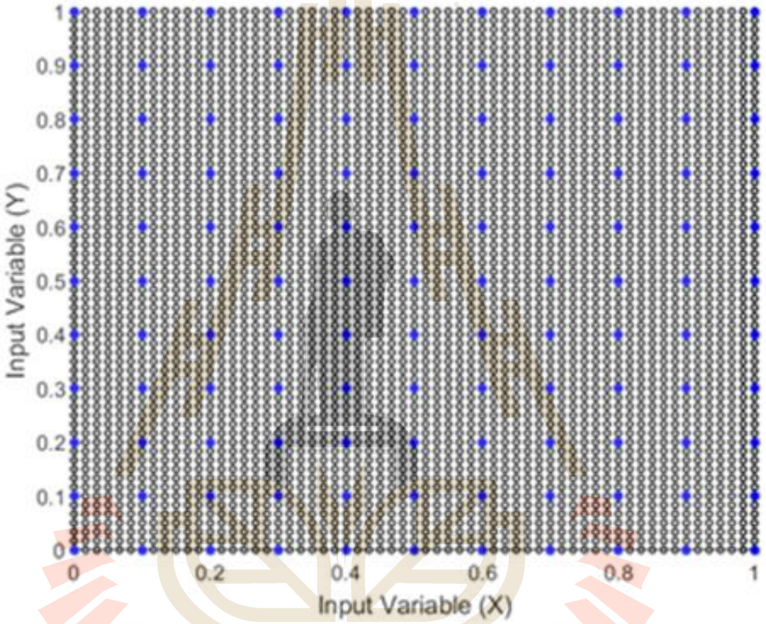


Figure 4.23 Node-distribution, projected on xy-plane, in the case with 121 training data points (depicted with blue dots) and 3600 validation data points (depicted with black dots).

มหาวิทยาลัยเทคโนโลยีสุรนารี

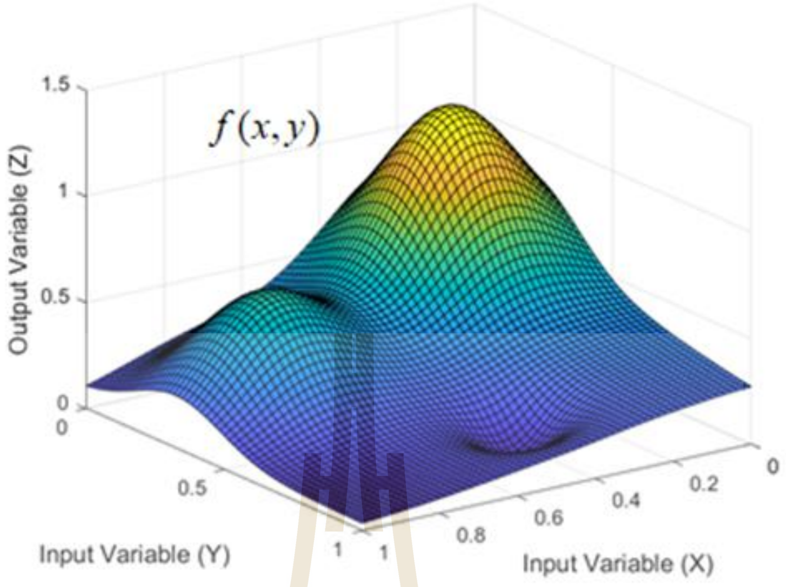


Figure 4.24 Surface plot of $f(x,y)$.

Similar to the previous case of sine function, a number of experiments are conducted and some discovery is provided in Table 4.17. From the Table, the mostly noticeable figure is the training error produced by using HS in both GA and MQ. This over-fitting phenomenon is directly resulted from the fact that it uses the whole set of TD as its centres, i.e. it simply reflects itself for all TVN. Even though it yields small validation errors, it cannot be crowned as the great model. On the other hand, FS and CS are seen to produce even smaller value of validation errors while the training errors still look reasonable. In both GA and MQ, it is found that CS leads to slightly better validation error than FS where the significant figure is revealed in terms of the number of centres required, i.e. m. Not only CS is capable of producing good results with small value of m, it is also seen to be mostly independent of the increase in training data. Even though CS1 is not being affected by TD, the smallest m is still seen in the case of using CS (with GA in particular). For all these reasons, it is concluded in this experiment that CS shape parameter is the best choice for both GA-RBF and MQ-RBF. Solution distributions approximated using GA-CS, MQ-CS and CS1 are illustrated in Figure 4.25 - Figure 4.30.

มหาวิทยาลัยเทคโนโลยีสุรนาร

Table 4.17 Error comparison for some chosen different TVN cases for the Franke's function.

TVN	RBF	Type of ε	ε	m	MSE	
					Training	Validation
121:3600	GA	HS	0.0815	121	1.5864e-30	0.0133
		FS	0.1607	83	0.0023	0.0119
		CS	0.4060	24	0.0063	0.0100
	MQ	HS	0.0815	106	8.4055e-04	0.0123
		FS	0.1607	53	0.0048	0.0098
		CS	0.4060	24	0.0061	0.0097
	CS1	-	-	66	0.0037	0.0106
441:3280	GA	HS	0.0408	441	3.2124e-30	0.0137
		FS	0.0842	257	0.0033	0.0114
		CS	0.4264	19	0.0087	0.0097
	MQ	HS	0.0408	135	0.0052	0.0097
		FS	0.0842	89	0.0059	0.0091
		CS	0.4264	20	0.0079	0.0089
	CS1	-	-	66	0.0064	0.0086
961:2760	GA	HS	0.0272	961	5.0043e-30	0.0137
		FS	0.0570	526	0.0035	0.0117
		CS	0.4103	19	0.0093	0.0091
	MQ	HS	0.0272	155	0.0067	0.0086
		FS	0.0570	113	0.0070	0.0085
		CS	0.4103	21	0.0086	0.0085
	CS1	-	-	66	0.0074	0.0081

มหาวิทยาลัยเทคโนโลยีสุรนารี

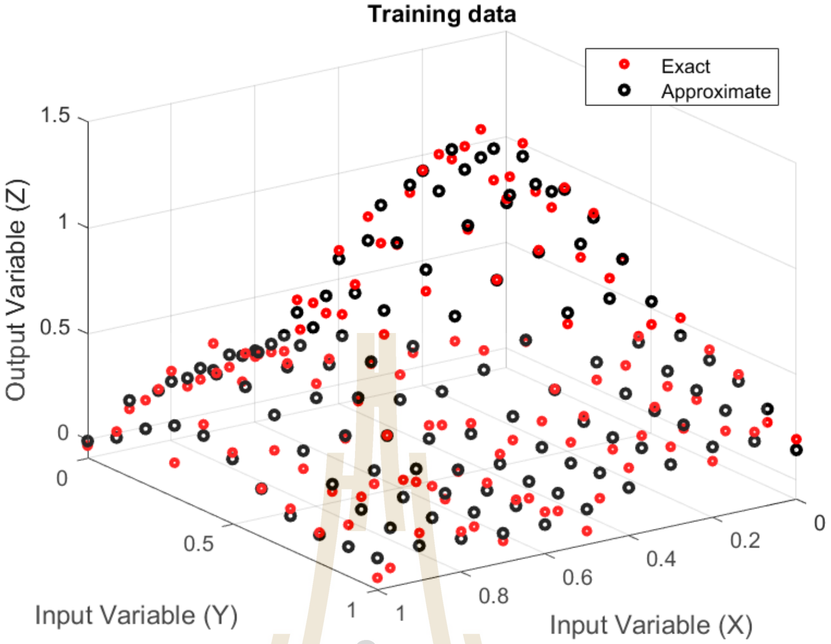


Figure 4.25 Pattern reconstruction yielded by using GA-CS with the training phase (Note that: the 'Exact' means $f_1(x, y)$ and represented in red).

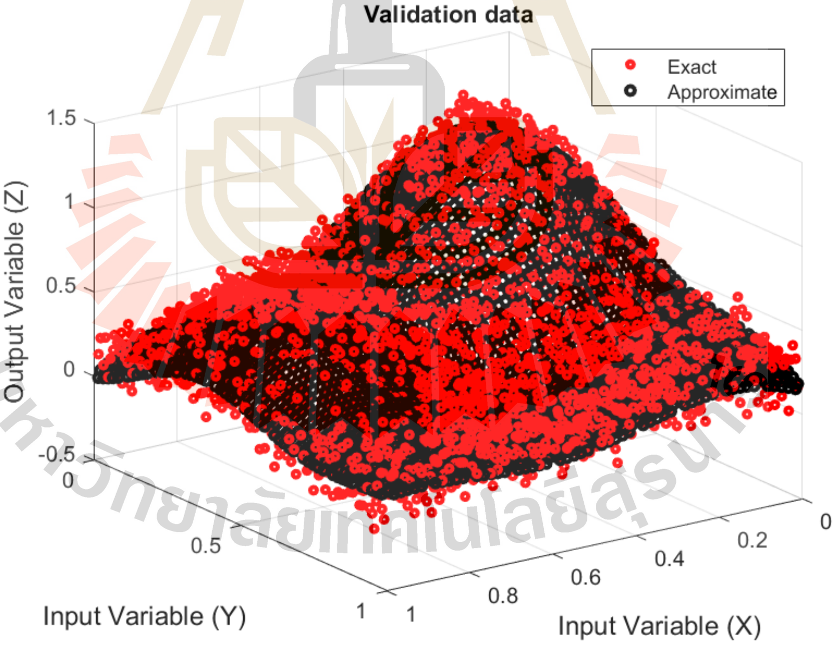


Figure 4.26 Pattern reconstruction yielded by using GA-CS for the validation phase (Note that: the 'Exact' means $f_1(x, y)$ and represented in red).

มหาวิทยาลัยเทคโนโลยีสุรนารี

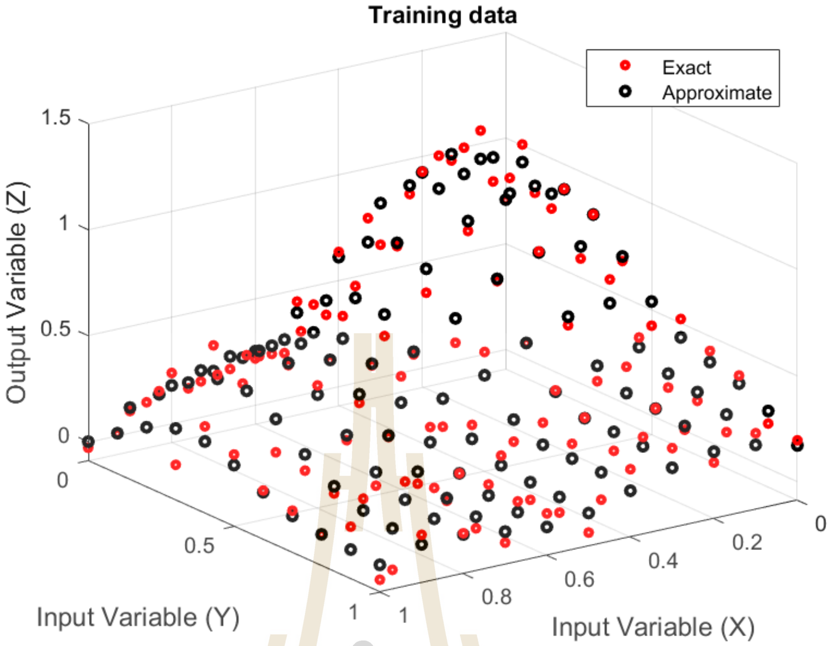


Figure 4.27 Pattern reconstruction yielded by using MQ-CS with the training phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red).

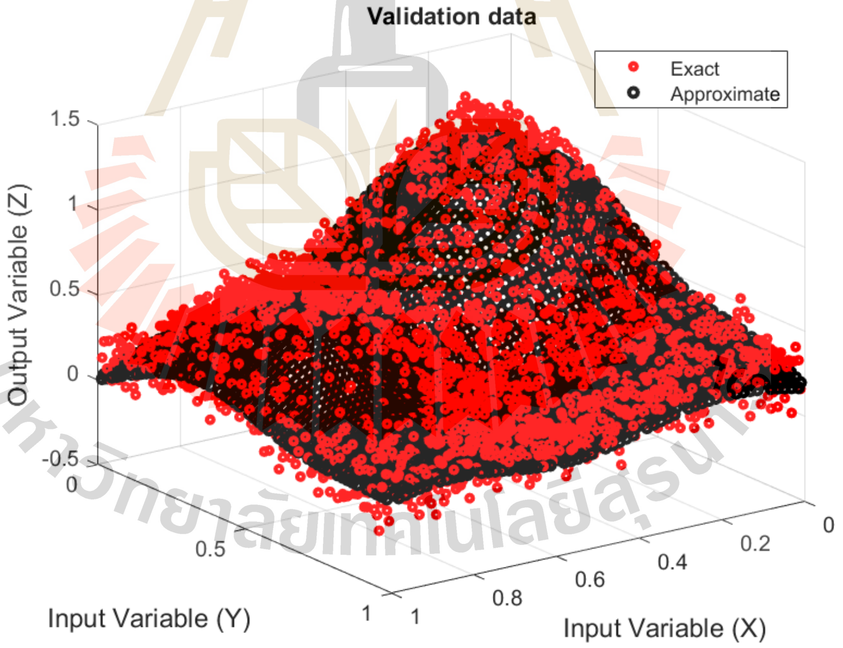


Figure 4.28 Pattern reconstruction yielded by using MQ-CS for the validation phase (Note that: the ‘Exact’ means $f_1(x,y)$ and represented in red).

มหาวิทยาลัยเทคโนโลยีสุรนารี

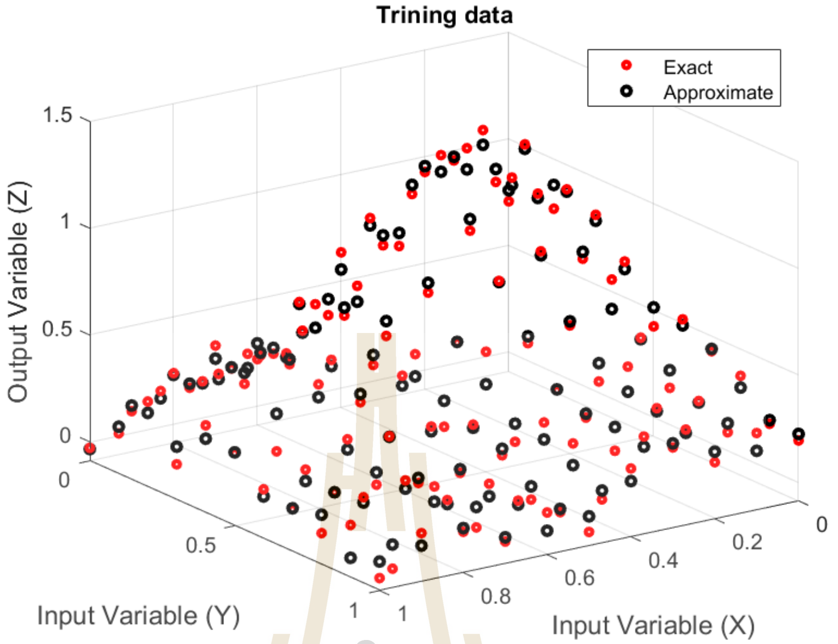


Figure 4.29 Pattern reconstruction yielded by using CS1 with the training phase (Note that: the ‘Exact’ means $f_1(x, y)$ and represented in red).

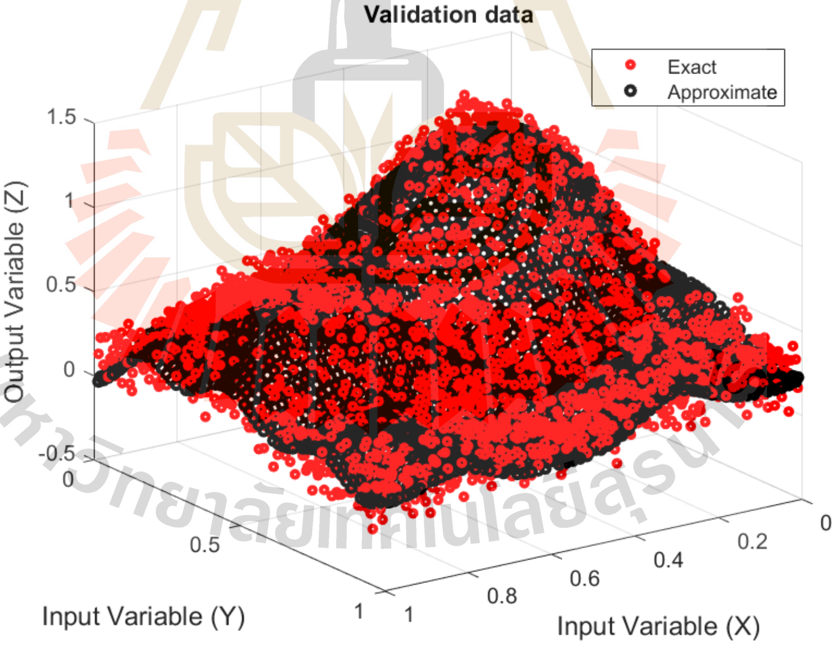


Figure 4.30 Pattern reconstruction yielded by using CS1 for the validation phase (Note that: the ‘Exact’ means $f_1(x, y)$ and represented in red).

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.3.5 Main Conclusion

Under the pattern recognition problem, it can be observed that RBFs involve normally require a suitable shape in order to produce reliable results. The trends observed from the 2 experiments represented in 1D and 2D were as follows:

- With HS for both GA and MQ, the number of centres is significantly increasing with more training data being involved. On the other hand, FS and CS are comparatively less affected. For all cases, it should be noted that the shapefree CS1 is not being at all affected by the increase in training dataset.
- With appropriate choice of validation error, these selected CS are found to perform equally well when compared to the famous GA and MQ.

4.4 Experiment 4: Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery

4.4.1 The Project Objective

The scheme was applied to tackle the problem of approximation of its derivatives using six forms of MQ with two choices of the variable shape parameter. The numerical results obtained in this study shall provide useful information on selecting both a suitable form of MQ and a reliable choice of MQ shape for further applications in general.

4.4.2 Variable Shape Parameters

The main focus of this work is on one of the popular form of basis function known as ‘multiquadric (MQ)’ in its generalized form defined as follows;

$$\phi_j(x,y) = \left(\varepsilon^2 + r_j^2\right)^{\beta} = \left(\varepsilon^2 + (x-x_j)^2 + (y-y_j)^2\right)^{\beta} \qquad (4.18)$$

where $\beta = \ldots, -3/2, -1/2, 1/2, 3/2, \ldots$. These values give direct effect on the singularity of the interpolation matrix and, unavoidably, the effectiveness of the methods’ performance. Different values yield different shape curves depicted in Figure 4.31.

มหาวิทยาลัยเทคโนโลยีสุรนารี

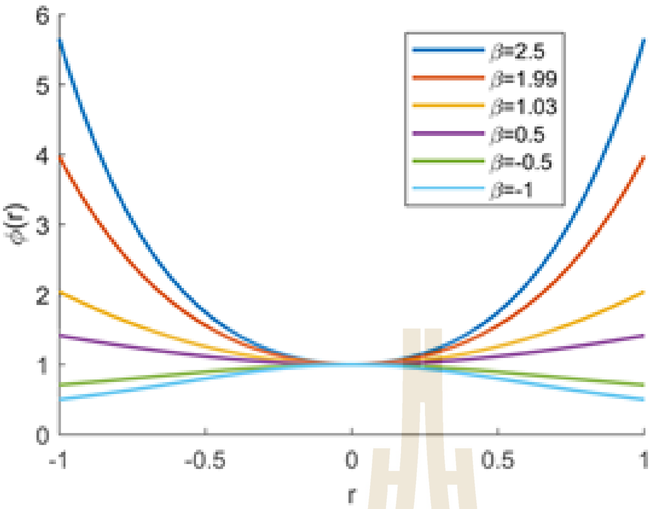


Figure 4.31 Generalized MQ with six values of β's.

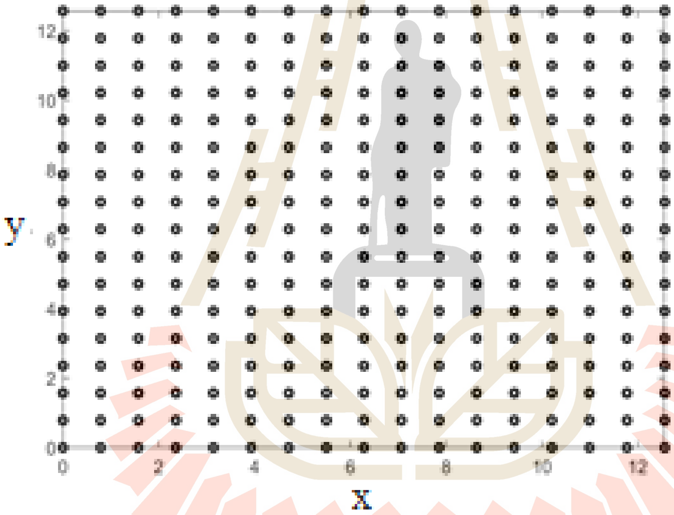


Figure 4.32 Generalized distribution of centres test function.

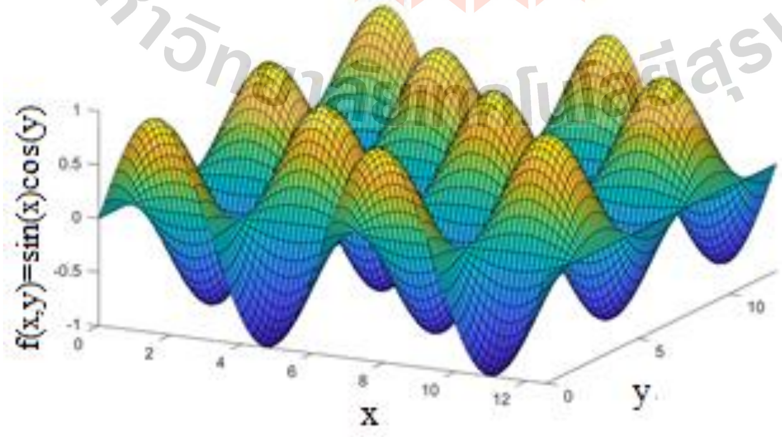


Figure 4.33 Generalized two-dimensional test function.

มหาวิทยาลัยเทคโนโลยีสุรนารี

One crucial factor affecting the method's performance is the shape parameter. Choices available in literature can be categorized into three forms; constant/fixed, variable, and iterative-based. This work focuses on two nonlinear-variable shape strategies (illustrated in Figure 4.34- 4.35)

- Strategy-1 (Stg1): by Nojavan (Nojavan, Abbasbandy and Allahviranloo, 2017), defined as;

$$\varepsilon_j = (c_{\min} + (c_{\max} - c_{\min})\exp(-j))^{-1} \tag{4.19}$$

- Strategy-2 (Stg2): by Xiang et.al. (2012) (Xiang, Wang, Ai, Sha and Shi, 2012), defined as;

$$\varepsilon_j = c_{\min} + (c_{\max} - c_{\min})\sin(j) \tag{4.20}$$

with $j = 1, 2, \ldots, N$. Apart from these chosen forms of shape selection strategies, those nicely documented in (Cheng, 2012) and (Chen, Hong and Lin, 2018) are also highly recommented for interested readers.

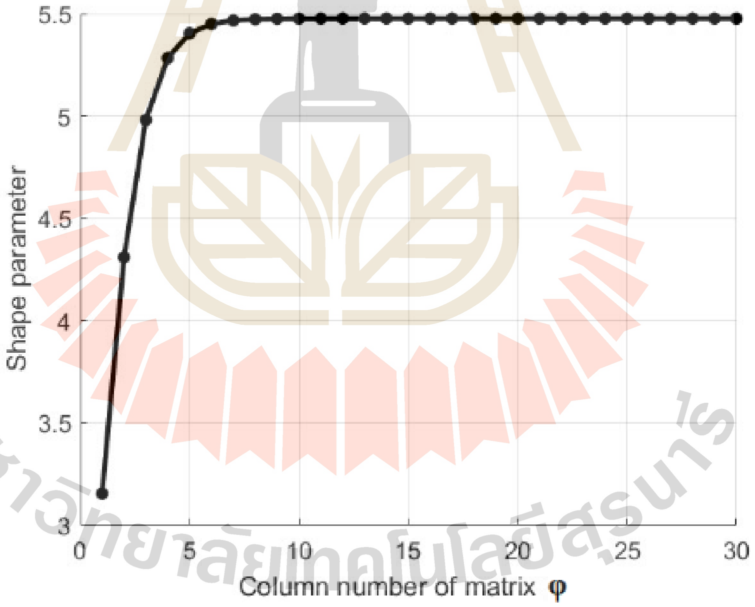


Figure 4.34 Multiquadrics' shape determining Stg1 by using $c_{\min} = 1/\sqrt{N}$, $c_{\max} = 3/\sqrt{N}$, and $N = 30$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

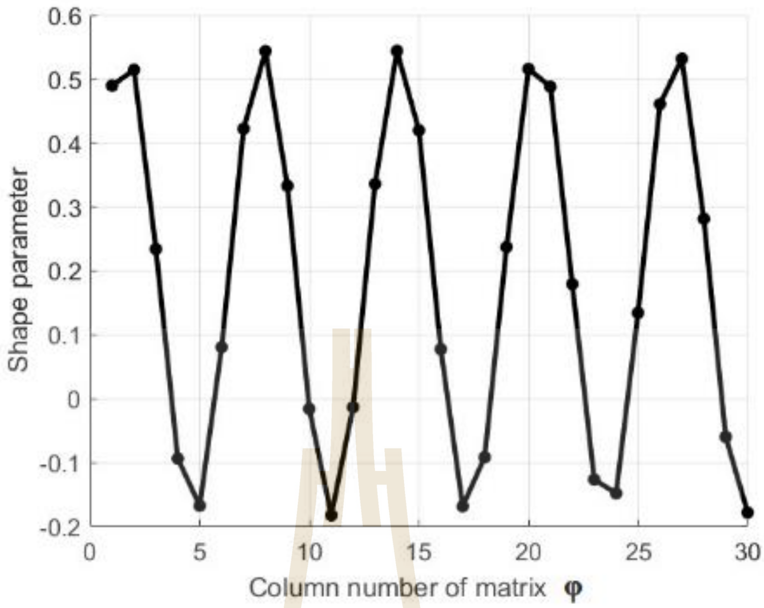


Figure 4.35 Multiquadrics' shape determining Step2 by using $c_{\min} = 1/\sqrt{N}$, $c_{\max} = 3/\sqrt{N}$, and $N = 30$.

4.4.3 Test Case 1: One-Dimensional Case

The first case deals with the approximation of a function in one dimension expressed below.

$$f(x) = e^{x^3} + \cos(2x) \tag{4.21}$$

To compare the overall accuracy, those shape values stated in the work of Maggie (Chenoweth and Sarra, 2009) were also tested. The total of 25 nodes uniformly distributed over a $-1 \le x \le 1$ domain was considered where the function's values at 17 equally-spaced nodes were numerically approximated. The results were obtained from all six forms of MQ with two variable shapes (Step1 and Step2) together with fixed values are presented in Table 4.18. It can be shown in the table that all cases performed reasonably well with not much of a significant difference. One interesting finding worth mentioning is that those values obtained by using a fixed value were found the best ones. This might be attributed to the fact that Maggie (Chenoweth and Sarra, 2009) had actually performed a tremendous amount of experiments to find the best value of shape before utilizing that value for the actual experiment.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 4.18 Mean absolute errors measured for approximation of the function and its derivative for all forms of generalized MQ when using different variable shape forms (compared with a fixed form from literature).

Type	Shape	$\beta = 2.50$	$\beta = 1.99$	$\beta = 1.03$	$\beta = 0.50$	$\beta = -0.50$	$\beta = -1.00$
$f(x)$	Stg1	28.70E-05	12.00E-04	71.60E-05	27.00E-04	11.66E-05	40.57E-05
	Stg2	61.07E-07	15.18E-06	42.77E-06	48.52E-06	42.48E-05	16.00E-04
	Maggie	16.00E-04	16.00E-04	59.44E-05	89.55E-06	74.68E-07	54.15E-07
$f'(x)$	Stg1	22.00E-04	70.00E-04	35.00E-04	81.00E-04	12.00E-04	30.00E-04
	Stg2	29.36E-05	71.61E-05	20.00E-04	23.00E-04	17.80E-03	66.00E-03
	Maggie	33.00E-04	86.00E-04	20.00E-04	21.20E-05	10.05E-05	91.73E-06

4.4.4 Test Case 2: Two-Dimensional Case

A more complex problem is now visited and a two-dimensional function is addressed, defined as follows:

$$f(x,y) = \cos(y) \cdot \sin(x) \tag{4.22}$$

This is defined on a $[0,4\pi]\times[0,4\pi]$ domain where the corresponding centres distribution is depicted in Figure 4.32, and its surface is being visualized in Figure 4.33. To focus on the two main variable shape strategies. Five densities of centres were tested for each strategy to observe the overall behavior of the variable when dealing with larger sizes of datasets. The set of 17×17 equally-spaced is used for the target locations for approximation. As depicted in Figures 4.36 - Figures 4.41, all three forms; the function itself, its x-direction derivative, and its y-direction derivative, reveal the same trends of accuracy. Those functions' approximation produced by shape strategy 1 (or Stg1) is found to be highly sensitive to the increase in centres (N) while Stg2 is not. This could be attributed to the fact that Stg1 is in an exponential curve where it can easily be affected by the pre-judged values of $c_{\min}$ and $c_{\max}$, whose values depend on N. The best accuracy is found for Stg1 when $\beta =$-0.50 and $N =$225 mare used where the comparatively worst results are revealed when $\beta =$2.50 and $N =$625 are used, see Figures 4.36, Figures 4.38 and Figures 4.40. The opposite scenario is found in the use of Stg2 where $\beta =$2.50 is clearly seen to outperform the rest for all three targeted function forms, see Figures 4.37, Figures 4.39 and Figures 4.41. As can be anticipated,

มหาวิทยาลัยเทคโนโลยีสุรนารี

Stg2 produces better solutions, for all values of β's, when more supporting centres are involved, indicating the benefits one can achieve when deploying the strategy for larger datasets. It can also be seen from these same figures that the fastest reduction in error is obtained from $\beta = 2.50$ whereas the slowest one is from $\beta = -1.00$.

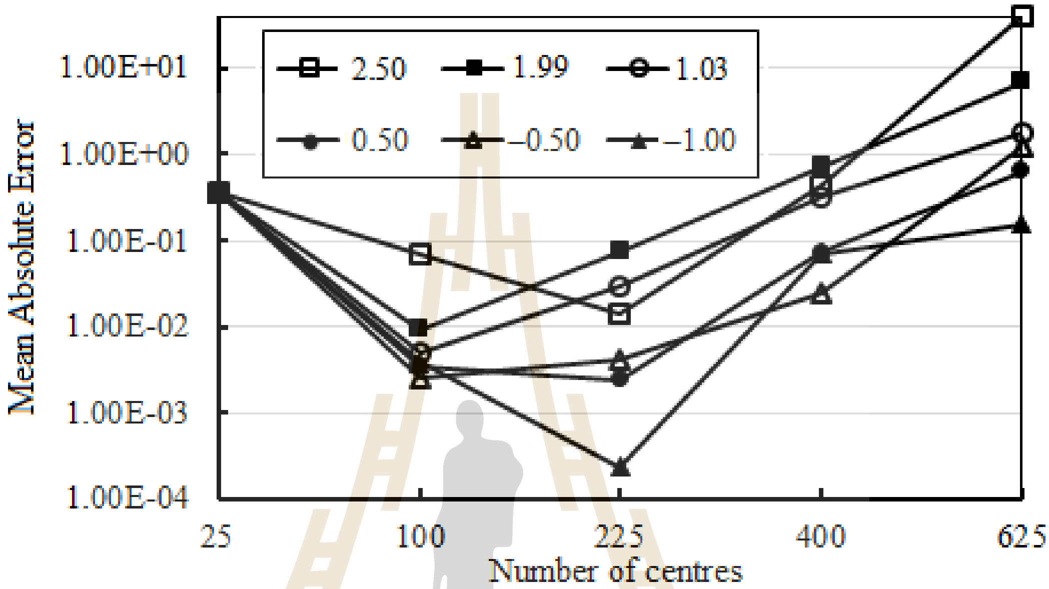


Figure 4.36 Mean absolute errors measured at different numbers of centres for approximation of $f(x,y)$ by using Stg1

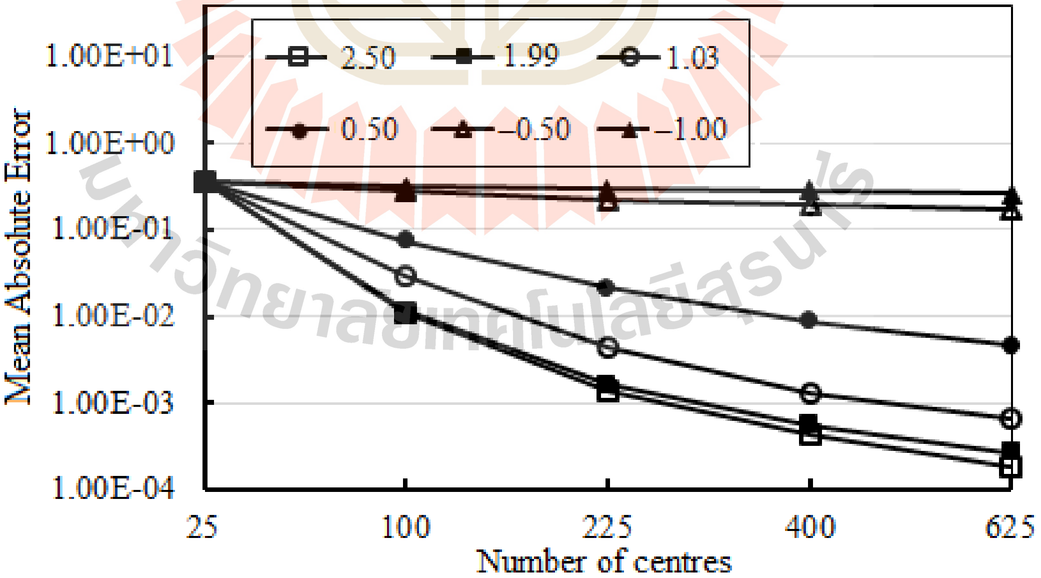


Figure 4.37 Mean absolute errors measured at different numbers of centres for approximation of $f(x,y)$ by using Stg2

มหาวิทยาลัยเทคโนโลยีสุรนารี

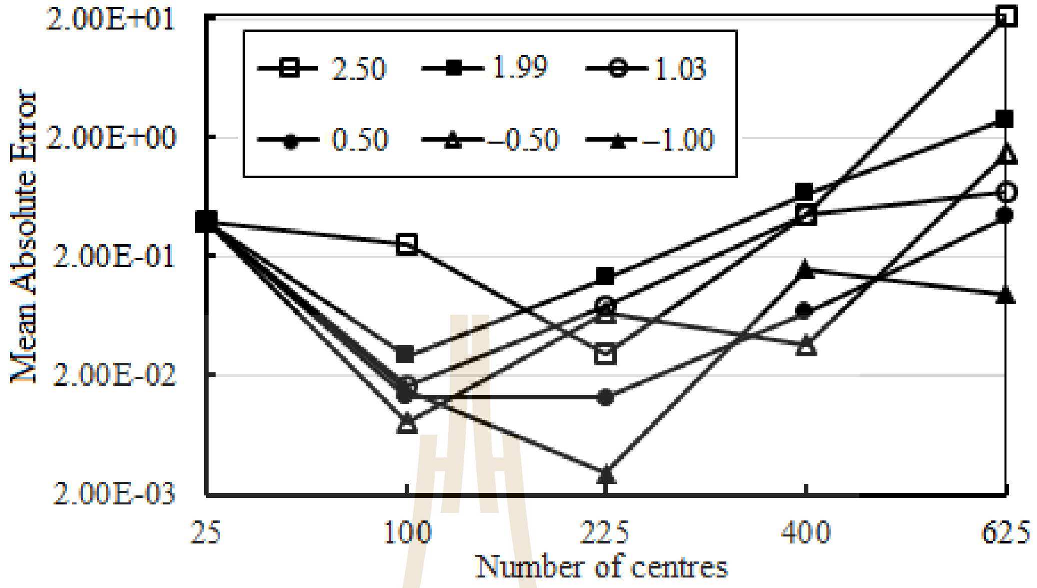


Figure 4.38 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial x$ by using Stg1

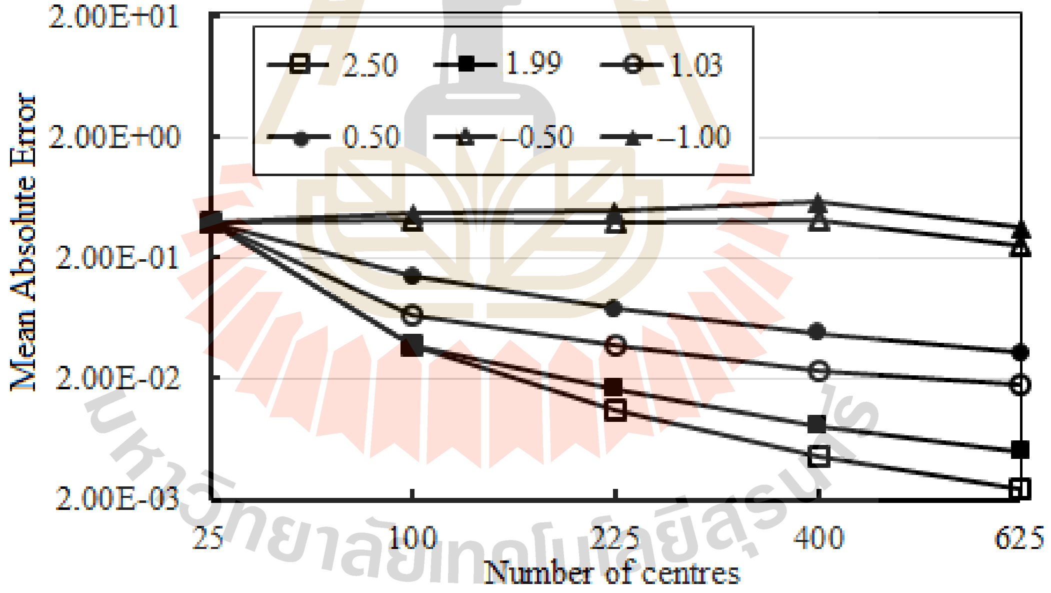


Figure 4.39 Mean absolute errors measured at different numbers of centres for approximation of $\partial f / \partial x$ by using Stg2

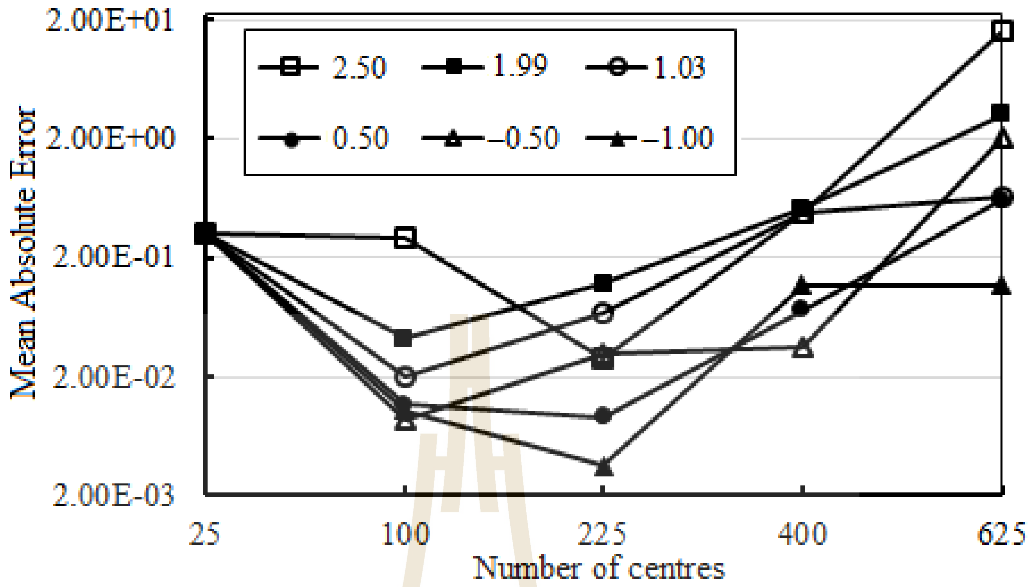


Figure 4.40 Mean absolute errors measured at different numbers of centres for approximation of $\partial f/\partial y$ by using Stg1.

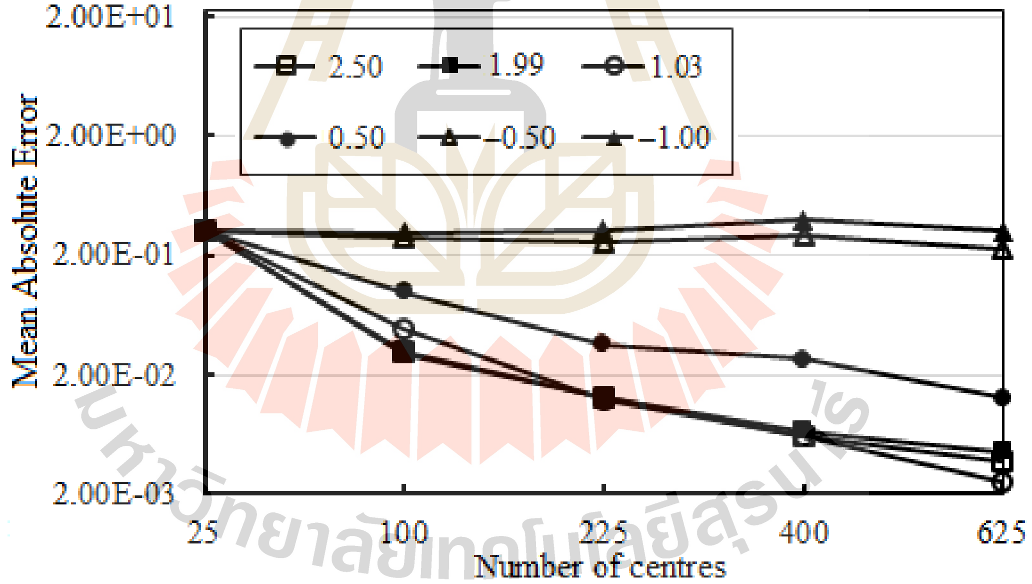


Figure 4.41 Mean absolute errors measured at different numbers of centres for approximation of $\partial f/\partial y$ by using Stg2

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.4.5 Main Conclusions

Emerging as an alternative numerical tool for approximating and recovering functions and their derivatives, multiquadric radial basis function neural networks (MQ-RBFNs) are under investigation in this work. Six generalized forms of MQ were numerically applied using two popular choices of shape parameters. It was found that the exponential variable shape (Stg-1) is highly sensitive to the number of centres for all forms of MQ used. On the other hand, the trigonometric variable shape (Stg-2) provides a monotone of reduction in errors with the increase of centres indicating a more reliable aspect for larger datasets. With the effectiveness in approximating derivatives of a function being also under the study, Stg-2 is seen promising with a great level of accuracy when the form $\varepsilon^2+r^2\;{}^{5/2}$ of MQ is used. This could well have a great effect in the era of solving partial differential equations which shall remain one of our future investigations.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER V

THE MAIN NUMERICAL EXPERIMENT

5.1 The Project Objective

In this work, therefore, the main focus is paid on numerical investigation of the effectiveness of some of the most shapeless RBFs. The challenge is carried out under the context of pattern recognition through the structure of Radial Basis Function Neural Networks (RBF-NNs) where the effectiveness is validated using several criteria; accuracy, condition number, CUP-time and storage, user's interference, and the sensitivity to other factors.

Sixteen forms of shapeless RBFs, all of which have been receiving great attention over the past decade, are paid attention in this work and they are listed in Table 5.1. For the sake of comparison, the well-known shape-contained RBF namely 'Multiquadric, MQ' is also parallelly studied where the shape choosing scheme proposed by Carlson, 1991 (Carlson and Foley, 1991) is used. Differences in the curve profiles for each RBF can clearly be seen and noticed in Figure 5.1 – Figure 5.4.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.1 Type of Radial Basis Function.

No.	Name of RBF	Abbreviation	Definition
1	Multiquadric	MQ	$\sqrt{r^2+\varepsilon^2}$
2	Polyharmonic spline	PS	$\begin{cases} r^{2k-1}, k\in\mathbb{N} \\ r^{2k}\ln\ r\ , k\in\mathbb{N} \end{cases}$
3	Thin plate splines	TPS	$r^2\log r$
4	Wu's CS-RBFs (Wu, 1995)	WU1	$1-r{}_{+}^{7}\ 5r^6+35r^5+101r^4+147r^3+101r^2+35r+5$
5	Wu's CS-RBFs (Wu, 1995)	WU2	$1-r{}_{+}^{6}\ 6r^5+30r^4+72r^3+82r^2+36r+6$
6	Wu's CS-RBFs (Wu, 1995)	WU3	$1-r{}_{+}^{5}\ 5r^4+25r^3+48r^2+40r+8$
7	Wu's CS-RBFs (Wu, 1995)	WU4	$1-r{}_{+}^{4}\ 5r^3+20r^2+29r+16$
8	Wendland's CS-RBFs (Wendland, 1995)	WL1	$1-r{}_{+}$
9	Wendland's CS-RBFs (Wendland, 1995)	WL2	$1-r{}_{+}^{3}\ 3r+1$
10	Wendland's CS-RBFs (Wendland, 1995)	WL3	$1-r{}_{+}^{5}\ 8r^2+5r+1$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.1 (continued).

No.	Name of RBF	Abbreviation	Definition
11	Wendland's CS-RBFs (Wendland, 1995)	WL4	$(1-r)_+^2$
12	Wendland's CS-RBFs (Wendland, 1995)	WL5	$(1-r)_+^4(4r+1)$
13	Wendland's CS-RBFs (Wendland, 1995)	WL6	$(1-r)_+^6(35r^2+18r+3)$
14	Wendland's CS-RBFs (Wendland, 1995)	WL7	$(1-r)_+^8(32r^3+25r^2+8r+1)$
15	Buhmann (Buhmann, 1998)	BUH1	$\left(2r^4\log r-\frac{7r^4}{2}+\frac{16r^3}{3}-2r^2+\frac{1}{6}\right)_+$
16	Buhmann (Buhmann, 1998)	BUH2	$\left(\frac{112r^{\frac{9}{2}}}{45}+\frac{16r^{\frac{7}{2}}}{3}-7r^4-\frac{14r^2}{15}+\frac{1}{9}\right)_+$
17	Buhmann (Buhmann, 1998)	BUH3	$\left(\frac{1}{18}-r^2+\frac{4r^3}{9}+\frac{r^4}{2}-\frac{4r^3\log r}{3}\right)_+$

มหาวิทยาลัยเทคโนโลยีสุรนารี

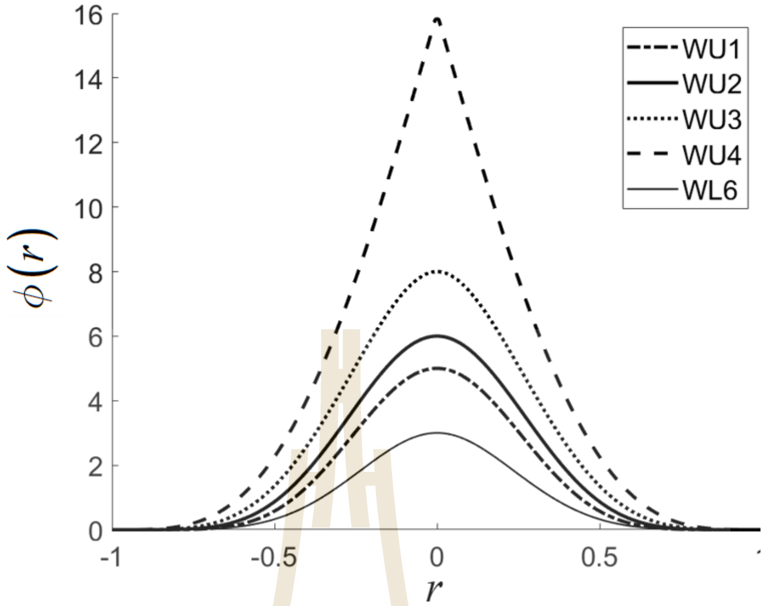


Figure 5.1 One-dimensional curve for some types of RBFs under this investigation of those containing no shapes (or shapefree) and $2 \leq \phi \; r \; \leq 16$.

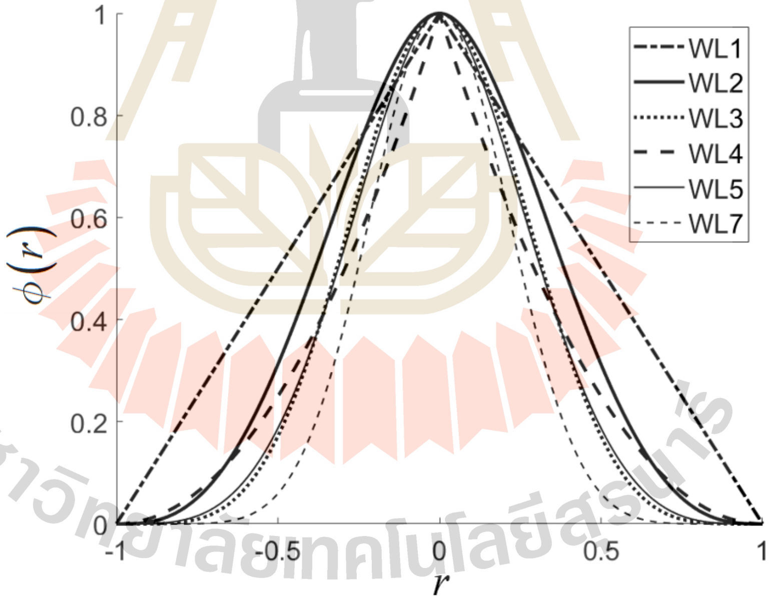


Figure 5.2 One-dimensional curve for some types of RBFs under this investigation of those containing no shapes (or shapefree) and $0.2 \leq \phi \; r \; \leq 2$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

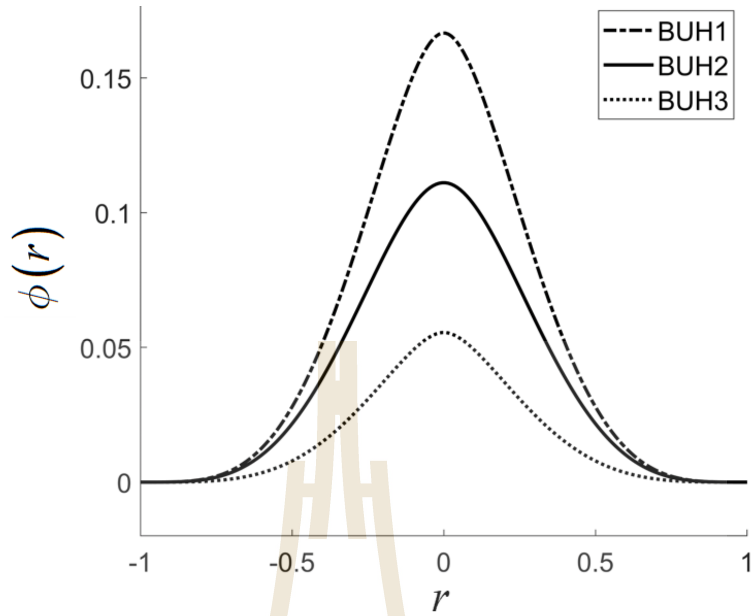
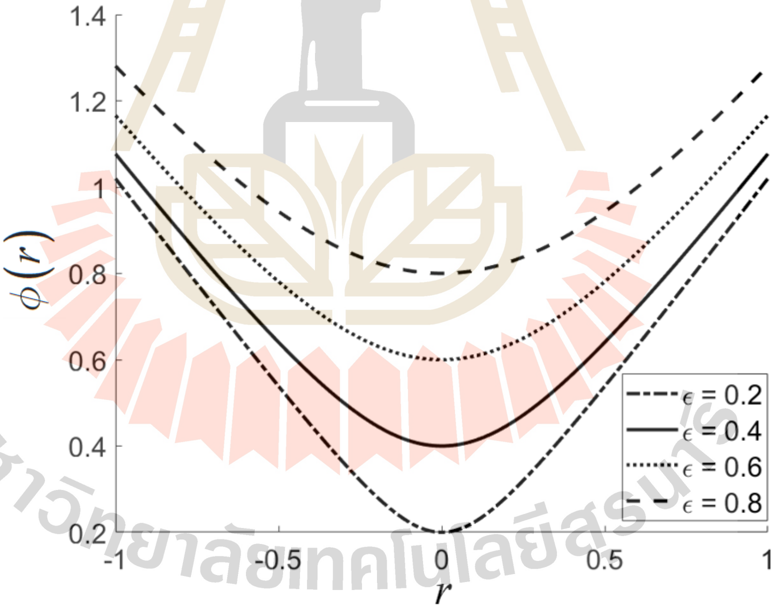


Figure 5.3 One-dimensional curve for some types of RBFs under this investigation of those containing no shapes (or shapefree) and $0 \leq \phi \;\; r \;\; \leq 0.2$.



มหาวิทยาลัยเทคโนโลยีสุรนารี

Figure 5.4 One-dimensional curve for the famous MQ with different shape values under this investigation.

5.2 Numerical Experiments and General Discussion

In this section, two well-known benchmarking test cases are numerically experimented using all the RBF forms studied in this work. Nevertheless, with the limitation of the space, only those significant and relevant results are being illustrated and discussed.

5.2.1 Experiment 1: Franke's function

This first experiment is concerned with one of the most well-known testing functions invented by Franke (Franke, 1982) , and to be referred to as 'Franke's function' hereafter. The expression of the function is as follows:

$$\begin{aligned} z = f(x,y) &= 0.75\exp\left(-\frac{(9x-2)^2+(9y-2)^2}{4}\right)+0.75\exp\left(-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}\right) \\ &\quad +0.5\exp\left(-\frac{(9x-7)^2+(9y-3)^2}{4}\right)-0.2\exp(-(9x-4)^2-(9y-7)^2). \end{aligned} \tag{5.1}$$

The results validation for all RBFs for this first case is to be numerically investigated at two different noise levels (SNR); $SNR=15$ and 30, as illustrated in Figure 5.5, Figure 5.6, Figure 5.9, and Figure 5.10.

As can be seen in Figure 5.5, Figure 5.6, Figure 5.9, and Figure 5.10 that the accuracy, measured by both L_∞ and L_{RMS} from both cases of node distribution manners, slightly decreases with the increase of noise variance. This is not surprising as the pattern becomes more complicated with data being more disturbed with noise.

In terms of the number of centres (m) generated by the RC-algorithm, at $SNR=30$ and with comparatively low values of L_∞, L_{RMS}, and condition numbers, it is found that $m=87$, $m=100$ and $m=83$ are reproduced for WU3, WL5, and BUH2, respectively. This figure can be seen as a good success of these RBFs, taking 2,500 original into consideration, while good accuracy is still achievable (see Table 5.2 and Table 5.4). Table 5.3 also shows the similar figure of these RBFs $SNR=30$ confirming their positive potential for further uses. The ability to preserve the data's pattern while using small number of nodes is desirable for any kind of RBF in practical applications. Small number in m also strongly affects both CPU-time and storage required for each computation process as clearly seen in Figure 5.7 and Figure 5.8 for $SNR=30$ and Figure 5.11 and Figure 5.12 for $SNR=15$.

5.2.1.1 Performance at $SNR=30$

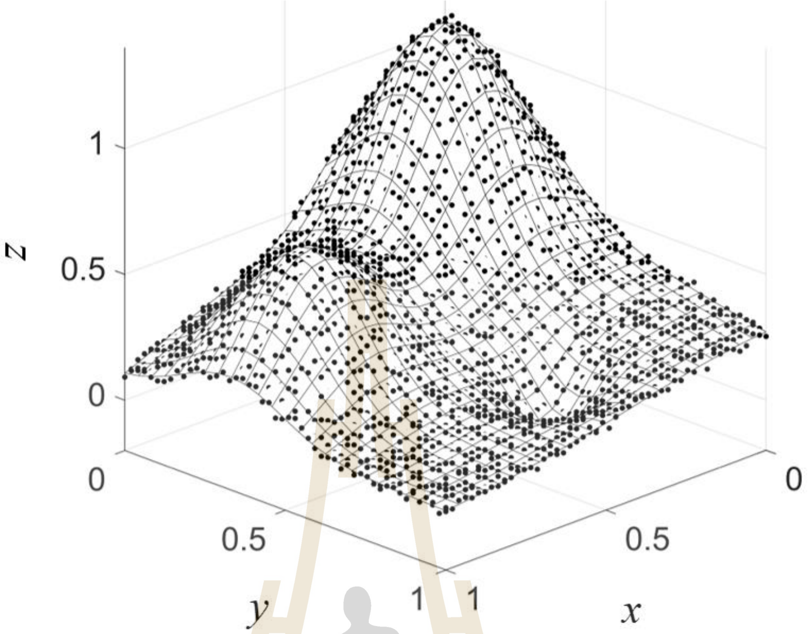


Figure 5.5 Noised-$z(x,y)$ training datasets with $SNR=30$ based on uniformly distribution of $\{(x,y)\}$.

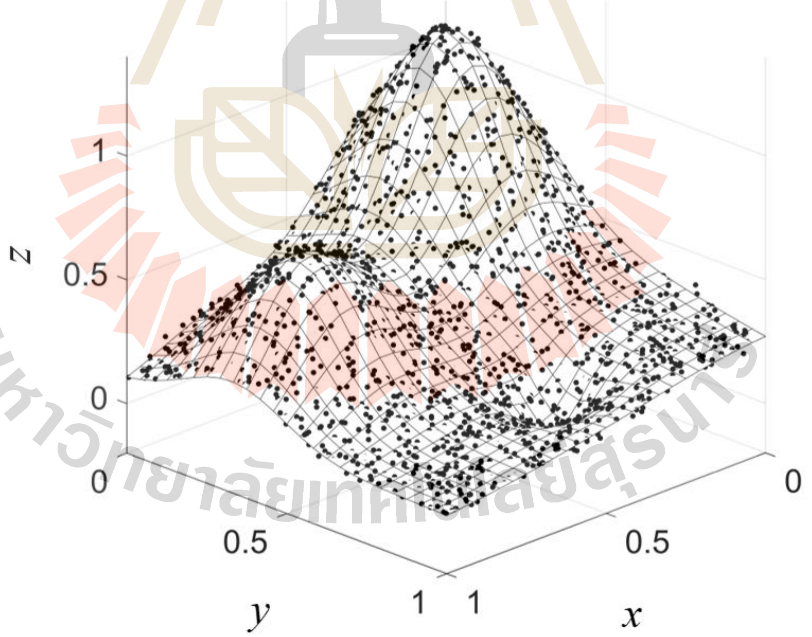


Figure 5.6 Noised-$z(x,y)$ training datasets with $SNR=30$ based on non-uniformly (randomly) distribution of $\{(x,y)\}$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.2 Training and testing errors measured for Franke's function at $SNR = 30$.

Types of RBF	Uniform nodes distribution				Random nodes distribution			
	L_∞		L_{RMSE}		L_∞		L_{RMSE}	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MQ+ Carlson	1.77E-01	1.55E-01	3.57E-02	3.18E-02	1.99E-01	2.19E-01	3.78E-02	3.58E-02
PS	5.50E-02	7.60E-01	1.15E-02	8.39E-02	5.75E-02	2.60E+00	1.24E-02	1.20E-01
TPS	5.48E-02	3.54E-02	1.51E-02	6.50E-03	5.73E-02	1.52E-01	1.49E-02	7.60E-03
WU1	1.07E-01	8.19E-02	2.04E-02	1.33E-02	9.10E-02	8.66E-02	2.08E-02	1.45E-02
WU2	8.03E-02	5.80E-02	1.84E-02	1.00E-02	6.03E-02	5.00E-02	1.72E-02	8.20E-03
WU3	5.80E-02	2.58E-02	1.57E-02	3.90E-03	5.75E-02	2.26E-02	1.54E-02	3.80E-03
WU4	5.65E-02	3.07E-02	1.34E-02	7.10E-03	4.63E-02	3.08E-02	1.31E-02	7.10E-03
WL1	6.26E-02	5.13E-02	1.53E-02	6.00E-03	6.72E-02	7.88E-02	1.53E-02	6.60E-03
WL2	6.19E-02	3.72E-02	1.65E-02	6.20E-03	5.84E-02	3.09E-02	1.58E-02	4.90E-03
WL3	6.84E-02	4.64E-02	1.70E-02	7.50E-03	5.92E-02	3.61E-02	1.62E-02	5.40E-03
WL4	5.93E-02	2.98E-02	1.38E-02	6.70E-03	5.70E-02	3.90E-02	1.37E-02	6.30E-03
WL5	5.64E-02	1.76E-02	1.55E-02	3.70E-03	5.74E-02	2.01E-02	1.54E-02	3.90E-03
WL6	5.83E-02	2.45E-02	1.59E-02	4.40E-03	6.09E-02	3.71E-02	1.60E-02	5.70E-03
WL7	6.14E-02	3.72E-02	1.63E-02	5.80E-03	6.12E-02	3.74E-02	1.60E-02	5.60E-03
BUH1	5.74E-02	2.00E-02	1.55E-02	4.00E-03	5.56E-02	2.78E-02	1.52E-02	3.70E-03
BUH2	5.66E-02	1.95E-02	1.56E-02	3.60E-03	5.74E-02	2.39E-02	1.54E-02	3.50E-03
BUH3	5.68E-02	1.45E-02	1.51E-02	4.10E-03	5.63E-02	1.80E-02	1.50E-02	4.30E-03

มหาวิทยาลัยเทคโนโลยีสุรนารี

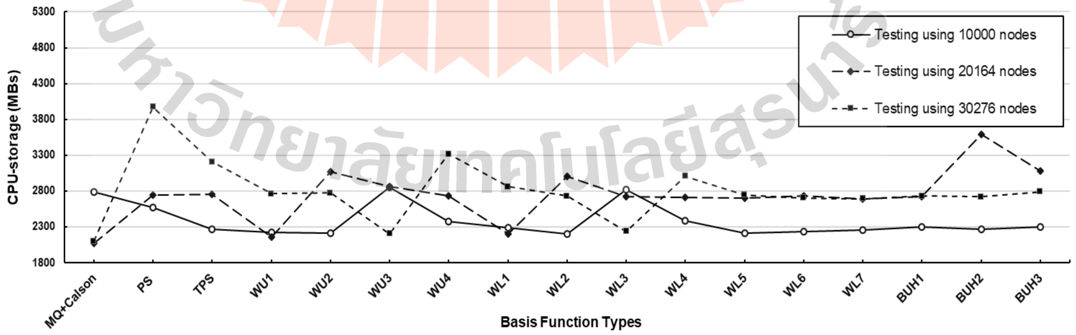


Figure 5.7 CPU-storage measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR = 30$.

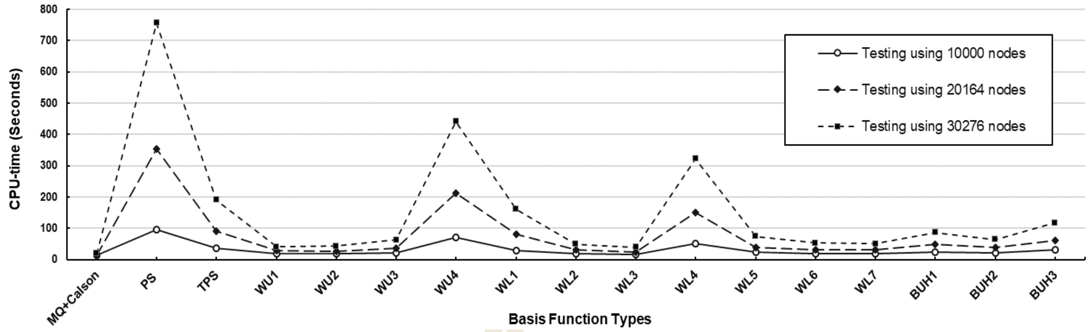


Figure 5.8 CPU-time measurement observed at three different sizes of testing datasets as uniform nodes distribution using $SNR = 30$.

5.2.1.2 Performance at $SNR = 15$

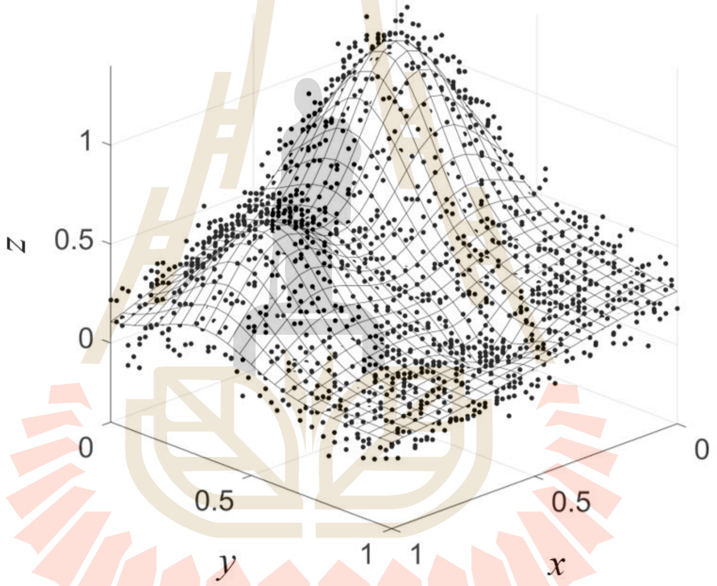


Figure 5.9 Noised- $z(x, y)$ training datasets with $SNR = 15$ based on uniformly distribution of $\{(x, y)\}$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

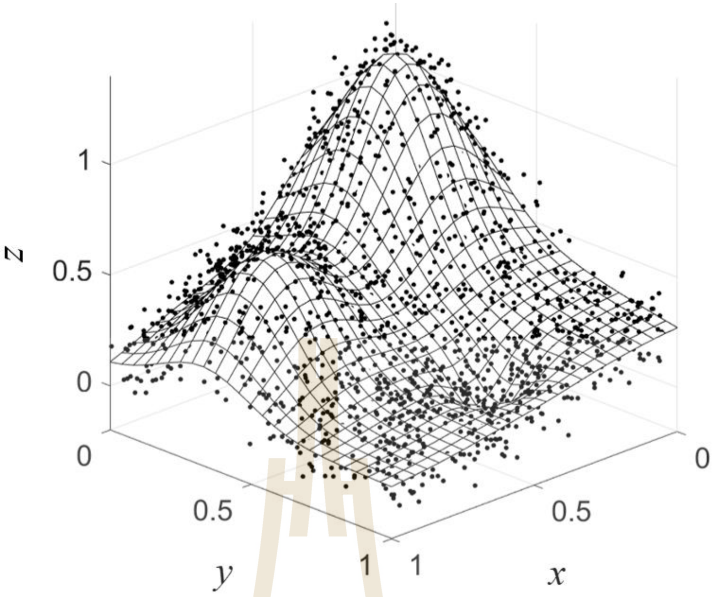


Figure 5.10 Noised-$z(x, y)$ training datasets with $SNR = 15$ based on non-uniformly (randomly) distribution of $\{(x, y)\}$.

Table 5.3 Training and testing errors measured for Franke's function at $SNR = 15$.

Types of RBF	Uniform nodes distribution				Random nodes distribution			
	L_∞		L_{RMSE}		L_∞		L_{RMSE}	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MQ+Carlson	3.50E-01	2.55E-01	9.84E-02	3.88E-02	4.02E-01	2.02E-01	9.30E-02	3.75E-02
PS	2.90E-01	4.27E+00	6.36E-02	3.28E-01	3.02E-01	3.92E+00	6.55E-02	3.34E-01
TPS	2.76E-01	1.56E-01	8.40E-02	3.18E-02	2.90E-01	1.92E-01	8.15E-02	3.19E-02
WU1	3.56E-01	8.46E-02	8.97E-02	1.76E-02	3.09E-01	1.06E-01	8.63E-02	1.91E-02
WU2	3.39E-01	6.11E-02	8.93E-02	1.56E-02	3.00E-01	7.43E-02	8.54E-02	1.55E-02
WU3	3.26E-01	6.36E-02	8.82E-02	1.81E-02	2.75E-01	6.31E-02	8.44E-02	1.82E-02
WU4	3.36E-01	2.02E-01	7.60E-02	4.27E-02	2.62E-01	1.92E-01	7.35E-02	3.82E-02
WL1	3.17E-01	1.03E-01	8.51E-02	2.91E-02	3.02E-01	1.11E-01	8.22E-02	2.53E-02
WL2	3.30E-01	6.13E-02	8.89E-02	1.66E-02	2.99E-01	6.40E-02	8.51E-02	1.57E-02
WL3	3.44E-01	5.93E-02	8.93E-02	1.49E-02	2.91E-01	6.91E-02	8.52E-02	1.52E-02
WL4	3.52E-01	1.68E-01	7.94E-02	3.87E-02	2.70E-01	1.61E-01	7.66E-02	3.42E-02
WL5	3.22E-01	6.49E-02	8.81E-02	1.86E-02	2.72E-01	5.87E-02	8.44E-02	1.83E-02
WL6	3.32E-01	7.15E-02	8.87E-02	1.64E-02	2.77E-01	7.17E-02	8.49E-02	1.73E-02
WL7	3.29E-01	7.20E-02	8.89E-02	1.63E-02	2.83E-01	7.05E-02	8.50E-02	1.68E-02
BUH1	3.25E-01	7.75E-02	8.77E-02	2.04E-02	2.77E-01	8.06E-02	8.39E-02	2.06E-02
BUH2	3.20E-01	8.29E-02	8.83E-02	1.77E-02	2.73E-01	5.88E-02	8.46E-02	1.73E-02
BUH3	3.20E-01	1.03E-01	8.68E-02	2.34E-02	2.85E-01	9.31E-02	8.34E-02	2.25E-02

มหาวิทยาลัยเทคโนโลยีสุรนารี

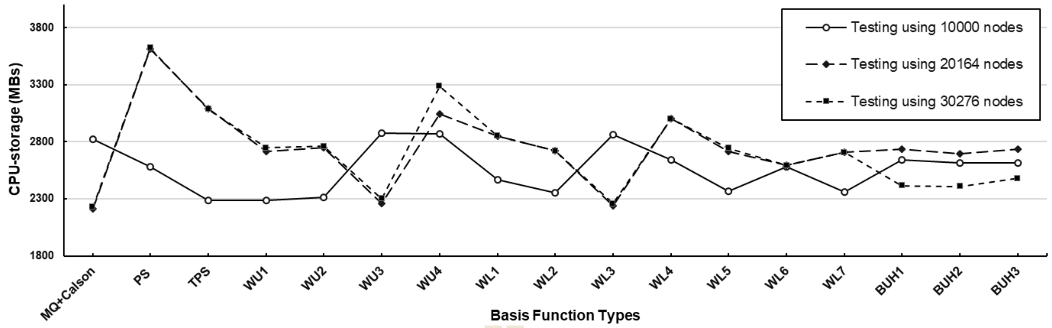


Figure 5.11 CPU-storage measurement observed at three different sizes of testing datasets with random nodes distribution using $SNR = 15$.

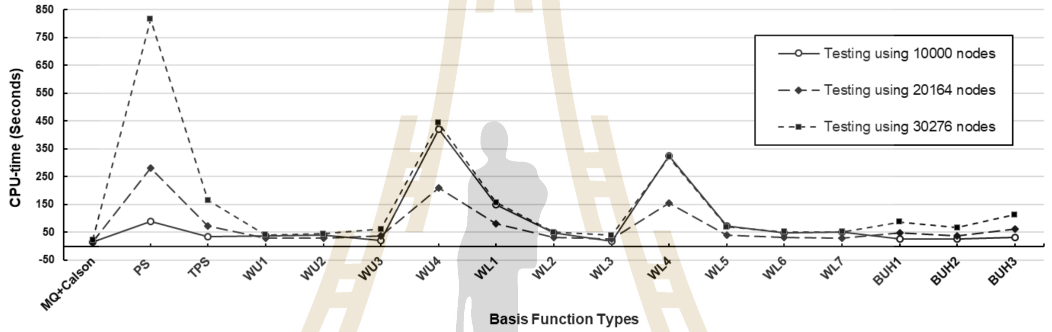


Figure 5.12 CPU-time measurement observed at three different sizes of testing datasets with random nodes distribution using $SNR = 15$.

Table 5.4 The number of centres (m) and the condition number $(Cond_{\infty}(\cdot))$ of all radial basis function candidates for the Franke's function test case.

Types of RBF	$SNR = 30$				$SNR = 15$			
	m		$Cond_{\infty}(\cdot)$		m		$Cond_{\infty}(\cdot)$	
	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.
MQ+Carlson	19	19	5.41E+03	6.44E+03	19	19	5.84E+03	6.02E+03
PS	1239	996	2.92E+04	2.39E+04	1239	996	2.92E+04	2.39E+04
TPS	295	243	1.40E+04	1.32E+04	295	243	1.40E+04	1.32E+04
WU1	41	39	8.26E+03	6.62E+03	41	39	8.26E+03	6.62E+03
WU2	47	47	6.99E+03	9.16E+03	47	47	6.99E+03	9.16E+03
WU3	87	83	9.01E+03	9.35E+03	87	83	9.01E+03	9.35E+03
WU4	689	662	9.29E+03	8.32E+03	689	662	9.29E+03	8.32E+03
WL1	245	234	9.62E+03	9.77E+03	245	234	9.62E+03	9.77E+03
WL2	64	62	9.28E+03	8.53E+03	64	62	9.28E+03	8.53E+03
WL3	51	50	7.02E+03	7.86E+03	51	50	7.02E+03	7.86E+03

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.4 (continued).

Types of RBF	$SNR=30$				$SNR=15$			
	m		$Cond_\infty(\cdot)$		m		$Cond_\infty(\cdot)$	
	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.
WL4	511	504	9.25E+03	7.95E+03	511	504	9.25E+03	7.95E+03
WL5	100	94	1.23E+04	1.01E+04	100	94	1.23E+04	1.01E+04
WL6	69	65	1.07E+04	8.81E+03	69	65	1.07E+04	8.81E+03
WL7	64	62	8.71E+03	8.57E+03	64	62	8.71E+03	8.57E+03
BUH1	117	113	1.13E+04	1.01E+04	117	113	1.13E+04	1.01E+04
BUH2	83	78	1.22E+04	9.22E+03	83	78	1.22E+04	9.22E+03
BUH3	164	154	1.10E+04	1.24E+04	164	154	1.10E+04	1.24E+04

5.2.2 Experiment 2: F7

For the second test case, we study one of the functions called F7 in the investigated nicely carried out by R. J. Renka and R. Brown (Renka and Brown, 1999), (and shall be referred to as '**F7**' in this work as well). The function is defined as follows.

$$z = f(x,y) = 2\cos(10x)\sin(10y) + \sin(10xy). \tag{5.2}$$

Similarly to the first experiment, two different noise levels (SNR); $SNR=15$ and 30 are under the investigation with their nodes distributions are illustrated in Figure 5.13, Figure 5.14, Figure 5.17 and Figure 5.18, respectively.

With the results obtained in these cases, shown in Table 5.5, Table 5.6, and Table 5.7, together with information provided in Figure 5.15 and Figure 5.16, it has been observed that the overall trends in accuracy (both L_∞ and L_{RMS}) are similar to those discovered in the first experiment (with F1) where some smaller errors are noticed in the cases of WL6 and WL7. Moreover, the similarity to the first experiment in terms of the number of centres (m) produced by each shapeless RBF after undergoing the RC-algorithm is also noticeable here. This is due to the fact that with no parameter involved this values are dependent only the distance between two nodes. In the case of MQ, nevertheless, with different values of shape parameter generated by Carlson algorithm, the change in the number of centres (m) shall be anticipated.

5.2.2.1 Performance at $SNR = 30$

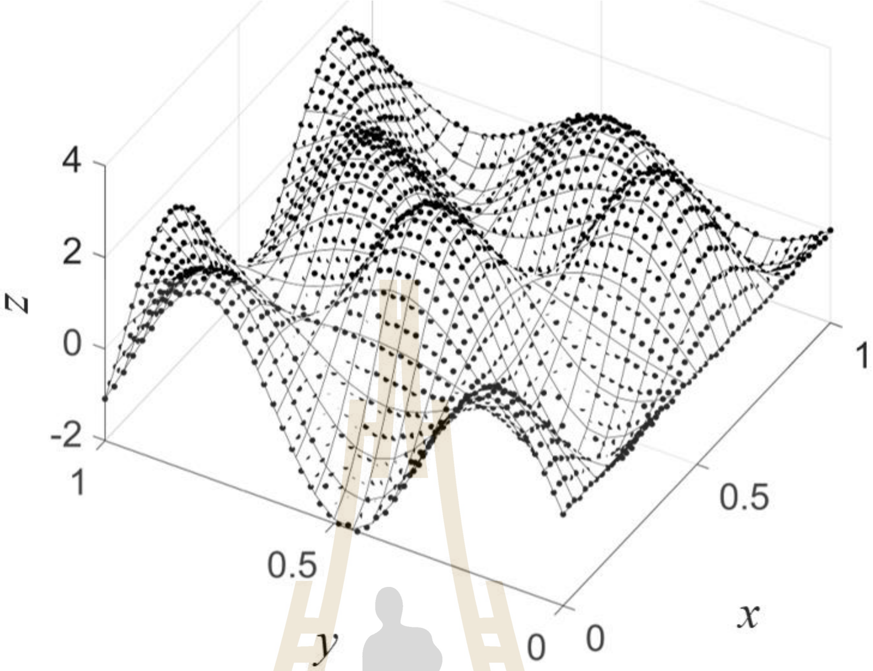


Figure 5.13 Noised- $z(x, y)$ training datasets based on uniformly distribution of $\{(x, y)\}$.

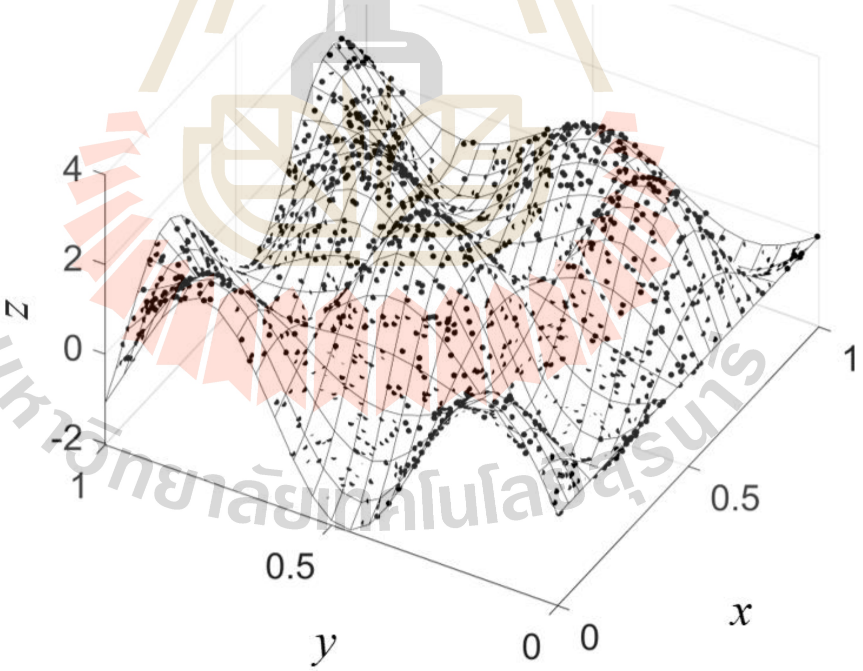


Figure 5.14 Noised- $z(x, y)$ training datasets based on non-uniformly (randomly) distribution of $\{(x, y)\}$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.5 Training and testing errors measured for F7 at $SNR = 30$.

Types of RBF	Uniform nodes distribution				Random nodes distribution			
	L_∞		L_{RMSE}		L_∞		L_{RMSE}	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MQ+Carlson	4.93E-01	4.38E-01	1.03E-01	9.38E-02	5.84E-01	6.33E-01	1.06E-01	1.10E-01
PS	2.90E-01	1.26E+01	6.65E-02	1.15E+00	2.33E-01	2.28E+01	4.78E-02	1.18E+00
TPS	3.88E-01	4.48E-01	6.88E-02	5.98E-02	4.55E-01	1.14E+00	6.24E-02	8.27E-02
WU1	7.16E-01	6.96E-01	1.49E-01	1.40E-01	1.12E+00	1.78E+00	1.71E-01	1.83E-01
WU2	4.84E-01	4.74E-01	9.45E-02	8.42E-02	5.00E-01	5.72E-01	8.64E-02	8.66E-02
WU3	1.93E-01	1.94E-01	4.65E-02	3.06E-02	2.14E-01	2.72E-01	4.46E-02	3.50E-02
WU4	1.14E-01	7.67E-02	3.03E-02	1.63E-02	1.39E-01	7.50E-01	2.94E-02	2.67E-02
WL1	1.08E+00	1.08E+00	7.74E-02	6.27E-02	8.93E-01	2.80E+00	7.54E-02	9.60E-02
WL2	3.99E-01	4.03E-01	7.21E-02	6.05E-02	6.38E-01	7.21E-01	7.25E-02	7.09E-02
WL3	4.92E-01	4.77E-01	9.19E-02	8.18E-02	2.31E-01	3.27E-01	5.80E-02	5.27E-02
WL4	1.29E-01	7.42E-02	3.20E-02	1.54E-02	1.69E-01	8.44E-01	3.20E-02	3.03E-02
WL5	1.86E-01	1.43E-01	3.94E-02	2.08E-02	1.98E-01	2.62E-01	4.20E-02	3.20E-02
WL6	2.00E-01	2.17E-01	4.17E-02	2.45E-02	2.18E-01	2.22E-01	4.33E-02	3.14E-02
WL7	1.89E-01	1.72E-01	4.41E-02	2.68E-02	1.95E-01	2.13E-01	4.23E-02	2.99E-02
BUH1	1.27E-01	8.67E-02	3.62E-02	1.60E-02	1.89E-01	5.10E-01	3.85E-02	2.76E-02
BUH2	1.66E-01	1.36E-01	4.10E-02	2.36E-02	2.17E-01	2.37E-01	4.55E-02	3.57E-02
BUH3	1.47E-01	8.09E-02	3.45E-02	1.30E-02	1.23E-01	4.18E-01	3.47E-02	1.88E-02

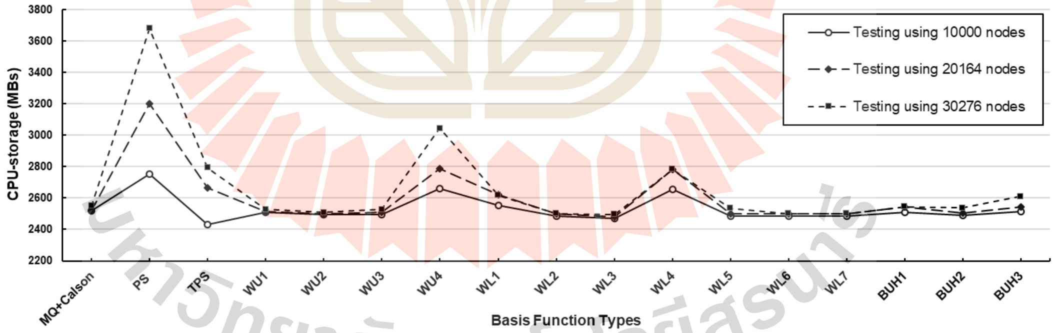


Figure 5.15 CPU-storage measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR = 30$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

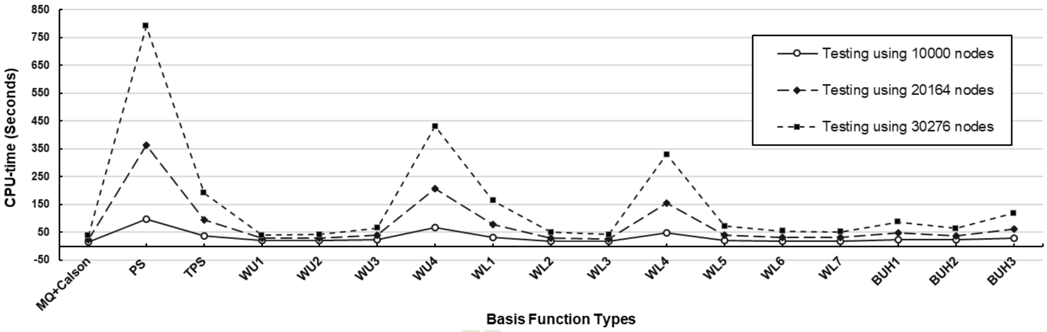


Figure 5.16 CPU-time measurement observed at three different sizes of testing datasets with uniform nodes distribution using $SNR = 30$.

5.2.2.2 Performance at $SNR = 15$

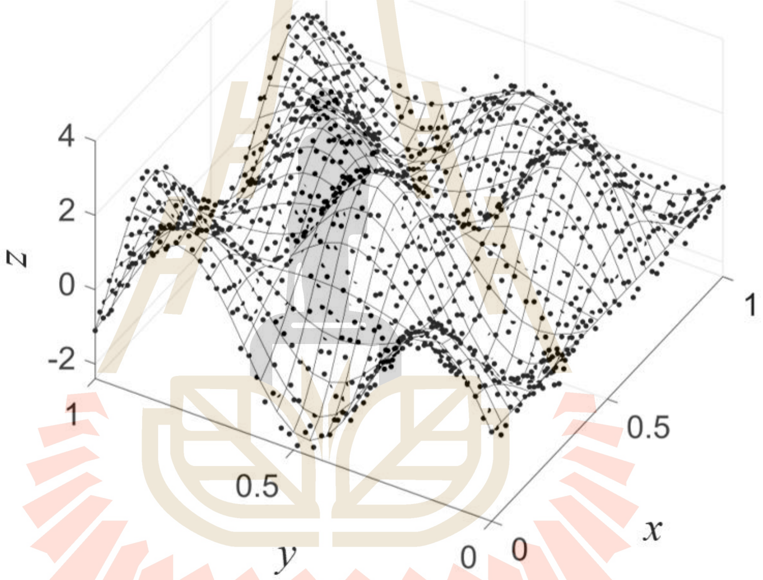


Figure 5.17 Noised-$z(x, y)$ training datasets based on uniformly distribution of $\{(x, y)\}$.

มหาวิทยาลัยเทคโนโลยีสุรนารี

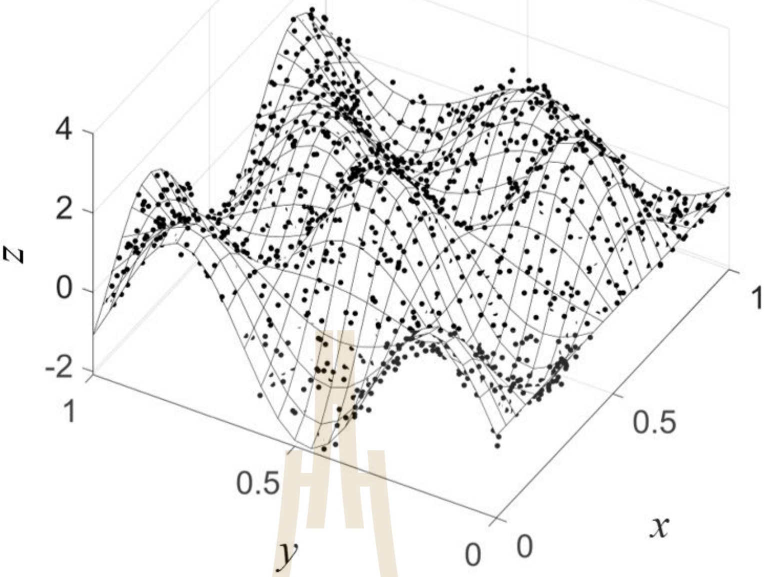


Figure 5.18 Noised- $z(x, y)$ training datasets based non-uniformly (randomly) distribution of $\{(x, y)\}$.

Table 5.6 Training and testing errors measured for F7 at $SNR = 15$.

Types of RBF	Uniform nodes distribution				Random nodes distribution			
	L_∞		L_{RMSE}		L_∞		L_{RMSE}	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
MQ+Carlson	1.12E+00	7.32E-01	2.37E-01	1.38E-01	9.91E-01	7.55E-01	2.15E-01	1.16E-01
PS	6.88E-01	1.58E+01	1.52E-01	1.35E+00	6.82E-01	2.56E+01	1.50E-01	1.35E+00
TPS	7.51E-01	4.71E-01	1.92E-01	8.70E-02	7.27E-01	1.28E+00	1.88E-01	1.10E-01
WU 1	9.39E-01	7.39E-01	2.40E-01	1.41E-01	1.26E+00	1.96E+00	2.54E-01	1.86E-01
WU 2	8.34E-01	5.10E-01	2.10E-01	8.69E-02	9.69E-01	5.20E-01	2.05E-01	8.92E-02
WU 3	7.31E-01	1.91E-01	1.93E-01	4.60E-02	7.85E-01	2.83E-01	1.90E-01	4.88E-02
WU 4	7.00E-01	4.72E-01	1.65E-01	8.91E-02	6.37E-01	8.65E-01	1.64E-01	8.95E-02
WL 1	1.15E+00	1.13E+00	1.96E-01	8.31E-02	8.27E-01	3.24E+00	1.93E-01	1.15E-01
WL 2	7.43E-01	3.51E-01	2.01E-01	6.72E-02	9.69E-01	6.64E-01	2.02E-01	7.77E-02
WL 3	8.41E-01	4.51E-01	2.10E-01	8.38E-02	9.16E-01	3.31E-01	1.94E-01	5.86E-02
WL 4	7.23E-01	3.70E-01	1.72E-01	8.04E-02	6.43E-01	8.41E-01	1.71E-01	7.98E-02
WL 5	7.08E-01	1.51E-01	1.90E-01	4.33E-02	8.01E-01	2.79E-01	1.89E-01	5.04E-02
WL 6	7.05E-01	1.57E-01	1.92E-01	3.88E-02	7.92E-01	2.69E-01	1.90E-01	4.36E-02
WL 7	7.05E-01	1.89E-01	1.92E-01	3.85E-02	7.88E-01	2.06E-01	1.90E-01	4.24E-02
BUH 1	7.05E-01	1.55E-01	1.89E-01	4.40E-02	7.40E-01	4.54E-01	1.87E-01	4.98E-02
BUH 2	7.14E-01	1.67E-01	1.91E-01	4.16E-02	8.34E-01	2.82E-01	1.91E-01	4.96E-02
BUH 3	7.20E-01	2.23E-01	1.87E-01	5.05E-02	7.82E-01	3.79E-01	1.84E-01	5.21E-02

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.7 The number of centres (m) and the condition number $(Cond_\infty(\cdot))$ of all radial basis function candidates for the F7 test case.

Types of RBF	$SNR = 30$				$SNR = 15$			
	m		$Cond_\infty(\cdot)$		m		$Cond_\infty(\cdot)$	
	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.	Ufm.	Rdm.
MQ+Carlson	49	47	9.93E+03	1.20E+04	43	46	1.06E+04	8.56E+03
PS	1239	996	2.92E+04	2.39E+04	1239	996	2.92E+04	2.39E+04
TPS	295	243	1.40E+04	1.32E+04	295	243	1.40E+04	1.32E+04
WU1	41	39	8.26E+03	6.62E+03	41	39	8.26E+03	6.62E+03
WU2	47	47	6.99E+03	9.16E+03	47	47	6.99E+03	9.16E+03
WU3	87	83	9.01E+03	9.35E+03	87	83	9.01E+03	9.35E+03
WU4	689	662	9.29E+03	8.32E+03	689	662	9.29E+03	8.32E+03
WL1	245	234	9.62E+03	9.77E+03	245	234	9.62E+03	9.77E+03
WL2	64	62	9.28E+03	8.53E+03	64	62	9.28E+03	8.53E+03
WL3	51	50	7.02E+03	7.86E+03	51	50	7.02E+03	7.86E+03
WL4	511	504	9.25E+03	7.95E+03	511	504	9.25E+03	7.95E+03
WL5	100	94	1.23E+04	1.01E+04	100	94	1.23E+04	1.01E+04
WL6	69	65	1.07E+04	8.81E+03	69	65	1.07E+04	8.81E+03
WL7	64	62	8.71E+03	8.57E+03	64	62	8.71E+03	8.57E+03
BUH1	117	113	1.13E+04	1.01E+04	117	113	1.13E+04	1.01E+04
BUH2	83	78	1.22E+04	9.22E+03	83	78	1.22E+04	9.22E+03
BUH3	164	154	1.10E+04	1.24E+04	164	154	1.10E+04	1.24E+04

From all the numerical results obtained so far and in addition to the criteria stated, it has to be acknowledged that 'overfitting and overfitting' is another crucial figure to be considered. Regarding all cases, a significant reduction in accuracy, measured by both error norms, produced by PS type of RBF strongly indicates its overfitting nature and shall not be recommended for practical uses. On the other hand, WL5, BUH1, WU3, and BUH2 are seen to be slightly under fitting for both numerical demonstrations. The best types of RBF in terms of this aspect are WU1, WU2, WL1, WL4, and MQ.

มหาวิทยาลัยเทคโนโลยีสุรนารี

5.3 Main Conclusions

This work aims to provide insights of the use of sixteen forms of radial basis function (RBF) containing no shape parameter, so they are referred to as 'shapeless RBFs'. The challenge under the main experiment is the problem of pattern recognition through neural networks computation process and architecture. The ability to deal with large and noised datasets of each shapeless RBF is measured under several criteria against the well-known shape-containing RBF called multiquadric (MQ). Two testing functions are tackled and important findings observed based on each criterion are listed below.

1. Accuracy (L_∞ and L_{RMSE}): The best three shapeless RBFs are WL6, WL7, and BUH3, whereas PS, WU1, and MQ are found to produce the least accurate results.

2. The condition number ($Cond_\infty(\cdot)$): The best three are WU1, WU2, and WL3 with the comparatively worse ones being PS, TPS, and WL5.

3. In terms of CPU (time and storage) and also the number of centres (m): MQ, WL3 and WL6 are seen to be the most desirable whereas PS, WU4, and WL4 are not so promising under these criteria.

4. User's Interference: It is obvious that as long as no parameter-turning process is required, there is no need to interfere the algorithm and this is one desirable aspect of using shapeless RBF. This is also the case for the sensitivity-to-parameter criteria.

5. Ease of implementation: It is observed that all shapeless RBF types under this work are as equally simple when it comes to implementing the scheme. For MQ, on the other hand, an additional coding routine may be needed for the process of a reliable shape searching procedure, as is the case in this investigation for Carlson algorithm.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Together with the 'overfitting and underfitting' aspect, this work suggests that shapeless RBFs from Wu's and Wendland's families are highly promising, whereas the rest forms are still skeptical for practical uses. Apart from this useful piece of information for pattern recognition application, it might be considered a weakness of the work that the figure discovered so far may change when dealing with other kinds of applications such as direct interpolation, function approximation, and recovery, as well as solving different equations (DEs) under the concepts of meshless or meshfree methods. Furthermore, other kinds of node distribution of sizes may result in different aspects. This is all set as our future research direction and is highly recommended for interested researchers to further explore.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER VI

CONCLUSIONS

In this study, the main attention is paid to numerically investigate the overall performances of newly-proposed radial basis functions (RBFs). These RBFs have a special feature where they contain no shape parameter so they are referred to as 'shapeless-RBF'. A structure of neural network under the application of pattern recognition is utilized and a total of sixteen forms of shapeless RBFs have been paid attention to. Before reaching the point where all these RBFs were investigated, there had been five numerical experiments carried out in three main different contexts; pattern recognition, function recovery, and PDEs solution approximation.

Together with all the preliminary results previously obtained, the main and important findings discovered in this work can be listed as follows:

1. As long as a shape-contained RBF is in use, the most challenging task is still the search for an optimal value of the shape, no matter what context it is applied under.

2. Different forms of RBF require different values of shape meaning that there is no such thing as 'optimal shape' for all RBFs.

3. Taking into consideration all the performance criteria stated in this work, shapeless RBFs have been proved to comparatively be promising whereas the pain of searching for a good shape has been eliminated.

4. Amongst the sixteen forms of shapeless RBFs under this investigation, the results have shown that those in Wu's and Wendland's families can well be good alternatives.

It is also recommended in this work that the experiment is extended to more complex problems with higher dimension and/or node distribution manners. Moreover, other applications such as function recovery or solving differential equations are also of interest.

มหาวิทยาลัยเทคโนโลยีสุรนารี

REFERENCES

มหาวิทยาลัยเทคโนโลยีสุรนารี

Radial basis function artificial neural network for the investigation of Thyroid Cytological Lesions, *Journal of Tyroid Research*, *2020*.

Franke, R. (1979). A critical comparison of some methods for interpolation of scattered data, Naval Postgraduate School, Technical Report NPS-53-79-003.

Franke, R. (1982) Scattered Data Interpolation: Tests of Some Method, *Mathematics of Computation, 38*(157), 181–200. doi.org/10.2307/2007474

Hardy, R. L. (1971). Multiquadric equations of topography and other irregular surfaces, *Journal of Geophysical Research, 76*(8), 1905-1915. doi.org/10.1029/JB076i008 p01905

Hardy, R. L. (1977) Least Squares Prediction Photogrammetric, *Engineering and Remote Sensing, 43*(4), 475-492.

Hemageetha, N., and Nasir, G. M. (2013). Radial basis function model for vegetable price prediction, Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, India. 424-428. doi: 10.1109/ICPRIME.2013.6496514

Kaennakham, S., and Chuathong, N. (2017) Solution to a Convection-Diffusion Problem Using a New Variable Inverse-Multiquadric Parameter in a Collocation Meshfree Scheme, *International Journal of Multiphysics, 4*, 359-374, doi.org/10.21152/ 1750-9548.11.4.359

Lazzaro, D., and Montefusco, L. B. (2002) Radial basis functions for the multivariate interpolation of large scattered data sets, *Journal of Computational and Applied Mathematics, 140*, 521–536.

Nojavan, H., Abbasbandy, S., and Allahviranloo, T. (2017) Variable shape parameter strategy in local radial basis functions collocation method for solving the 2D nonlinear coupled Burgers' equations. *Mathematics, 5*(3), 1-21. doi.org/10.3390/math5030038.

Paolanti, M., and Frontoni, E. (2020) Multidisciplinary pattern recognition applications: a review, *Emanuele Frontoni, 37*. doi: 10.1016/j.cosrev.2020.100276

Parasher, M., Sharma, S., Sharma, A.K., and Gupta, J.P. (2011) Anatomy on pattern recognition, *Indian Journal of Computer Science and Engineering (IJCSE), 2*(3), 371-378.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: a review, in Algorithms for approximation, Clarendon Press, United States, 143–167.

Rashedi, K. A., Ismail, M. T., Hamadneh, N. N., Wadi, S. AL, Jaber J. J. and Tahir, M. (2021). Application of radial basis function neural network coupling particle swarm optimization algorithm to classification of Saudi Arabia stock returns, *Journal of Mathematics, 5,* 1-8. doi: https://doi.org/10.1155/2021/5593705

Renka, R. J., and Brown, R. (1999) Algorithm 792: accuracy tests of ACM algorithms for interpolation of scattered data in the plane, ACM Transactions on Mathematical Software, 25(1), 78–94. doi.org/10.1145/305658.305745.

Rippa, S. (1999) An algorithm for selecting a good value for the parameter c in radial basis function interpolation. *Advances in Computational Mathematics, 11*, 193–210. doi.org/10.1023/A:1018975909870

Shastry, K. A., Sanjay, H. A., and Deexith, G. (2017). Quadratic-radial-basis-function-kernel for classifying multi-class agricultural datasets with continuous attributes, *Applied Soft Computing Journal, 58*, 65-74. https://doi.org/10.1016/j.asoc.2017.04.049

Shin, M. (1998) Design and evaluation of radial basis function model for function approximation. Syracuse University, Syracuse, NY, United States.

Shin, M., and Park C. (2000) A radial basis function approach to pattern recognition and its applications, *ETRI Journal, 22*(2).

Tavaen, S., Viriyapong, R., and Kaennakham, S. (2020) Performances of non-parameterised radial basis functions in pattern recognition applications, *Journal of Physics: Conference Series, 1706*(1):012165, doi: 10.1088/1742-6596/1706/1/012165

Winston, P. H. (1993). Artificial Intelligence. Addison-Wesley Publishing Company.

Wang, R., Li, D., and Miao, K. (2020). Optimized radial basis function neural network based intelligent control algorithm of unmanned surface vehicles. *Journal of*

มหาวิทยาลัยเทคโนโลยีสุรนารี

Marine Science and Engineering, 8(3). doi: 10.3390/jmse8030210

Wendland, H. (1995) Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, *Advances in Computational Mathematics, 4*, 389-396. doi: https://doi.org/10.1007/BF02123482.

Wu, Z. (1995) Compactly supported positive definite radial basis functions, *Advances in Computational Mathematics, 4*, 283-292.

Xiang, S., Wang, K., Ai, Y., Sha, Y., and Shi, H. (2012) Trigonometric variable shape parameter and exponent strategy for generalized multiquadric radial basis function approximation. *Applied Mathematical Modelling, 36*(5), 1931-1931

มหาวิทยาลัยเทคโนโลยีสุรนารี

APPENDICES

มหาวิทยาลัยเทคโนโลยีสุรนารี

APPENDIX A

MAIN CODES

มหาวิทยาลัยเทคโนโลยีสุรนารี

A.1 Main Code 1: Experiment 2

A.1.1 Test Case 1: Linear interpolation

A.1.1.1 Training and validation errors for candidate models for linear interpolation case.

```
clc
clear all
close all

%******************************************************************************************
%--------------- Step 1: Generate training dataset -------------------------------------------
%******************************************************************************************
for i=1:100
X_trn(i)=2*pi*((i-1)/100);
end
Y_trn =awgn(2*X_trn +1,15,'measured')';% Y_trn with Gaussian noise
for i=1:100
for j=1:100
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end
XY_trn;

N_tr = length(X_trn);

%******************************************************************************************
%--------------- Step 2: Normalizing both X_trn and Y_trn ------------------------------------
%******************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn -min(X_trn))./(x1max);                % Normalized X_trn
Y_trnN=(Y_trn -min(Y_trn))./(max(Y_trn)-min(Y_trn));   % Normalized Y_trn

%******************************************************************************************
%--------------- Step 3: Choosing the shape parameter----------------------------------------
%******************************************************************************************
shp = input('Enter the shape parameter = ');

%******************************************************************************************
%--------------- Step 4: Construct the interpolation matrix-----------------------------------
%--------------- CS_RBF1, CS1, CS2, G -------------------------------------------------------
%******************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)- X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
end

%***************************************************************************************************************
%--------------- Step 5:Using SVD to get the number of centres-------------------------------
%----------------------- out of matrix CS_RBF1, CS1, CS2, G ---------------------------------------------
%***************************************************************************************************************
delta = 0.01/100;

%--------------- Step 5.1:CS_RBF1 -----------------------------------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
i = i+1 ;
end

%--------------- Step 5.2:CS1 -----------------------------------------------------------------------------------------
[U_CS1,S_CS1,V_CS1] = svd(CS1);
S11_CS1 = S_CS1(1,1);
% The firsts singular of singular value
S1_CS1 = S11_CS1*(delta);
for i=1:N_tr
Sii_CS1 = S_CS1(i,i);
if Sii_CS1 <= S1_CS1
M_CS1 = i-1;
break
end
i = i+1 ;
end

%--------------- Step 5.3:CS2 -----------------------------------------------------------------------------------------
[U_CS2,S_CS2,V_CS2] = svd(CS2);
S11_CS2 = S_CS2(1,1);
% The firsts singular of singular value
S1_CS2 = S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 = S_CS2(i,i);
if Sii_CS2 <= S1_CS2
M_CS2 = i-1;
break
end
i = i+1 ;
end

%--------------- Step 5.4:Gaussian RBF-----------------------------------------------------------------------------
[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1 ;
break
end
i = i+1 ;
end

%******************************************************************************************
%---------------- Step 6: Using QR to find the centres of--------------------------------------------
%------------------------ basis function and the location of centres ----------------------
%******************************************************************************************
%---------------- Step 6.1: CS_RBF1 -------------------------------------------------------------------
V1_CS_RBF1 = -V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);
V2_CS_RBF1 = [V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1] = qr(V2_CS_RBF1);
mu_CS_RBF1 = X_trnN*P_CS_RBF1;
Xctr_CS_RBF1 = mu_CS_RBF1(1,1:M_CS_RBF1);

%---------------- Step 6.2: CS1 -----------------------------------------------------------------------
V1_CS1 = -V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);
%partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 = [V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1] = qr(V2_CS1);
mu_CS1 = X_trnN*P_CS1;
Xctr_CS1 = mu_CS1(1,1:M_CS1);

%---------------- Step 6.3: CS2 -----------------------------------------------------------------------
V1_CS2 = -V_CS2;
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);
%partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2 = [V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2] = qr(V2_CS2);
mu_CS2 = X_trnN*P_CS2;
Xctr_CS2 = mu_CS2(1,1:M_CS2);

%---------------- Step 6.4: Gaussian RBF---------------------------------------------------------------
V1 = -V;
V11=V1(1:M,1:M);                %partition matrix V
V21=V1(M+1:N_tr,1:M);
V2 = [V11' V21'];
[Q,R,P] = qr(V2);
mu = X_trnN*P;
Xctr = mu(1,1:M)

%******************************************************************************************
%---------------- Step 7: Construct the interpolation matrix-------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%*************************************************************************************************
%--------------- Step 7.1: CS_RBF1 ----------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS_RBF1
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j)== 0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 7.2: CS1 --------------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS1
r1_CS1(i,j)=abs(X_trnN (1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 7.3: CS2 --------------------------------------------------------- ---------
for i=1:N_tr
for j=1:M_CS2
r1_CS2(i,j)=abs(X_trnN (1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j)== 0
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end
end

%--------------- Step 7.4: Gaussian RBF------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN (1,i)-Xctr(1,j));
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%*************************************************************************************************
%--------------- Step 8: Computing the weight 'w' -------------------------------------------------
%*************************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 =pinv(Phi_CS_RBF1)*Y_trnN;
w_CS1 =pinv(Phi_CS1)*Y_trnN;
w_CS2 =pinv(Phi_CS2)*Y_trnN;
w=pinv(Phi)*Y_trnN;

%*************************************************************************************************
%--------------- Step 9: Computing the approximation of -------------------------------------------
%--------------- training dataset------------------------------------------------------------------
%*************************************************************************************************
%--------------- Step 9.1: CS_RBF1 ----------------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 = YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

%--------------- Step 9.2: CS1 ---------------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 = YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.3: CS2 ---------------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 = YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.4: Gaussian RBF-------------------------------------------------------------
YappxN=Phi*w;
Yappx = YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%*****************************************************************************************************
%--------------- Step 10: Computing the training error with MSE---------------------------
%*****************************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn -Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn -Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn -Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn -Yappx).^2);

T_Trn = table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%*****************************************************************************************************
%--------------- Step 11: Generate validation dataset ------------------------------------------
%*****************************************************************************************************
%-------------//// Generate a validation dataset of 1000 data points--------------
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
End

Y_vld=awgn((2*X_vld)+1,15,'measured')';
% Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);

%*****************************************************************************************************
%--------------- Step 12: Construct the interpolation matrix-------------------------------
%--------------- from Xvld to Xctr -----------------------------------------------------------------
%*****************************************************************************************************
%--------------- Step 12.1: CS_RBF1 ----------------------------------------------------------------
for i=1:1000
for j=1:M_CS_RBF1
r2_CS_RBF1(i,j)=abs(X_vldN (1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
end

%-------------- Step 12.2: CS1 ------------------------------------------------------------------------------------
for i=1:1000
for j=1:M_CS1
r2_CS1(i,j)=abs(X_vldN (1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-
(14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%-------------- Step 12.3: CS2 ------------------------------------------------------------------------------------
for i=1:1000
for j=1:M_CS2
r2_CS2(i,j)=abs(X_vldN (1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-
(4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end

%-------------- Step 12.4: Gaussian RBF-----------------------------------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN (1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%*******************************************************************************************************
%-------------- Step 13: Construct the YvldapxN from Phi2---------------------------------------------------------
%--------------and weight 'w' ---------------------------------------------------------------------------------
%*******************************************************************************************************
%-------------- Step 13.1: CS_RBF1 ---------------------------------------------------------------------------------
YvldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
Yvldapx_CS_RBF1 = YvldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%-------------- Step 13.2: CS1 ------------------------------------------------------------------------------------
YvldapxN_CS1=Phi2_CS1*w_CS1;
Yvldapx_CS1 = YvldapxN_CS1.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%-------------- Step 13.3: CS2 ------------------------------------------------------------------------------------
YvldapxN_CS2=Phi2_CS2*w_CS2;
Yvldapx_CS2 = YvldapxN_CS2.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%-------------- Step 13.4: Gaussian RBF-----------------------------------------------------------------------------
YvldapxN=Phi2*w;
Yvldapx = YvldapxN.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%*******************************************************************************************************
%-------------- Step 14: Computing the validation error with MSE --------------------
%*******************************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld -Yvldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld -Yvldapx_CS1).^2);
Err_Vld_CS2=(1/1000)*sum((Y_vld -Yvldapx_CS2).^2);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Err_Vld =(1/1000)*sum((Y_vld -Yvldapx).^2);

T_Vld = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

%**********************************************************************************************
%--------------- Step 15: Results Plotting -----------------------------------------------------------
%**********************************************************************************************
figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
xlabel('Input Variable (x)', 'FontSize',14);
ylabel('Output Variable (y)', 'FontSize',14);
grid on

figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn, Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=2X+1', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
grid on

figure;
hold on
plot(X_vld,Y_vld,'ob','markersize',3);
plot(X_vld,Yvldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Yvldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Yvldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Yvldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=2X+1', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
grid on
%---------------------------------------- THIS IS THE END ------------------------------------------
```

A.1.1.2 Using different numbers of centres and RBFs for linear interpolation.

```
clc
clear all
close all

%**********************************************************************************************
%--------------- Step 1: Generate training dataset -------------------------------------------
%**********************************************************************************************
for i=1:100
X_trn (i)=2*pi*((i-1)/100);
end
Y_trn =awgn(2*X_trn +1,15,'measured')';
% Y_trn with Gaussian
for i=1:100
for j=1:100
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end

XY_trn;
N_tr = length(X_trn);

%******************************************************************************************************
%--------------- Step 2:Normalizing both X_trn and Y_trn ---------------------------------------------
%******************************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn-min(X_trn))./(x1max);
 % Normalized X_trn
Y_trnN=(Y_trn-min(Y_trn))./(max(Y_trn)-min(Y_trn));
% Normalized Y_trn

%******************************************************************************************************
%--------------- Step 3:Choosing the shape parameter-------------------------------------------------
%******************************************************************************************************
shp = input('Enter the shape parameter = ');

%******************************************************************************************************
%--------------- Step 4:Construct the interpolation matrix-------------------------------------------
%---------------CS_RBF1, CS1, CS2, G ----------------------------------------------------------------
%******************************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)-X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%******************************************************************************************************
%--------------- Step 5:Using SVD to get the number of centres --------------------------------------
%---------------------- out of matrix CS_RBF1, CS1, CS2, G -----------------------------------------
%******************************************************************************************************
delta = 0.01/100;

%--------------- Step 5.1:CS_RBF1 --------------------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
i =i+1 ;
end

%-------------- Step 5.2: CS1 --------------------------------------------------------------------------------
[U_CS1,S_CS1,V_CS1]=svd(CS1);
S11_CS1 =S_CS1(1,1);  %The firsts singular of singular value
S1_CS1 =S11_CS1*(delta);

for i=1:N_tr
Sii_CS1 =S_CS1(i,i);
if Sii_CS1 <=S1_CS1
M_CS1 =i-1;
break
end
i =i+1 ;
end

%-------------- Step 5.3: CS2 --------------------------------------------------------------------------------
[U_CS2,S_CS2,V_CS2]=svd(CS2);
S11_CS2 =S_CS2(1,1);  %The firsts singular of singular value
S1_CS2 =S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 =S_CS2(i,i);
if Sii_CS2 <=S1_CS2
M_CS2 =i-1;
break
end
i =i+1 ;
end

%-------------- Step 5.4: Gaussian RBF-------------------------------------------------------------------------
[U,S,V]=svd(G);
S11 =S(1,1);
S1 =S11*(delta);

for i=1:N_tr
Sii =S(i,i);
if Sii <= S1 ;
M =i-1
break
end
i =i+1 ;
end

%*********************************************************************************************************
%-------------- Step 6: Using QR to find the centres of basis function  ---
%---------------------- and the location of centres -----------------------------------------------
%*********************************************************************************************************
%-------------- Step 6.1: CS_RBF1 ----------------------------------------------------------------------------
V1_CS_RBF1 =-V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix V
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);V2_CS_RBF1 = [V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1]=qr(V2_CS_RBF1);
mu_CS_RBF1=X_trnN*P_CS_RBF1;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Xctr_CS_RBF1_ex = mu_CS_RBF1(1,1:M_CS_RBF1);
Xctr_CS_RBF1 = randsample(Xctr_CS_RBF1_ex,M)

%------------- Step 6.2: CS1 -----------------------------------------------------------------
V1_CS1 = -V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);              %partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 = [V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1] = qr(V2_CS1);
mu_CS1 = X_trnN*P_CS1;
Xctr_CS1_ex = mu_CS1(1,1:M_CS1);

Xctr_CS1 = randsample(Xctr_CS1_ex,M)

%------------- Step 6.3: CS2 -----------------------------------------------------------------
V1_CS2 = -V_CS2;
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);              %partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2 = [V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2] = qr(V2_CS2);
mu_CS2 = X_trnN*P_CS2;
Xctr_CS2_ex = mu_CS2(1,1:M_CS2);

Xctr_CS2 = randsample(Xctr_CS2_ex,M)

%------------- Step 6.4: Gaussian RBF--------------------------------------------------------
V1 = -V;
V11=V1(1:M,1:M);              %partition matrix V
V21=V1(M+1:N_tr,1:M);
V2 = [V11' V21'];

[Q,R,P] = qr(V2);
mu = X_trnN*P;
Xctr = mu(1,1:M)

%*******************************************************************************************
%------------- Step 7: Construct the interpolation matrix    -----------------------------
%*******************************************************************************************
%------------- Step 7.1: CS_RBF1 -------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j) == 0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

%------------- Step 7.2: CS1 -----------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS1(i,j)=abs(X_trnN(1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
end

%-------------- Step 7.3: CS2 ------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS2(i,j)=abs(X_trnN(1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-
(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j)== 0
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end

%-------------- Step 7.4: Gaussian RBF---------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN(1,i)-Xctr(1,j));
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%*****************************************************************************************
%-------------- Step 8: Computing the weight 'w' ------------------------------------------------
%*****************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 = pinv(Phi_CS_RBF1)*Y_trnN
w_CS1 = pinv(Phi_CS1)*Y_trnN
w_CS2 = pinv(Phi_CS2)*Y_trnN
w=pinv(Phi)*Y_trnN

%*****************************************************************************************
%-------------- Step 9: Computing the approximation of training dataset -
%*****************************************************************************************
%-------------- Step 9.1: CS_RBF1 ------------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 = YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.2: CS1 -----------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 = YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.3: CS2 -----------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 = YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.4: Gassian RBF---------------------------------------------------------
YappxN=Phi*w;
Yappx = YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%*****************************************************************************************
%-------------- Step 10: Computing the training error with MSE------------------------------
%*****************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn-Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn-Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn-Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn-Yappx).^2);

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
T_Trn = table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%**************************************************************************************************
%--------------- Step 11: Generate validation dataset -----------------------------------------------
%**************************************************************************************************
%----//// Generate a validation dataset of 1000 data points in  ////---------
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
end
Y_vld=awgn((2*X_vld)+1,15,'measured')';
% Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);
%**************************************************************************************************
%--------------- Step 12: Construct the interpolation matrix-----------------------------------
%---------------from X_vld to Xctr -----------------------------------------------------------------
%**************************************************************************************************
%--------------- Step 12.1: CS_RBF1 -----------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS_RBF1(i,j)=abs(X_vldN(1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-
    (4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 12.2: CS1 ---------------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS1(i,j)=abs(X_vldN(1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-
    (14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 12.3: CS2 ---------------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS2(i,j)=abs(X_vldN(1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-
    (4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

%--------------- Step 12.4: Gaussian RBF-----------------------------------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN(1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%*****************************************************************************************************************
%--------------- Step 13: Construct the Y_vldapxN from Phi2 and weight 'w'
%*****************************************************************************************************************
%--------------- Step 13.1: CS_RBF1 ----------------------------------------------------------------------------------
Y_vldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
Y_vldapx_CS_RBF1 = Y_vldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.2: CS1 --------------------------------------------------------------------------------------
Y_vldapxN_CS1=Phi2_CS1*w_CS1;
Y_vldapx_CS1 = Y_vldapxN_CS1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.3: CS2 --------------------------------------------------------------------------------------
Y_vldapxN_CS2=Phi2_CS2*w_CS2;
Y_vldapx_CS2 = Y_vldapxN_CS2.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.4: Gassian RBF--------------------------------------------------------------------------------
Y_vldapxN=Phi2*w;
Y_vldapx = Y_vldapxN.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%*****************************************************************************************************************
%--------------- Step 14: Computing the validation error with MSE --------------------
%*****************************************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld-Y_vldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld-Y_vldapx_CS1).^2);
Err_Vld_CS2=(1/1000)*sum((Y_vld-Y_vldapx_CS2).^2);
Err_Vld =(1/1000)*sum((Y_vld-Y_vldapx).^2);

T_Vld = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

Y1 = 2*X_trn+1;

for i=1:100
Y3(i) = 20;
End

%----- Step 12 : Results Plotting ------------------------------------------------------------------------------------
figure;
hold on
plot(X_trn,Y_trn,'ok','LineWidth',1);
plot(X_trn,Y3,'ok','LineWidth',1);
plot(X_trn,Y1,'-k','LineWidth',1);
xlabel('x', 'FontSize',14);
ylabel('y=2x+1', 'FontSize',14);
grid on

figure;
hold on
plot(X_trn,Y_trn,'ok','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn,Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=2X+1', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
grid on

figure;
hold on
plot(X_vld,Y_vld,'ok','markersize',3);
plot(X_vld,Y_vldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Y_vldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Y_vldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Y_vldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=2X+1', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
%---------------------------------------------------------- THIS IS THE END ---------------------------------------------
```

A.1.2 Test Case 2: Parabola function

A.1.2.1 Training and validation errors for candidate models for parabola function case.

```
clc
clear all
close all

%********************************************************************************************************************
%-------------- Step 1:Generate training dataset ---------------------------------------------------------------------
%********************************************************************************************************************
for i=1:100
X_trn(i)=2*pi*((i-1)/100);
end
Y_trn =awgn((X_trn.^2)/6- X_trn +4,15,'measured')';
for i=1:100
for j=1:100
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end
XY_trn;

N_tr = length(X_trn);

%********************************************************************************************************************
%-------------- Step 2:Normalizing both X_trn and Y_trn -----------------------------------------
%********************************************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn -min(X_trn))./(x1max);                    %Normalized X_trn
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Y_trnN=(Y_trn -min(Y_trn))./(max(Y_trn)-min(Y_trn));   % Normalized Y_trn

%****************************************************************************************
%--------------- Step 3: Choosing the shape parameter-----------------------------------
%****************************************************************************************
shp = input('Enter the shape parameter = ');

%****************************************************************************************
%--------------- Step 4: Construct the interpolation matrix----------------------------
%--------------- CS_RBF1, CS1, CS2, G -----------------------------------------------------
%****************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)-X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%****************************************************************************************
%--------------- Step 5: Using SVD to get the number of centres -----------------------
%----------------------- out of matrix CS_RBF1, CS1, CS2, G ------------------------------
%****************************************************************************************
delta = 0.01/100;

%--------------- Step 5.1: CS_RBF1 -------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
i = i+1 ;
end

%--------------- Step 5.2: CS1 ------------------------------------------------------------
[U_CS1,S_CS1,V_CS1] = svd(CS1);
S11_CS1 = S_CS1(1,1);
% The firsts singular of singular value
S1_CS1 = S11_CS1*(delta);
for i=1:N_tr
Sii_CS1 = S_CS1(i,i);
if Sii_CS1 <= S1_CS1
M_CS1 = i-1;
break
end
i = i+1 ;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end

%-------------- Step 5.3: CS2 -------------------------------------------------------------------------
[U_CS2,S_CS2,V_CS2] = svd(CS2);
S11_CS2 = S_CS2(1,1);
% The firsts singular of singular value
S1_CS2 = S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 = S_CS2(i,i);
if Sii_CS2 <= S1_CS2
M_CS2 = i-1;
break
end
i = i+1 ;
end
%-------------- Step 5.4: Gaussian RBF-----------------------------------------------------------------
[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1 ;
break
end
i = i+1 ;
end

%****************************************************************************************************
%-------------- Step 6: Using QR to find the centres of-----------------------------------------------
%------------------------- basis function and the location of centres ---------------------
%****************************************************************************************************
%-------------- Step 6.1: CS_RBF1 ------------------------------------------------------------------
V1_CS_RBF1 = V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);
V2_CS_RBF1 = [V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1] = qr(V2_CS_RBF1);
mu_CS_RBF1 = X_trnN*P_CS_RBF1;
Xctr_CS_RBF1 = mu_CS_RBF1(1,1:M_CS_RBF1);

%-------------- Step 6.2: CS1 ----------------------------------------------------------------------
V1_CS1 = -V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);
%partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 = [V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1] = qr(V2_CS1);
mu_CS1 = X_trnN*P_CS1;
Xctr_CS1 = mu_CS1(1,1:M_CS1);

%-------------- Step 6.3: CS2 ----------------------------------------------------------------------
V1_CS2 = -V_CS2;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);
%partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2 =[V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2]= qr(V2_CS2);
mu_CS2 = X_trnN*P_CS2;
Xctr_CS2 = mu_CS2(1,1:M_CS2);

%--------------- Step 6.4: Gaussian RBF----------------------------------------------------------------------------------
V1 = -V;
V11=V1(1:M,1:M) ;              %partition matrix V
V21=V1(M+1:N_tr,1:M) ;
V2 =[V11' V21'];
[Q,R,P]= qr(V2);
mu = X_trnN*P;
Xctr = mu(1,1:M)

%***************************************************************************************************
%--------------- Step 7: Construct the interpolation matrix ------------------------------
%***************************************************************************************************
%--------------- Step 7.1: CS_RBF1 -------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS_RBF1
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j)== 0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 7.2: CS1 ----------------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS1
r1_CS1(i,j)=abs(X_trnN (1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 7.3: CS2 ----------------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS2
r1_CS2(i,j)=abs(X_trnN (1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j)== 0
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end
end

%--------------- Step 7.4: Gaussian RBF----------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN (1,i)-Xctr(1,j));
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%************************************************************************************************
%-------------- Step 8: Computing the weight 'w' -----------------------------------------------
%************************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 = pinv(Phi_CS_RBF1)*Y_trnN;
w_CS1 = pinv(Phi_CS1)*Y_trnN;
w_CS2 = pinv(Phi_CS2)*Y_trnN;
w=pinv(Phi)*Y_trnN;

%************************************************************************************************
%-------------- Step 9: Computing the approximation of ----------------------------------------
%-------------- training dataset-----------------------------------------------------------------
%************************************************************************************************
%-------------- Step 9.1: CS_RBF1 --------------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 = YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.2: CS1 ------------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 = YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.3: CS2 ------------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 = YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%-------------- Step 9.4: Gaussian RBF-----------------------------------------------------------
YappxN=Phi*w;
Yappx = YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%************************************************************************************************
%-------------- Step 10: Computing the training error with MSE---------------------------
%************************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn -Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn -Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn -Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn -Yappx).^2);

T_Trn = table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%************************************************************************************************
%-------------- Step 11: Generate validation dataset -------------------------------------------
%************************************************************************************************
%-------------//// Generate a validation dataset of 1000 data points
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
End

Y_vld=awgn((X_vld.^2)/6-X_vld+4,15,'measured')';
% Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);

%************************************************************************************************************
%--------------- Step 12:Construct the interpolation matrix-----------------------------------
%--------------- from Xvld to Xctr ------------------------------------------------------------------
%************************************************************************************************************
%--------------- Step 12.1:CS_RBF1 ---------------------------------------------------------------
for i=1:1000
for j=1:M_CS_RBF1
r2_CS_RBF1(i,j)=abs(X_vldN (1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-
   (4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 12.2:CS1 -------------------------------------------------------------------
for i=1:1000
for j=1:M_CS1
r2_CS1(i,j)=abs(X_vldN (1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-
   (14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 12.3:CS2 -------------------------------------------------------------------
for i=1:1000
for j=1:M_CS2
r2_CS2(i,j)=abs(X_vldN (1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-
   (4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end

%--------------- Step 12.4:Gaussian RBF-----------------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN (1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%************************************************************************************************************
%--------------- Step 13:Construct the YvldapxN from Phi2-----------------------------------------
%---------------and weight 'w' --------------------------------------------------------------------
%************************************************************************************************************
%--------------- Step 13.1:CS_RBF1 ---------------------------------------------------------------
YvldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Yvldapx_CS_RBF1 = YvldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.2: CS1 ------------------------------------------------------------------------------------------------
YvldapxN_CS1=Phi2_CS1*w_CS1;
Yvldapx_CS1 = YvldapxN_CS1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.3: CS2 ------------------------------------------------------------------------------------------------
YvldapxN_CS2=Phi2_CS2*w_CS2;
Yvldapx_CS2 = YvldapxN_CS2.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.4: Gaussian RBF-----------------------------------------------------------------------------------------
YvldapxN=Phi2*w;
Yvldapx = YvldapxN.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%*****************************************************************************************************************************
%--------------- Step 14: Computing the validation error with MSE --------------------
%*****************************************************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld -Yvldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld -Yvldapx_CS1).^2);
Err_Vld_CS2=(1/1000)*sum((Y_vld -Yvldapx_CS2).^2);
Err_Vld =(1/1000)*sum((Y_vld -Yvldapx).^2);

T_Vld = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

%*****************************************************************************************************************************
%--------------- Step 15: Results Plotting ----------------------------------------------------------------------------------
%*****************************************************************************************************************************
figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
xlabel('Input Variable (x)', 'FontSize',14);
ylabel('Output Variable (y)', 'FontSize',14);
grid on

figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn, Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=(X.^2)/6-X+4', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
grid on

figure;
hold on
plot(X_vld,Y_vld,'ob','markersize',3);
plot(X_vld,Yvldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Yvldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Yvldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Yvldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=(X.^2)/6-X+4', 'FontSize',14);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location', 'SouthEast')
grid on

%--------------------------------------- THIS IS THE END ------------------------------------------------------------
```

A.1.2.2 Using different numbers of centres and RBFs for parabola function.

```
clc
clear all
close all

%*************************************************************************************************
%-------------- Step 1: Generate training dataset ----------------------------------------
%*************************************************************************************************
for i=1:100
X_trn (i)=2*pi*((i-1)/100);
end
Y_trn =awgn((X_trn.^2)/6- X_trn +4,15,'measured')';
% Y_trn with Gaussian
for i=1:100
for j=1:100
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end

XY_trn;
N_tr = length(X_trn);

%*************************************************************************************************
%-------------- Step 2: Normalizing both X_trn and Y_trn ------------------
%*************************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn-min(X_trn))./(x1max);
 % Normalized X_trn
Y_trnN=(Y_trn-min(Y_trn))./(max(Y_trn)-min(Y_trn));
% Normalized Y_trn

%*************************************************************************************************
%-------------- Step 3: Choosing the shape parameter--------------------------------------------
%*************************************************************************************************
shp = input('Enter the shape parameter = ');

%*************************************************************************************************
%-------------- Step 4: Construct the interpolation matrix----------------------------------
%--------------CS_RBF1, CS1, CS2, G -----------------------------------------------------------------
%*************************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)- X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%*******************************************************************************************
%---------- Step 5:Using SVD to get the number of centres ----------------------
%--------------- out of matrix CS_RBF1, CS1, CS2, G ----------------------------------
%*******************************************************************************************
delta = 0.01/100;

%---------- Step 5.1:CS_RBF1 --------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
i = i+1 ;
end

%---------- Step 5.2:CS1 ------------------------------------------------------------
[U_CS1,S_CS1,V_CS1] = svd(CS1);
S11_CS1 = S_CS1(1,1);  % The firsts singular of singular value
S1_CS1 = S11_CS1*(delta);

for i=1:N_tr
Sii_CS1 = S_CS1(i,i);
if Sii_CS1 <= S1_CS1
M_CS1 = i-1;
break
end
i = i+1 ;
end

%---------- Step 5.3:CS2 ------------------------------------------------------------
[U_CS2,S_CS2,V_CS2] = svd(CS2);
S11_CS2 = S_CS2(1,1);  % The firsts singular of singular value
S1_CS2 = S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 = S_CS2(i,i);
if Sii_CS2 <= S1_CS2
M_CS2 = i-1;
break
end
i = i+1 ;
end

%---------- Step 5.4:Gaussian RBF----------------------------------------------------
[U,S,V] = svd(G);
S11 = S(1,1);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
S1 = S11*(delta);

for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1
break
end
i = i+1 ;
end

%*************************************************************************************************************
%---------------- Step 6: Using QR to find the centres of basis function ---
%------------------------ and the location of centres ------------------------------------------------------------
%*************************************************************************************************************
%---------------- Step 6.1: CS_RBF1 ---------------------------------------------------------------------------------
V1_CS_RBF1 = -V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix V
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);V2_CS_RBF1 = [V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1] = qr(V2_CS_RBF1);
mu_CS_RBF1 = X_trnN*P_CS_RBF1;
Xctr_CS_RBF1_ex = mu_CS_RBF1(1,1:M_CS_RBF1);
Xctr_CS_RBF1 = randsample(Xctr_CS_RBF1_ex,M)

%---------------- Step 6.2: CS1 ---------------------------------------------------------------------------------------
V1_CS1 = -V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);               %partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 = [V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1] = qr(V2_CS1);
mu_CS1 = X_trnN*P_CS1;
Xctr_CS1_ex = mu_CS1(1,1:M_CS1);

Xctr_CS1 = randsample(Xctr_CS1_ex,M)

%---------------- Step 6.3: CS2 ---------------------------------------------------------------------------------------
V1_CS2 = -V_CS2;
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);               %partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2 = [V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2] = qr(V2_CS2);
mu_CS2 = X_trnN*P_CS2;
Xctr_CS2_ex = mu_CS2(1,1:M_CS2);

Xctr_CS2 = randsample(Xctr_CS2_ex,M)

%---------------- Step 6.4: Gaussian RBF----------------------------------------------------------------------------
V1 = -V;
V11=V1(1:M,1:M);               %partition matrix V
V21=V1(M+1:N_tr,1:M);
V2 = [V11' V21'];

[Q,R,P] = qr(V2);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
mu = X_trnN*P;
Xctr = mu(1,1:M)

%****************************************************************************************************
%--------------- Step 7: Construct the interpolation matrix------------------------------------------
%****************************************************************************************************
%--------------- Step 7.1: CS_RBF1 -----------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j) == 0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 7.2: CS1 -------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS1(i,j)=abs(X_trnN(1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 7.3: CS2 -------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS2(i,j)=abs(X_trnN(1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j) ==  0
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end

%--------------- Step 7.4: Gaussian RBF-----------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN(1,i)-Xctr(1,j));
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%****************************************************************************************************
%--------------- Step 8: Computing the weight 'w' --------------------------------------------------
%****************************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 = pinv(Phi_CS_RBF1)*Y_trnN
w_CS1 = pinv(Phi_CS1)*Y_trnN
w_CS2 = pinv(Phi_CS2)*Y_trnN
w=pinv(Phi)*Y_trnN

%****************************************************************************************************
%--------------- Step 9: Computing the approximation of training dataset ---
%****************************************************************************************************
```

```
%--------------- Step 9.1: CS_RBF1 -------------------------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 = YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.2: CS1 -----------------------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 = YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.3: CS2 -----------------------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 = YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.4: Gassian RBF------------------------------------------------------------------------
YappxN=Phi*w;
Yappx = YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%****************************************************************************************************
%--------------- Step 10: Computing the training error with MSE---------------------------
%****************************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn-Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn-Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn-Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn-Yappx).^2);

T_Trn = table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%****************************************************************************************************
%--------------- Step 11: Generate validation dataset -----------------------------------------
%****************************************************************************************************
%----//// Generate a validation dataset of 1000 data points in ////--------
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
end
Y_vld=awgn((X_vld.^2)/6-X_vld+4,15,'measured')';
% Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);
%****************************************************************************************************
%--------------- Step 12: Construct the interpolation matrix---------------------------------
%---------------from X_vld to Xctr -----------------------------------------------------------------
%****************************************************************************************************
%--------------- Step 12.1: CS_RBF1 ------------------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS_RBF1(i,j)=abs(X_vldN(1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 12.2: CS1 -----------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS1(i,j)=abs(X_vldN(1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-(14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 12.3: CS2 -----------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS2(i,j)=abs(X_vldN(1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-(4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end

%--------------- Step 12.4: Gaussian RBF-----------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN(1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%******************************************************************************************
%--------------- Step 13: Construct the Y_vldapxN from Phi2 and weight 'w'
%******************************************************************************************
%--------------- Step 13.1: CS_RBF1 -------------------------------------------------------
Y_vldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
Y_vldapx_CS_RBF1 = Y_vldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.2: CS1 -----------------------------------------------------------
Y_vldapxN_CS1=Phi2_CS1*w_CS1;
Y_vldapx_CS1 = Y_vldapxN_CS1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.3: CS2 -----------------------------------------------------------
Y_vldapxN_CS2=Phi2_CS2*w_CS2;
Y_vldapx_CS2 = Y_vldapxN_CS2.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.4: Gassian RBF-----------------------------------------------------
Y_vldapxN=Phi2*w;
Y_vldapx = Y_vldapxN.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%******************************************************************************************
%--------------- Step 14: Computing the validation error with MSE --------------------
%******************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld-Y_vldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld-Y_vldapx_CS1).^2);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Err_Vld_CS2=(1/1000)*sum((Y_vld-Y_vldapx_CS2).^2);
Err_Vld  =(1/1000)*sum((Y_vld-Y_vldapx).^2);

T_Vld  = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

Y1  =(X_trn.^2)/6- X_trn +4;

for i=1:100
Y3(i)= 20;
End

%----- Step 12 : Results Plotting ------------------------------------------------------------
figure;
hold on
plot(X_trn,Y_trn,'ok','LineWidth',1);
plot(X_trn,Y3,'ok','LineWidth',1);
plot(X_trn,Y1,'-k','LineWidth',1);
xlabel('x', 'FontSize',14);
ylabel('Y=(X.^2)/6-X+4', 'FontSize',14);
grid on

figure;
hold on
plot(X_trn,Y_trn,'ok','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn,Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=(X.^2)/6-X+4', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location','SouthEast')
grid on

figure;
hold on
plot(X_vld,Y_vld,'ok','markersize',3);
plot(X_vld,Y_vldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Y_vldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Y_vldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Y_vldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=(X.^2)/6-X+4', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location','SouthEast')
%--------------------------------------- THIS IS THE END ---------------------------------------
```

A.1.3 Test Case 3: Sine interpolation

A.1.3.1 Training and validation errors for candidate models for sine interpolation case.

```
clc
clear all
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
close all

%*****************************************************************************************************
%--------------- Step 1: Generate training dataset ------------------------------------------------
%*****************************************************************************************************
for i=1:100
X_trn(i)=2*pi*((i-1)/100);
end
Y_trn=awgn(sin(X_trn),0.5,'measured')';
for i=1:100
for j=1:100
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end
XY_trn;

N_tr = length(X_trn);

%*****************************************************************************************************
%--------------- Step 2: Normalizing both X_trn and Y_trn ------------------------------------------
%*****************************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn -min(X_trn))./(x1max);              % Normalized X_trn
Y_trnN=(Y_trn -min(Y_trn))./(max(Y_trn)-min(Y_trn));  % Normalized Y_trn

%*****************************************************************************************************
%--------------- Step 3: Choosing the shape parameter------------------------------------------------
%*****************************************************************************************************
shp = input('Enter the shape parameter = ');

%*****************************************************************************************************
%--------------- Step 4: Construct the interpolation matrix-----------------------------------------
%--------------- CS_RBF1, CS1, CS2, G ---------------------------------------------------------------
%*****************************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)-X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%*****************************************************************************************************
%--------------- Step 5: Using SVD to get the number of centres -------------------------------------
%---------------------- out of matrix CS_RBF1, CS1, CS2, G -------------------------------------------
%*****************************************************************************************************
delta = 0.01/100;

%--------------- Step 5.1: CS_RBF1 -------------------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
```

```
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
i = i+1 ;
end

%-------------- Step 5.2: CS1 -------------------------------------------------------------------------------
[U_CS1,S_CS1,V_CS1] = svd(CS1);
S11_CS1 = S_CS1(1,1);
% The firsts singular of singular value
S1_CS1 = S11_CS1*(delta);
for i=1:N_tr
Sii_CS1 = S_CS1(i,i);
if Sii_CS1 <= S1_CS1
M_CS1 = i-1;
break
end
i = i+1 ;
end

%-------------- Step 5.3: CS2 -------------------------------------------------------------------------------
[U_CS2,S_CS2,V_CS2] = svd(CS2);
S11_CS2 = S_CS2(1,1);
% The firsts singular of singular value
S1_CS2 = S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 = S_CS2(i,i);
if Sii_CS2 <= S1_CS2
M_CS2 = i-1;
break
end
i = i+1 ;
end
%-------------- Step 5.4: Gaussian RBF------------------------------------------------------------------------
[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1 ;
break
end
i = i+1 ;
end

%****************************************************************************************************
%-------------- Step 6: Using QR to find the centres of----------------------------------------------
%---------------------- basis function and the location of centres ----------------------
%****************************************************************************************************
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%--------------- Step 6.1: CS_RBF1 ------------------------------------------------------------------------
V1_CS_RBF1 =-V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);
V2_CS_RBF1 =[V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1]=qr(V2_CS_RBF1);
mu_CS_RBF1 =X_trnN*P_CS_RBF1;
Xctr_CS_RBF1 =mu_CS_RBF1(1,1:M_CS_RBF1);

%--------------- Step 6.2: CS1 ------------------------------------------------------------------------------
V1_CS1 =-V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);
%partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 =[V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1]=qr(V2_CS1);
mu_CS1 =X_trnN*P_CS1;
Xctr_CS1 =mu_CS1(1,1:M_CS1);

%--------------- Step 6.3: CS2 ------------------------------------------------------------------------------
V1_CS2 =-V_CS2;
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);
%partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2 =[V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2]=qr(V2_CS2);
mu_CS2 =X_trnN*P_CS2;
Xctr_CS2 =mu_CS2(1,1:M_CS2);

%--------------- Step 6.4: Gaussian RBF---------------------------------------------------------------------
V1 =-V;
V11=V1(1:M,1:M);                        %partition matrix V
V21=V1(M+1:N_tr,1:M);
V2 =[V11' V21'];
[Q,R,P]=qr(V2);
mu =X_trnN*P;
Xctr =mu(1,1:M)

%**********************************************************************************************************
%--------------- Step 7: Construct the interpolation matrix-------------------------------------------------
%**********************************************************************************************************
%--------------- Step 7.1: CS_RBF1 ------------------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS_RBF1
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j)==0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%--------------- Step 7.2: CS1 ------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS1
r1_CS1(i,j)=abs(X_trnN (1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 7.3: CS2 ------------------------------------------------------------
for i=1:N_tr
for j=1:M_CS2
r1_CS2(i,j)=abs(X_trnN (1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j)==0
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end
end

%--------------- Step 7.4: Gaussian RBF------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN (1,i)-Xctr(1,j));
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%******************************************************************************************
%--------------- Step 8: Computing the weight 'w' ------------------------------------------
%******************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 =pinv(Phi_CS_RBF1)*Y_trnN;
w_CS1 =pinv(Phi_CS1)*Y_trnN;
w_CS2 =pinv(Phi_CS2)*Y_trnN;
w=pinv(Phi)*Y_trnN;

%******************************************************************************************
%--------------- Step 9: Computing the approximation of ------------------------------------
%--------------- training dataset----------------------------------------------------------
%******************************************************************************************
%--------------- Step 9.1: CS_RBF1 ---------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 = YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.2: CS1 -------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 = YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.3: CS2 -------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 = YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.4: Gaussian RBF------------------------------------------------------
YappxN=Phi*w;
Yappx = YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%*****************************************************************************************
%-------------- Step 10:Computing the training error with MSE---------------------------
%*****************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn -Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn -Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn -Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn -Yappx).^2);

T_Trn = table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%*****************************************************************************************
%-------------- Step 11:Generate validation dataset ------------------------------------------
%*****************************************************************************************
%-------------//// Generate a validation dataset of 1000 data points ////---------
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
End

Y_vld=awgn(sin(X_vld.),15,'measured')';
% Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);

%*****************************************************************************************
%-------------- Step 12:Construct the interpolation matrix-------------------------------
%-------------- from Xvld to Xctr ------------------------------------------------------------
%*****************************************************************************************
%-------------- Step 12.1:CS_RBF1 -----------------------------------------------------------
for i=1:1000
for j=1:M_CS_RBF1
r2_CS_RBF1(i,j)=abs(X_vldN (1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-
    (4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
end
end

%-------------- Step 12.2:CS1 -----------------------------------------------------------------
for i=1:1000
for j=1:M_CS1
r2_CS1(i,j)=abs(X_vldN (1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-
    (14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%-------------- Step 12.3:CS2 -----------------------------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนาร

```
for i=1:1000
for j=1:M_CS2
r2_CS2(i,j)=abs(X_vldN (1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-(4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end

%--------------- Step 12.4: Gaussian RBF----------------------------------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN (1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%****************************************************************************************************************
%--------------- Step 13: Construct the YvldapxN from Phi2---------------------------------------------------------
%---------------and weight 'w' -----------------------------------------------------------------------------------
%****************************************************************************************************************
%--------------- Step 13.1: CS_RBF1 -------------------------------------------------------------------------------
YvldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
Yvldapx_CS_RBF1 = YvldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.2: CS1 -----------------------------------------------------------------------------------
YvldapxN_CS1=Phi2_CS1*w_CS1;
Yvldapx_CS1 = YvldapxN_CS1.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.3: CS2 -----------------------------------------------------------------------------------
YvldapxN_CS2=Phi2_CS2*w_CS2;
Yvldapx_CS2 = YvldapxN_CS2.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.4: Gaussian RBF----------------------------------------------------------------------------
YvldapxN=Phi2*w;
Yvldapx = YvldapxN.* (max(Y_vld)-min(Y_vld))+min(Y_vld);

%****************************************************************************************************************
%--------------- Step 14: Computing the validation error with MSE -------------------------------------------------
%****************************************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld -Yvldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld -Yvldapx_CS1).^2);
Err_Vld_CS2=(1/1000)*sum((Y_vld -Yvldapx_CS2).^2);
Err_Vld =(1/1000)*sum((Y_vld -Yvldapx).^2);

T_Vld = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

%****************************************************************************************************************
%--------------- Step 15: Results Plotting ------------------------------------------------------------------------
%****************************************************************************************************************
figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
xlabel('Input Variable (x)', 'FontSize',14);
ylabel('Output Variable (y)', 'FontSize',14);
grid on
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

figure;
hold on
plot(X_trn, Y_trn,'ob','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn, Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=sin(X)', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
    'SouthEast')
grid on

figure;
hold on
plot(X_vld,Y_vld,'ob','markersize',3);
plot(X_vld,Yvldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Yvldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Yvldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Yvldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=sin(X)', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
    'SouthEast')
grid on

%---------------------------------------------------- THIS IS THE END ----------------------------------------------------
```

A.1.3.2 Using different numbers of centres and RBFs for parabola function.

```
clc
clear all
close all

%*****************************************************************************************************************
%-------------- Step 1: Generate training dataset ------------------------------------------------------
%*****************************************************************************************************************
for i=1:100
X_trn (i)=2*pi*((i-1)/100);
End
Y_trn=awgn(sin(X_trn),0.5,'measured')';
% Y_trn with Gaussian
for i=1:100
for j=1:100
XY_trn (i,1)=X_trn (i);
XY_trn (j,2)=Y_trn (j);
end
end

XY_trn;
N_tr = length(X_trn);

%*****************************************************************************************************************
%--------------- Step 2: Normalizing both X_trn and Y_trn ------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%***********************************************************************************************
x1max = max(X_trn);
X_trnN=(X_trn-min(X_trn))./(x1max);
 % Normalized X_trn
Y_trnN=(Y_trn-min(Y_trn))./(max(Y_trn)-min(Y_trn));
% Normalized Y_trn

%***********************************************************************************************
%-------------- Step 3: Choosing the shape parameter------------------------------------------------
%***********************************************************************************************
shp = input('Enter the shape parameter = ');

%***********************************************************************************************
%-------------- Step 4: Construct the interpolation matrix---------------------------------------
%--------------CS_RBF1, CS1, CS2, G ------------------------------------------------------------
%***********************************************************************************************
for i=1:N_tr
for j=1:N_tr
r(i,j)=abs(X_trnN (1,i)- X_trnN (1,j));
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3+2*(r(i,j).^2).*log(r(i,j));
CS1(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
if  r(i,j) == 0
CS_RBF1(i,j)=(1/3)+r(i,j).^2-(4/3)*r(i,j).^3;
CS2(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%***********************************************************************************************
%-------------- Step 5: Using SVD to get the number of centres---------------------------------
%---------------------- out of matrix CS_RBF1, CS1, CS2, G --------------------------------------
%***********************************************************************************************
delta = 0.01/100;

%-------------- Step 5.1: CS_RBF1 ---------------------------------------------------------------
[U_CS_RBF1,S_CS_RBF1,V_CS_RBF1] = svd(CS_RBF1);
S11_CS_RBF1 = S_CS_RBF1(1,1);
% The firsts singular of singular value
S1_CS_RBF1 = S11_CS_RBF1*(delta);

for i=1:N_tr
Sii_CS_RBF1 = S_CS_RBF1(i,i);
if Sii_CS_RBF1 <= S1_CS_RBF1
M_CS_RBF1 = i-1;
break
end
i = i+1 ;
end

%-------------- Step 5.2: CS1 -------------------------------------------------------------------
[U_CS1,S_CS1,V_CS1] = svd(CS1);
S11_CS1 = S_CS1(1,1);  % The firsts singular of singular value
S1_CS1 = S11_CS1*(delta);

for i=1:N_tr
Sii_CS1 = S_CS1(i,i);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
if Sii_CS1 <= S1_CS1
M_CS1 = i-1;
break
end
i = i+1 ;
end

%------------ Step 5.3: CS2 ------------------------------------------------------------------------
[U_CS2,S_CS2,V_CS2] = svd(CS2);
S11_CS2 = S_CS2(1,1);  % The firsts singular of singular value
S1_CS2 = S11_CS2*(delta);

for i=1:N_tr
Sii_CS2 = S_CS2(i,i);
if Sii_CS2 <= S1_CS2
M_CS2 = i-1;
break
end
i = i+1 ;
end

%------------ Step 5.4: Gaussian RBF-----------------------------------------------------------------
[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1
break
end
i = i+1 ;
end

%****************************************************************************************************
%------------ Step 6: Using QR to find the centres of basis function  --
%-------------------- and the location of centres ---------------------------------------------------
%****************************************************************************************************
%------------ Step 6.1: CS_RBF1 --------------------------------------------------------------------
V1_CS_RBF1 = -V_CS_RBF1;
V11_CS_RBF1=V1_CS_RBF1(1:M_CS_RBF1,1:M_CS_RBF1);
%partition matrix V
V21_CS_RBF1=V1_CS_RBF1(M_CS_RBF1+1:N_tr,1:M_CS_RBF1);V2_CS_RBF1 = [V11_CS_RBF1' V21_CS_RBF1'];

[Q_CS_RBF1,R_CS_RBF1,P_CS_RBF1] = qr(V2_CS_RBF1);
mu_CS_RBF1 = X_trnN*P_CS_RBF1;
Xctr_CS_RBF1_ex = mu_CS_RBF1(1,1:M_CS_RBF1);
Xctr_CS_RBF1 = randsample(Xctr_CS_RBF1_ex,M)

%------------ Step 6.2: CS1 ------------------------------------------------------------------------
V1_CS1 = -V_CS1;
V11_CS1=V1_CS1(1:M_CS1,1:M_CS1);            %partition matrix V
V21_CS1=V1_CS1(M_CS1+1:N_tr,1:M_CS1);
V2_CS1 = [V11_CS1' V21_CS1'];

[Q_CS1,R_CS1,P_CS1] = qr(V2_CS1);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
mu_CS1  = X_trnN*P_CS1;
Xctr_CS1_ex  = mu_CS1(1,1:M_CS1);

Xctr_CS1  = randsample(Xctr_CS1_ex,M)

%--------------- Step 6.3: CS2 ---------------------------------------------------------------------------------
V1_CS2  = -V_CS2;
V11_CS2=V1_CS2(1:M_CS2,1:M_CS2);              %partition matrix V
V21_CS2=V1_CS2(M_CS2+1:N_tr,1:M_CS2);
V2_CS2  = [V11_CS2' V21_CS2'];

[Q_CS2,R_CS2,P_CS2] = qr(V2_CS2);
mu_CS2  = X_trnN*P_CS2;
Xctr_CS2_ex  = mu_CS2(1,1:M_CS2);

Xctr_CS2 = randsample(Xctr_CS2_ex,M)

%--------------- Step 6.4: Gaussian RBF--------------------------------------------------------------------------
V1  = -V;
V11=V1(1:M,1:M);              %partition matrix V
V21=V1(M+1:N_tr,1:M);
V2  = [V11' V21'];

[Q,R,P] = qr(V2);
mu  = X_trnN*P;
Xctr  = mu(1,1:M)

%*************************************************************************************************************
%--------------- Step 7: Construct the interpolation matrix-------------------------------------------------
%*************************************************************************************************************
%--------------- Step 7.1: CS_RBF1 -------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS_RBF1(i,j)=abs(X_trnN(1,i)-Xctr_CS_RBF1(1,j));
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3+2*(r1_CS_RBF1(i,j).^2).*log(r1_CS_RBF1(i,j));
if  r1_CS_RBF1(i,j) == 0
Phi_CS_RBF1(i,j)=(1/3)+r1_CS_RBF1(i,j).^2-(4/3)*r1_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 7.2: CS1 --------------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS1(i,j)=abs(X_trnN(1,i)-Xctr_CS1(1,j));
Phi_CS1(i,j)=(112/45)*r1_CS1(i,j).^(9/2)+(16/3)*r1_CS1(i,j).^(7/2)-7*r1_CS1(i,j).^4-(14/15)*r1_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 7.3: CS2 --------------------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1_CS2(i,j)=abs(X_trnN(1,i)-Xctr_CS2(1,j));
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4-(4/3)*r1_CS2(i,j).^3.*log(r1_CS2(i,j));
if  r1_CS2(i,j) ==  0
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Phi_CS2(i,j)=(1/18)-r1_CS2(i,j).^2+(4/9)*r1_CS2(i,j).^3+0.5*r1_CS2(i,j).^4;
end
end

%--------------- Step 7.4: Gaussian RBF------------------------------------------------------------------------
for i=1:N_tr
for j=1:M
r1(i,j)=abs(X_trnN(1,i)-Xctr(1,j));
Phi(i,j)=exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%*****************************************************************************************************
%--------------- Step 8: Computing the weight 'w' ----------------------------------------------------
%*****************************************************************************************************
%Moore-Penrose Pseudoinverse of matrix
w_CS_RBF1 =pinv(Phi_CS_RBF1)*Y_trnN
w_CS1 =pinv(Phi_CS1)*Y_trnN
w_CS2 =pinv(Phi_CS2)*Y_trnN
w=pinv(Phi)*Y_trnN

%*****************************************************************************************************
%--------------- Step 9: Computing the approximation of training dataset --
%*****************************************************************************************************
%--------------- Step 9.1: CS_RBF1 -------------------------------------------------------------------
YappxN_CS_RBF1=Phi_CS_RBF1*w_CS_RBF1;
Yappx_CS_RBF1 =YappxN_CS_RBF1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.2: CS1 -----------------------------------------------------------------------
YappxN_CS1=Phi_CS1*w_CS1;
Yappx_CS1 =YappxN_CS1.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.3: CS2 -----------------------------------------------------------------------
YappxN_CS2=Phi_CS2*w_CS2;
Yappx_CS2 =YappxN_CS2.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%--------------- Step 9.4: Gassian RBF----------------------------------------------------------------
YappxN=Phi*w;
Yappx =YappxN.*(max(Y_trn)-min(Y_trn))+min(Y_trn);

%*****************************************************************************************************
%--------------- Step 10: Computing the training error with MSE----------------------------
%*****************************************************************************************************
Err_Trn_CS_RBF1=(1/N_tr)*sum((Y_trn-Yappx_CS_RBF1).^2);
Err_Trn_CS1=(1/N_tr)*sum((Y_trn-Yappx_CS1).^2);
Err_Trn_CS2=(1/N_tr)*sum((Y_trn-Yappx_CS2).^2);
Err_Trn=(1/N_tr)*sum((Y_trn-Yappx).^2);

T_Trn =table(Err_Trn_CS_RBF1,Err_Trn_CS1,Err_Trn_CS2,Err_Trn)

%*****************************************************************************************************
%--------------- Step 11: Generate validation dataset ------------------------------------------------
%*****************************************************************************************************
%---//// Generate a validation dataset of 1000 data points in ////---------
for i=1:1000
X_vld(i)=2*pi*((i-1)/1000);
End

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Y_vld=awgn(sin(X_vld),0.5,'measured')';
%Y_vld with Gaussian noise

for i=1:1000
for j=1:1000
XY_vld(i,1)=X_vld(i);
XY_vld(j,2)=Y_vld(j);
end
end
XY_vld;

x2max = max(X_vld);
X_vldN=(X_vld-min(X_vld))./(x2max);
%******************************************************************************************************
%--------------- Step 12: Construct the interpolation matrix-----------------------------------
%---------------from X_vld to Xctr ------------------------------------------------------------------------
%******************************************************************************************************
%--------------- Step 12.1: CS_RBF1 ----------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS_RBF1(i,j)=abs(X_vldN(1,i)-Xctr_CS_RBF1(1,j));
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-
   (4/3)*r2_CS_RBF1(i,j).^3+2*(r2_CS_RBF1(i,j).^2).*log(r2_CS_RBF1(i,j));
if  r2_CS_RBF1(i,j) == 0
Phi2_CS_RBF1(i,j)=(1/3)+r2_CS_RBF1(i,j).^2-(4/3)*r2_CS_RBF1(i,j).^3;
end
end
end

%--------------- Step 12.2: CS1 ----------------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS1(i,j)=abs(X_vldN(1,i)-Xctr_CS1(1,j));
Phi2_CS1(i,j)=(112/45)*r2_CS1(i,j).^(9/2)+(16/3)*r2_CS1(i,j).^(7/2)-7*r2_CS1(i,j).^4-
   (14/15)*r2_CS1(i,j).^2+(1/9);
end
end

%--------------- Step 12.3: CS2 ----------------------------------------------------------------------
for i=1:1000
for j=1:M
r2_CS2(i,j)=abs(X_vldN(1,i)-Xctr_CS2(1,j));
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4-
   (4/3)*r2_CS2(i,j).^3.*log(r2_CS2(i,j));
if  r2_CS2(i,j) == 0
Phi2_CS2(i,j)=(1/18)-r2_CS2(i,j).^2+(4/9)*r2_CS2(i,j).^3+0.5*r2_CS2(i,j).^4;
end
end
end

%--------------- Step 12.4: Gaussian RBF----------------------------------------------------------
for i=1:1000
for j=1:M
r2(i,j)=abs(X_vldN(1,i)-Xctr(1,j));
Phi2(i,j)=exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%***************************************************************************************************
%--------------- Step 13:Construct the Y_vldapxN from Phi2 and weight 'w'
%***************************************************************************************************
%--------------- Step 13.1:CS_RBF1 ------------------------------------------------------------------
Y_vldapxN_CS_RBF1=Phi2_CS_RBF1*w_CS_RBF1;
Y_vldapx_CS_RBF1 = Y_vldapxN_CS_RBF1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.2:CS1 ----------------------------------------------------------------------
Y_vldapxN_CS1=Phi2_CS1*w_CS1;
Y_vldapx_CS1 = Y_vldapxN_CS1.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.3:CS2 ----------------------------------------------------------------------
Y_vldapxN_CS2=Phi2_CS2*w_CS2;
Y_vldapx_CS2 = Y_vldapxN_CS2.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%--------------- Step 13.4:Gassian RBF----------------------------------------------------------------
Y_vldapxN=Phi2*w;
Y_vldapx = Y_vldapxN.*(max(Y_vld)-min(Y_vld))+min(Y_vld);

%***************************************************************************************************
%--------------- Step 14:Computing the validation error with MSE -------------------
%***************************************************************************************************
Err_Vld_CS_RBF1=(1/1000)*sum((Y_vld-Y_vldapx_CS_RBF1).^2);
Err_Vld_CS1=(1/1000)*sum((Y_vld-Y_vldapx_CS1).^2);
Err_Vld_CS2=(1/1000)*sum((Y_vld-Y_vldapx_CS2).^2);
Err_Vld =(1/1000)*sum((Y_vld-Y_vldapx).^2);

T_Vld = table(Err_Vld_CS_RBF1,Err_Vld_CS1,Err_Vld_CS2,Err_Vld)

Y1 = sin(X_trn);

for i=1:100
Y3(i) = 20;
End

%---- Step 12 : Results Plotting --------------------------------------------------------------------
figure;
hold on
plot(X_trn,Y_trn,'ok','LineWidth',1);
plot(X_trn,Y3,'ok','LineWidth',1);
plot(X_trn,Y1,'-k','LineWidth',1);
xlabel('x', 'FontSize',14);
ylabel('Y=sin(X)', 'FontSize',14);
grid on

figure;
hold on
plot(X_trn,Y_trn,'ok','markersize',3);
plot(X_trn,Yappx_CS_RBF1,'-k','LineWidth',2);
plot(X_trn,Yappx_CS1,'-r','LineWidth',2);
plot(X_trn,Yappx_CS2,'-m','LineWidth',2);
plot(X_trn,Yappx,'-g','LineWidth',2);
title('Training data')
xlabel('X', 'FontSize',14);
ylabel('Y=sin(X)', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
'SouthEast')
grid on
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

figure;
hold on
plot(X_vld,Y_vld,'ok','markersize',3);
plot(X_vld,Y_vldapx_CS_RBF1,'-k','LineWidth',2);
plot(X_vld,Y_vldapx_CS1,'-r','LineWidth',2);
plot(X_vld,Y_vldapx_CS2,'-m','LineWidth',2);
plot(X_vld,Y_vldapx,'-g','LineWidth',2);
title('Validation data')
xlabel('X', 'FontSize',14);
ylabel('Y=sin(X)', 'FontSize',14);
legend('Exact', 'CS RBF1', 'CS1', 'CS2', 'Gaussian' ,'Location',
 'SouthEast')
%------------------------------ THIS IS THE END ------------------------------
```

A.2 Main Code 2: Experiment 3 Selected Main Codes

```
clear all
close all

%**************************************************************************
%----------- Step 1:Generate training and testing dataset -----------
%**************************************************************************
%----------- Step 1.1:Generate all dataset ------------------------------
x=linspace(0,1,61);
y=linspace(0,1,61);
L=length(x);

for i=1:L
for j=1:L
XO(j+(i-1)*L)=x(i);
YO(j+(i-1)*L)=y(j);
end
end

ZO=awgn(0.75.*exp((((9.*XO-2).^2)+((9.*YO-2).^2))./(-4)) +...
0.75.*exp(((9.*XO+1).^2)./(-49)-(9.*YO+1)./10) +...
0.5.*exp((((9.*XO-7).^2)+((9.*YO-3).^2))./(-4))-...
0.2.*exp( -(9.*XO-4).^2 -(9.*YO-7).^2 ),15,'measured')';
N = length(ZO);

for i=1:N
XYZO(i,1)=XO(i);
XYZO(i,2)=YO(i);
XYZO(i,3)=ZO(i);
end
XYZO;

%----------- Step 1.2:Generate training dataset via choosing-----------
%----------- training from all dataset---------------------------------
xN=linspace(0,1,11);
yN=linspace(0,1,11);
LN=length(xN);

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=1:LN
for j=1:LN
X_tr(j+(i-1)*LN)=xN(i);
Y_tr(j+(i-1)*LN)=yN(j);
end
end
N_tr = LN*LN;
%--------------- Step 1.3: Choose training from all dataset ------------------------
for i=1:N_tr
for j=1:N
if X_tr(i) == XYZO(j,1)
if Y_tr(i) == XYZO(j,2)
Z_tr(i) = XYZO(j,3);
end
end
end
end

for i=1:N_tr
XYZ_tr(i,1)=X_tr(i);
XYZ_tr(i,2)=Y_tr(i);
XYZ_tr(i,3)=Z_tr(i);
End
XYZ_tr;

%--------------- Step 1.4: Select testing dataset other-----------------------------------
%--------------- than the training dataset of the entire dataset ------------
XYZ_vld = setdiff(XYZO, XYZ_tr, 'rows');

N_vld = N-N_tr;

for i=1:N_vld
X_vld(i)= XYZ_vld(i,1);
Y_vld(i)= XYZ_vld(i,2);
Z_vld(i)= XYZ_vld(i,3);
End

%*********************************************************************************************************
%--------------- Step 2: Choose a method to find the shape parameter ------
%*********************************************************************************************************
%--------------- Step 2.1: Choose Hardy method ---------------------------
for i=1:N_tr
for j=1:N_tr
H(i,j)=((X_tr(i)-X_tr(j))^2+(Y_tr(i)-Y_tr(j))^2)^(1/2);
if i==j
H(i,j)= 999999;
end
end
end

md = min(H)
d = sum(md)/(N_tr)
shp = 0.815*d;

%--------------- Step 2.2: Choose Franke method ------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
% for i=1:N_tr
%  for j=1:N_tr
%   F(i,j)=((X_tr(i)-X_tr(j))^2+(Y_tr(i)-Y_tr(j))^2)^(1/2);
%  end
% end
%
% fD = max(F)
     % D = max(fD)
     % shp = (1.25*D)/sqrt(N_tr);

     %-------------- Step 2.3: Choose Carlson method ----------------------------
     % xN = (X_tr - min(X_tr))/(max(X_tr)-min(X_tr));
     % yN = (Y_tr - min(Y_tr))/(max(Y_tr)-min(Y_tr))
     % zN = (Z_tr - min(Z_tr))/(max(Z_tr)-min(Z_tr))
     %
     % for i=1:N_tr
     %  for j=1:N_tr
     %   if j==1
     %    P(i,j)=1;
     %   end
     %   if j==2
     %              P(i,j)=xN(i);
     %   end
     %   if j==3
     %    P(i,j)=yN(i);
     %   end
     %   if j==4
     %    P(i,j)=xN(i)^2;
     %   end
     %   if j==5
     %    P(i,j)=xN(i)*yN(i);
     %   end
     %   if j==6
     %    P(i,j)=yN(i)^2;
     %   end
     %  end
     % end
     %
     %% c = inv(P' * P)*(P' * zN');
     % c = pinv(P)* zN';
     % zApp = P*c;
     % V = sum((zN-zApp').^2 / N_tr)
     % shp = 1/(1+(120*V));

     %---- Step 5 Using RC Algorithm ----------------------------------------------
     %---/// Step 5.1 Construct the G matrix ///-----------------------------------
     %****************************************************************************************
     %-------------- Step 3: Proceed with RC-Algorithm ---------------------------
     %****************************************************************************************
     for i=1:N_tr
     for j=1:N_tr
     r(i,j)=sqrt((X_tr(i)-X_tr(j)).^2 + (Y_tr(i)-Y_tr(j)).^2);
     G(i,j)=exp(-(r(i,j)^2)/(2*(shp^2)));
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
end

delta = 0.01/100;

[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

for i=1:N_tr
Sii = S(i,i);
if Sii <= S1 ;
M = i-1
break
else
M=N_tr;
break
end
i = i+1 ;
end
V1 = V;
V11=V1(1:M,1:M) ;
V21=V1(M+1:N_tr,1:M) ;
V2 = [V11' V21'];
[Q,R,P] = qr(V2);
XY_tr = [X_tr',Y_tr'];
mu = XY_tr'*P;
Xctr = mu(:,1:M)

%***************************************************************************
%------------ Step 4: Construct interpolation matrix -------------------
%***************************************************************************
for i=1:N_tr
for j=1:M
r1(i,j)=sqrt((X_tr(1,i)-Xctr(1,j)).^2  + (Y_tr(1,i)-Xctr(2,j)).^2 );
Phi(i,j)= exp(-(r1(i,j)^2)/(2*(shp^2)));
end
end

%***************************************************************************
%------------ Step 5: : Computing the weight 'w' --------------------
%***************************************************************************
Z_tr = Z_tr';
w=pinv(Phi)*Z_tr;

%***************************************************************************
%------------ Step 6: Computing the approximation of -------------------
%------------training dataset ------------------------------------------
%***************************************************************************
Zappx=Phi*w;

%***************************************************************************
%------------ Step 7: Computing the training error----------------------
%***************************************************************************
Err_Trn = (1/N_tr)*sum((Z_tr-Zappx).^2)
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

%**************************************************************************************************
%--------------- Step 8:Construct the interpolation matrix -------------
%---------------from Xvld to Xctr ----------------------------------------------------------------
%**************************************************************************************************
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2+ (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)= exp(-(r2(i,j)^2)/(2*(shp^2)));
end
end

%**************************************************************************************************
%--------------- Step 9:Construct the Zvldapx from Phi2---------------------
%---------------and weight 'w' -----------------------------------------------------------------
%**************************************************************************************************
Zvldapx=Phi2*w;

%**************************************************************************************************
%--------------- Step 10:Computing the test error with MSE -------------
%**************************************************************************************************
Z_vld = Z_vld';
Err_Vld =(1/N_vld)*sum((Z_vld-Zvldapx).^2)

%**************************************************************************************************
%--------------- Step 11:Results Plotting --------------------------------------------
%**************************************************************************************************
for i=1:N
Solu(i,1)=XYZO(i,1);
Solu(i,2)=XYZO(i,2);
Solu(i,3)=XYZO(i,3);
End

for i=1 :N_tr
Solu_tr(i,1)=XYZ_tr(i,1);
Solu_tr(i,2)=XYZ_tr(i,2);
Solu_tr(i,3)=XYZ_tr(i,3);
Solu_tr(i,4)=Zappx(i,1);
End

for i=1 :N_vld
Solu_vld(i,1)=XYZ_vld(i,1);
Solu_vld(i,2)=XYZ_vld(i,2);
Solu_vld(i,3)=XYZ_vld(i,3);
Solu_vld(i,4)=Zvldapx(i,1);
End

figure;
hold on
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,3),'ro','markersize',3,'LineWidth',2)
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,4),'ko','markersize',4,'LineWidth',2)
title('Data set')
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
        xlabel('Input Variable (X)', 'FontSize',14);
        ylabel('Input Variable (Y)', 'FontSize',14);
        zlabel('Output Variable (Z)', 'FontSize',14);
        legend({'Approximate','Exact'});
        hold off
        grid on

        figure;
        hold on
        plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,3),'ro','markersize',3,'LineWidth',2)
        plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,4),'ko','markersize',4,'LineWidth',2)
        title('Data set')
        xlabel('Input Variable (X)', 'FontSize',14);
        ylabel('Input Variable (Y)', 'FontSize',14);
        zlabel('Output Variable (Z)', 'FontSize',14);
        legend({'Approximate','Exact'});
        hold off
        grid on
        %-------------------------------------------- THIS IS THE END ------------------------------
```

A.3 Main Code 3: The Main Numerical Experiment

A.3.1 Experiment 1: Franke's function

```
clc
clear all
close all

%**********************************************************************************************
%-------------- Step 1:Generate training and testing dataset  ---------------------------
%**********************************************************************************************
%-------------- Step 1.1: Define the input data -------------------------------------------
x=linspace(0,1,50);
y=linspace(0,1,50);
N_O=length(x);
for i=1:N_O
for j=1:N_O
X_trn(j+(i-1)*N_O)=x(i);
Y_trn(j+(i-1)*N_O)=y(j);
end
end
N_trn=length(X_trn);
XY_trn=[X_trn' Y_trn'];

%-------------- Step 1.2: Define the test function with noise ---------------------------
%-------------- 1.2.1:Define performance at snr=30------------------------------------------------
Z_trn = awgn(0.75.*exp((((9.*X_trn-2).^2)+((9.*Y_trn-2).^2))/(-4)) + 0.75.*exp(((9.*X_trn+1).^2)./(-49)-(9.*Y_trn+1)./10) + 0.5.*exp((((9.*X_trn-7).^2)+((9.*Y_trn-3).^2))./(-4)) - 0.2.*exp( -(9.*X_trn-4).^2 -(9.*Y_trn-7).^2,30,'measured')';
%-------------- 1.2.2:Define performance at snr=15------------------------------------------------
Z_trn = awgn(0.75.*exp((((9.*X_trn-2).^2)+((9.*Y_trn-2).^2))/(-4)) + 0.75.*exp(((9.*X_trn+1).^2)./(-49)-(9.*Y_trn+1)./10) + 0.5.*exp((((9.*X_trn-7).^2)+((9.*Y_trn-3).^2))./(-4)) - 0.2.*exp( -(9.*X_trn-4).^2 -(9.*Y_trn-7).^2,15,'measured')';
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

XYZ_trn=[XY_trn Z_trn];

%-------------- Step 1.3: Define the testing dataset by using ---------------------------
%-------------- the 3 different dataset which is 10000(100x100), -------------------
%-------------- 20164(142x142) and 30276(174x174) in this step ---------------------
%-------------- 1.3.1 For 10000(100x100) nodes---------------------------------------------------------
X_t=linspace(0,1,100);
Y_t=linspace(0,1,100);
N_t=length(X_t);

%-------------- 1.3.2 For 20164(142x142) nodes-------------------------------------------------------
X_t=linspace(0,1,142);
Y_t=linspace(0,1,142);
N_t=length(X_t);

%-------------- 1.3.3 For 30276(174x174) nodes-------------------------------------------------------
X_t=linspace(0,1,174);
Y_t=linspace(0,1,174);
N_t=length(X_t);

for i=1:N_t
for j=1:N_t
XY_vld(j+(i-1)*N_t,1)=X_t(i);
XY_vld(j+(i-1)*N_t,2)=Y_t(j);
end
end
N_vld=length(XY_vld);

for i=1:N_vld
X_vld(i)= XY_vld(i,1);
Y_vld(i)= XY_vld(i,2);
End

Z_vld = 0.75.*exp((((9.*X_vld-2).^2)+((9.*Y_vld-2).^2))/(-4)) + 0.75.*exp(((9.*X_vld+1).^2)./(-49)-
  (9.*Y_vld+1)./10) + 0.5.*exp((((9.*X_vld-7).^2)+((9.*Y_vld-3).^2))./(-4)) - 0.2.*exp( -(9.*X_vld-4).^2 -
  (9.*Y_vld-7).^2 );

XYZ_vld=[XY_vld Z_vld'];

tic;
%***********************************************************************************************
%-------------- Step 2: Construct the interpolation matrix -------------------------------
%***********************************************************************************************
%-------------- 2.1 For MQ + Carlson RBF ----------------------------------------------------------
xN =(X_trn -min(X_trn))/(max(X_trn)-min(X_trn));
yN =(Y_trn -min(Y_trn))/(max(Y_trn)-min(Y_trn));
zN =(Z_trn -min(Z_trn))/(max(Z_trn)-min(Z_trn));
zN =zN';

for i=1:N_trn
for j=1:N_trn
if j==1
P(i,j)=1;
end
if j==2
P(i,j)=xN(i);
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
if j==3
P(i,j)=yN(i);
end
if j==4
P(i,j)=xN(i)^2;
end
if j==5
P(i,j)=xN(i)*yN(i);
end
if j==6
P(i,j)=yN(i)^2;
end
end
end

c = pinv(P)*zN';
zApp = P*c;
V = sum((zN-zApp').^2 / N_trn);
shp = 1/(1+(120*V));

for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=sqrt(r(i,j)^2 +shp^2);
end
end
%---------------- 2.2 For PS RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)= r(i,j)^((2*k)-1);
Else
G(i,j)=0;
end
end
end

%---------------- 2.3 For TPS RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1 & r(i,j)~= 0
G(i,j)=(r(i,j)^2)*log(r(i,j));
Else
G(i,j)=0;
end
end
end

%---------------- 2.4 For WU1 RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^7)*((5*r(i,j)^6)+(35*r(i,j)^5)+(101*r(i,j)^4)+(147*r(i,j)^3)+(101*r(i,j)^2)+35*r(i,j)+5);
Else
```

```
G(i,j)=0;
end
end
end

%-------------- 2.5 For  WU2 RBF--------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^6)*((6*r(i,j)^5)+(30*r(i,j)^4)+(72*r(i,j)^3)+(82*r(i,j)^2)+36*r(i,j)+6);
Else
G(i,j)=0;
end
end
end

%-------------- 2.6 For  WU3 RBF--------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^5)*((5*r(i,j)^4)+(25*r(i,j)^3)+(48*r(i,j)^2)+40*r(i,j)+8);
Else
G(i,j)=0;
end
end
end

%-------------- 2.7 For  WU4 RBF--------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^4)*((5*r(i,j)^3)+(20*r(i,j)^2)+29*r(i,j)+16);
Else
G(i,j)=0;
end
end
end

%-------------- 2.8 For  WL1 RBF--------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j)));
Else
G(i,j)=0;
end
end
end

%-------------- 2.9 For  WL2 RBF--------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^3)*(3*r(i,j)+1);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Else
G(i,j)=0;
end
end
end

%------------- 2.10 For WL3 RBF-----------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^5)*((8*r(i,j)^2)+5*r(i,j)+1);
Else
G(i,j)=0;
end
end
end

%------------- 2.11 For WL4 RBF-----------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^2);
Else
G(i,j)=0;
end
end
end
%------------- 2.12 For WL5 RBF-----------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^4)*(4*r(i,j)+1);
Else
G(i,j)=0;
end
end
end
%------------- 2.13 For WL6 RBF-----------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^6)*((35*r(i,j)^2)+18*r(i,j)+3);
Else
G(i,j)=0;
end
end
end

%------------- 2.14 For WL7 RBF-----------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^8)*((32*r(i,j)^3)+(25*r(i,j)^2)+8*r(i,j)+1);
Else
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
G(i,j)=0;
end
end
end

%-------------- 2.15 For  BUH1 RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=(1/6)-(2*r(i,j).^2)+(16/3)*r(i,j).^3 -(7/2)*r(i,j).^4 + 2*(r(i,j).^4).*log(r(i,j));
if  r(i,j) == 0
G(i,j)=(1/6)-(2*r(i,j).^2)+(16/3)*r(i,j).^3 -(7/2)*r(i,j).^4;
end
end
end

%-------------- 2.16 For  BUH2 RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
Else
G(i,j)=0;
end
end
end

%-------------- 2.17 For  BUH3 RBF------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
if  r(i,j) == 0
G(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%*********************************************************************************************
%-------------- Step 3: Using SVD to get the number of centres-----------------
%*********************************************************************************************
delta = 0.01/100;
[U,S,V] = svd(G);
S11 = S(1,1);
S1 = S11*(delta);

for i=1:N_trn
Sii = S(i,i);
if Sii <= S1 ;
M = i-1
break
end
i = i+1 ;
end

%*********************************************************************************************
%-------------- Step 4: Using QR to find the centres of basis function  --
%---------------------- and the location of centres ------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%************************************************************************************************
V1  = V;
V11=V1(1:M,1:M);             %partition matrix V
V21=V1(M+1:N_trn,1:M);
V2  =[V11' V21'];
[Q,R,P] = qr(V2);
XY_tr  =[X_trn',Y_trn'];
mu  = XY_trn'*P;
Xctr  = mu(:,1:M);

%************************************************************************************************
%---------- Step 5: Construct the new interpolation matrix    ---------------
%************************************************************************************************
for i=1:N_trn
for j=1:M
r1(i,j)=sqrt((X_trn(1,i)-Xctr(1,j)).^2  + (Y_trn(1,i)-Xctr(2,j)).^2 );
if 0<=r1(i,j) & r1(i,j)<=1
Phi(i,j)=((1-r1(i,j))^4)*((5*r1(i,j)^3)+(20*r1(i,j)^2)+29*r1(i,j)+16);
Else
Phi(i,j)=0;
end
end
end

%************************************************************************************************
%---------- Step 6: Computing the weight 'w' ----------------------------------------
%************************************************************************************************
w=pinv(Phi)*Z_trn;

%************************************************************************************************
%---------- Step 7: Computing the approximation of training dataset --
%************************************************************************************************
Zappx=Phi*w;

%************************************************************************************************
%---------- Step 8: Computing the training error-------------------------------------
%************************************************************************************************
Err_max_trn  = max(abs(Z_trn-Zappx))
Err_RMSE_trn  =sqrt((1/N_trn)*sum((Z_trn-Zappx).^2))

%************************************************************************************************
%---------- Step 9: Construct the interpolation matrix -----------------
%----------from Xvld to Xctr ------------------------------------------------------
%************************************************************************************************
%---------- 9.1 For  MQ + Carlson RBF-------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)= sqrt(r2(i,j)^2 + shp^2);
end
end

%---------- 9.2 For  PS RBF--------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Phi2(i,j)=r2(i,j)^((2*k)-1);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.3 For  TPS RBF--------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1  & r2(i,j) ~= 0
Phi2(i,j)=(r2(i,j)^2)*log(r2(i,j));
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.4 For  WU1 RBF--------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^7)*((5*r2(i,j)^6)+(35*r2(i,j)^5)+(101*r2(i,j)^4)+(147*r2(i,j)^3)+(101*r2(i,j)^2)+35*r2(i,j)+5);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.5 For  WU2 RBF--------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^6)*((6*r2(i,j)^5)+(30*r2(i,j)^4)+(72*r2(i,j)^3)+(82*r2(i,j)^2)+36*r2(i,j)+6);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.6 For  WU3 RBF--------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^5)*((5*r2(i,j)^4)+(25*r2(i,j)^3)+(48*r2(i,j)^2)+40*r2(i,j)+8);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.7 For  WU4 RBF--------------------------------------------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^4)*((5*r2(i,j)^3)+(20*r2(i,j)^2)+29*r2(i,j)+16);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.8 For WL1 RBF--------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j)));
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.9 For WL2 RBF--------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^3)*(3*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.10 For WL3 RBF-------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^5)*((8*r2(i,j)^2)+5*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.11 For WL4 RBF-------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^2);
else
Phi2(i,j)=0;
end
end
end

```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
%--------------- 9.12 For WL5 RBF------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^4)*(4*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.13 For WL6 RBF------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^6)*((35*r2(i,j)^2)+18*r2(i,j)+3);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.14 For WL7 RBF------------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^8)*((32*r2(i,j)^3)+(25*r2(i,j)^2)+8*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%--------------- 9.15 For BUH1 RBF-----------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)=(1/6)-(2*r2(i,j).^2)+(16/3)*r2(i,j).^3 -(7/2)*r2(i,j).^4 + 2*(r2(i,j).^4).*log(r2(i,j));
if  r2(i,j) == 0
Phi2(i,j)=(1/6)-(2*r2(i,j).^2)+(16/3)*r2(i,j).^3 -(7/2)*r2(i,j).^4;
end
end
end

%--------------- 9.16 For BUH2 RBF-----------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=(112/45)*r2(i,j).^(9/2)+(16/3)*r2(i,j).^(7/2)-7*r2(i,j).^4-(14/15)*r2(i,j).^2+(1/9);
else
Phi2(i,j)=0;
end
end
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

%--------------- 9.17 For BUH3 RBF-------------------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2   + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)=(1/18)-r2(i,j).^2+(4/9)*r2(i,j).^3+0.5*r2(i,j).^4-(4/3)*r2(i,j).^3.*log(r2(i,j));
if  r2(i,j) == 0
Phi2(i,j)=(1/18)-r2(i,j).^2+(4/9)*r2(i,j).^3+0.5*r2(i,j).^4;
end
end
end

%****************************************************************************************************
%--------------- Step 10:Construct the Zvldapx from Phi2 and weight 'w' --
%****************************************************************************************************
Zvldapx=Phi2*w;

%****************************************************************************************************
%--------------- Step 11:Computing the test error with MSE ------------------------------------
%****************************************************************************************************
Z_vld = Z_vld';
Err_max_vld = max(abs(Z_vld-Zvldapx))
Err_RMSE_vld =sqrt((1/N_vld)*sum((Z_vld-Zvldapx).^2))
toc;
%****************************************************************************************************
%--------------- Step 12:Find the condition number------------------------------------------------
%****************************************************************************************************
con_G=cond(G)
con_Phi = cond(Phi)

%****************************************************************************************************
%--------------- Step 13:Results Plotting ---------------------------------------------------------
%****************************************************************************************************
for i=1 :N_trn
Solu_tr(i,1)=XYZ_trn(i,1);
Solu_tr(i,2)=XYZ_trn(i,2);
Solu_tr(i,3)=XYZ_trn(i,3);
Solu_tr(i,4)=Zappx(i,1);
End

for i=1 :N_vld
Solu_vld(i,1)=XYZ_vld(i,1);
Solu_vld(i,2)=XYZ_vld(i,2);
Solu_vld(i,3)=XYZ_vld(i,3);
Solu_vld(i,4)=Zvldapx(i,1);
End

figure;
hold on
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,3),'ro','markersize',3,'LineWidth',
2)
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,4),'ko','markersize',4,'LineWidth',
2)
title('Data set')
xlabel('Input Variable (X)', 'FontSize',14);
ylabel('Input Variable (Y)', 'FontSize',14);
zlabel('Output Variable (Z)', 'FontSize',14);
legend({'Exact','Approximate'});
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
hold off
grid on

figure;
hold on
plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,3),'ro','markersize',3,'LineWidt
h',2)
plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,4),'ko','markersize',4,'LineWidt
h',2)
title('Data set')
xlabel('Input Variable (X)', 'FontSize',14);
ylabel('Input Variable (Y)', 'FontSize',14);
zlabel('Output Variable (Z)', 'FontSize',14);
legend({'Exact','Approximate'});
hold off
grid on
%------------------------------------------------- THIS IS THE END ------------------------------------------------------------------------
```

A.3.2 Experiment 2: F7

```
clc
clear all
close all

%*****************************************************************************************************************************************
%--------------- Step 1: Generate training and testing dataset ---------------------------
%*****************************************************************************************************************************************
%--------------- Step 1.1: Define the input data ---------------------------------------------------------------
x=linspace(0,1,50);
y=linspace(0,1,50);
N_O=length(x);
for i=1:N_O
for j=1:N_O
X_trn(j+(i-1)*N_O)=x(i);
Y_trn(j+(i-1)*N_O)=y(j);
end
end
N_trn=length(X_trn);
XY_trn=[X_trn' Y_trn'];

%--------------- Step 1.2: Define the test function with noise ---------------------------
%--------------- 1.2.1: Define performance at snr=30------------------------------------------------------------
Z_trn = awgn(2.*cos(10.* X_trn).*sin(10.*Y_trn)+sin(10.*
  X_trn.*Y_trn,30,'measured')';

%--------------- 1.2.2: Define performance at snr=15------------------------------------------------------------
Z_trn = awgn(2.*cos(10.* X_trn).*sin(10.*Y_trn)+sin(10.*
  X_trn.*Y_trn,15,'measured')';

XYZ_trn=[XY_trn Z_trn];

%--------------- Step 1.3: Define the testing dataset by using ---------------------------
%--------------- the 3 different dataset which is 10000(100x100), ---------------------
%--------------- 20164(142x142) and 30276(174x174) in this step ---------------------
%--------------- 1.3.1 For 10000(100x100) nodes-----------------------------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
X_t=linspace(0,1,100);
Y_t=linspace(0,1,100);
N_t=length(X_t);

%--------------- 1.3.2 For 20164(142x142) nodes------------------------------------------------------------
X_t=linspace(0,1,142);
Y_t=linspace(0,1,142);
N_t=length(X_t);

%--------------- 1.3.3 For 30276(174x174) nodes------------------------------------------------------------
X_t=linspace(0,1,174);
Y_t=linspace(0,1,174);
N_t=length(X_t);

for i=1:N_t
for j=1:N_t
XY_vld(j+(i-1)*N_t,1)=X_t(i);
XY_vld(j+(i-1)*N_t,2)=Y_t(j);
end
end
N_vld=length(XY_vld);

for i=1:N_vld
X_vld(i)= XY_vld(i,1);
Y_vld(i)= XY_vld(i,2);
End

Z_vld = 2.*cos(10.*X_vld).*sin(10.*Y_vld)+sin(10.*X_vld.*Y_vld);


XYZ_vld=[XY_vld Z_vld'];

tic;
%*****************************************************************************************************
%--------------- Step 2: Construct the interpolation matrix ------------------------------------------
%*****************************************************************************************************
%--------------- 2.1 For MQ + Carlson RBF--------------------------------------------------------------
xN = (X_trn - min(X_trn))/(max(X_trn)-min(X_trn));
yN = (Y_trn - min(Y_trn))/(max(Y_trn)-min(Y_trn));
zN = (Z_trn - min(Z_trn))/(max(Z_trn)-min(Z_trn));
zN = zN';

for i=1:N_trn
for j=1:N_trn
if j==1
P(i,j)=1;
end
if j==2
P(i,j)=xN(i);
end
if j==3
P(i,j)=yN(i);
end
if j==4
P(i,j)=xN(i)^2;
end
if j==5
P(i,j)=xN(i)*yN(i);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end
if j==6
P(i,j)=yN(i)^2;
end
end
end

c = pinv(P)*zN';
zApp = P*c;
V = sum((zN-zApp').^2 /N_trn);
shp = 1/(1+(120*V));

for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=sqrt(r(i,j)^2 +shp^2);
end
end
%-------------- 2.2 For PS RBF--------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)= r(i,j)^((2*k)-1);
Else
G(i,j)=0;
end
end
end

%-------------- 2.3 For TPS RBF-------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1 & r(i,j)~=0
G(i,j)=(r(i,j)^2)*log(r(i,j));
Else
G(i,j)=0;
end
end
end

%-------------- 2.4 For WU1 RBF-------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j)& r(i,j)<=1
G(i,j)=((1-r(i,j))^7)*((5*r(i,j)^6)+(35*r(i,j)^5)+(101*r(i,j)^4)+(147*r(i,j)^3)+(101*r(i,j)^2)+35*r(i,j)+5);
Else
G(i,j)=0;
end
end
end

%-------------- 2.5 For WU2 RBF-------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^6)*((6*r(i,j)^5)+(30*r(i,j)^4)+(72*r(i,j)^3)+(82*r(i,j)^2)+36*r(i,j)+6);
Else
G(i,j)=0;
end
end
end

%--------------- 2.6 For WU3 RBF--------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^5)*((5*r(i,j)^4)+(25*r(i,j)^3)+(48*r(i,j)^2)+40*r(i,j)+8);
Else
G(i,j)=0;
end
end
end

%--------------- 2.7 For WU4 RBF--------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^4)*((5*r(i,j)^3)+(20*r(i,j)^2)+29*r(i,j)+16);
Else
G(i,j)=0;
end
end
end

%--------------- 2.8 For WL1 RBF--------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j)));
Else
G(i,j)=0;
end
end
end

%--------------- 2.9 For WL2 RBF--------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^3)*(3*r(i,j)+1);
Else
G(i,j)=0;
end
end
end

%--------------- 2.10 For WL3 RBF-------------------------------------------------------------------------------
for i=1:N_trn
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^5)*((8*r(i,j)^2)+5*r(i,j)+1);
Else
G(i,j)=0;
end
end
end

%--------------- 2.11 For WL4 RBF-----------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^2);
Else
G(i,j)=0;
end
end
end

%--------------- 2.12 For WL5 RBF-----------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^4)*(4*r(i,j)+1);
Else
G(i,j)=0;
end
end
end

%--------------- 2.13 For WL6 RBF-----------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^6)*((35*r(i,j)^2)+18*r(i,j)+3);
Else
G(i,j)=0;
end
end
end

%--------------- 2.14 For WL7 RBF-----------------------------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=((1-r(i,j))^8)*((32*r(i,j)^3)+(25*r(i,j)^2)+8*r(i,j)+1);
Else
G(i,j)=0;
end
end
end

%--------------- 2.15 For BUH1 RBF---------------------------------------------------------------------------------------
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=(1/6)-(2*r(i,j).^2)+(16/3)*r(i,j).^3 -(7/2)*r(i,j).^4 + 2*(r(i,j).^4).*log(r(i,j));
if  r(i,j) == 0
G(i,j)=(1/6)-(2*r(i,j).^2)+(16/3)*r(i,j).^3 -(7/2)*r(i,j).^4;
end
end
end

%------------- 2.16 For BUH2 RBF---------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
if 0<=r(i,j) & r(i,j)<=1
G(i,j)=(112/45)*r(i,j).^(9/2)+(16/3)*r(i,j).^(7/2)-7*r(i,j).^4-(14/15)*r(i,j).^2+(1/9);
Else
G(i,j)=0;
end
end
end

%------------- 2.17 For BUH3 RBF---------------------------------------------------------------------
for i=1:N_trn
for j=1:N_trn
r(i,j)=sqrt((X_trn(i)-X_trn(j)).^2 +(Y_trn(i)-Y_trn(j)).^2);
G(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4-(4/3)*r(i,j).^3.*log(r(i,j));
if  r(i,j) == 0
G(i,j)=(1/18)-r(i,j).^2+(4/9)*r(i,j).^3+0.5*r(i,j).^4;
end
end
end

%*****************************************************************************************
%------------- Step 3:Using SVD to get the number of centres-------------------------
%*****************************************************************************************
delta  = 0.01/100;
[U,S,V] = svd(G);
S11  = S(1,1);
S1  = S11*(delta);

for i=1:N_trn
Sii  = S(i,i);
if Sii  <= S1  ;
M  = i-1
break
end
i  = i+1  ;
end

%*****************************************************************************************
%------------- Step 4:Using QR to find the centres of basis function  --
%-------------------- and the location of centres -----------------------------------------
%*****************************************************************************************
V1  = V;
V11=V1(1:M,1:M) ;             %partition matrix V
V21=V1(M+1:N_trn,1:M) ;
V2  = [V11'  V21'];
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
[Q,R,P] = qr(V2);
XY_tr = [X_trn',Y_trn'];
mu = XY_trn'*P;
Xctr = mu(:,1:M);

%******************************************************************************************
%-------------- Step 5: Construct the new interpolation matrix---------------------------
%******************************************************************************************
for i=1:N_trn
for j=1:M
r1(i,j)=sqrt((X_trn(1,i)-Xctr(1,j)).^2  + (Y_trn(1,i)-Xctr(2,j)).^2 );
if 0<=r1(i,j) & r1(i,j)<=1
Phi(i,j)=((1-r1(i,j))^4)*((5*r1(i,j)^3)+(20*r1(i,j)^2)+29*r1(i,j)+16);
Else
Phi(i,j)=0;
end
end
end

%******************************************************************************************
%-------------- Step 6: Computing the weight 'w' -----------------------------------------
%******************************************************************************************
w=pinv(Phi)*Z_trn;

%******************************************************************************************
%-------------- Step 7: Computing the approximation of training dataset --
%******************************************************************************************
Zappx=Phi*w;

%******************************************************************************************
%-------------- Step 8: Computing the training error------------------------------------------
%******************************************************************************************
Err_max_trn = max(abs(Z_trn-Zappx))
Err_RMSE_trn =sqrt((1/N_trn)*sum((Z_trn-Zappx).^2))

%******************************************************************************************
%-------------- Step 9: Construct the interpolation matrix ---------------------------------
%--------------from Xvld to Xctr ------------------------------------------------------------
%******************************************************************************************
%-------------- 9.1 For MQ + Carlson RBF----------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)= sqrt(r2(i,j)^2 + shp^2);
end
end

%-------------- 9.2 For PS RBF--------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=r2(i,j)^((2*k)-1);
else
Phi2(i,j)=0;
end
end
end
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```

%-------------- 9.3 For TPS RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1 & r2(i,j) ~= 0
Phi2(i,j)=(r2(i,j)^2)*log(r2(i,j));
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.4 For WU1 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^7)*((5*r2(i,j)^6)+(35*r2(i,j)^5)+(101*r2(i,j)^4)+(147*r2(i,j)^3)+(101*r2(i,j)^2)+35*r2(i,j)+5);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.5 For WU2 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^6)*((6*r2(i,j)^5)+(30*r2(i,j)^4)+(72*r2(i,j)^3)+(82*r2(i,j)^2)+36*r2(i,j)+6);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.6 For WU3 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^5)*((5*r2(i,j)^4)+(25*r2(i,j)^3)+(48*r2(i,j)^2)+40*r2(i,j)+8);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.7 For WU4 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^4)*((5*r2(i,j)^3)+(20*r2(i,j)^2)+29*r2(i,j)+16);
else
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
Phi2(i,j)=0;
end
end
end

%---------------- 9.8 For WL1 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j)));
else
Phi2(i,j)=0;
end
end
end

%---------------- 9.9 For WL2 RBF------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^3)*(3*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%---------------- 9.10 For WL3 RBF-----------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^5)*((8*r2(i,j)^2)+5*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%---------------- 9.11 For WL4 RBF-----------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^2);
else
Phi2(i,j)=0;
end
end
end

%---------------- 9.12 For WL5 RBF-----------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2 + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^4)*(4*r2(i,j)+1);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.13 For WL6 RBF----------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^6)*((35*r2(i,j)^2)+18*r2(i,j)+3);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.14 For WL7 RBF----------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=((1-r2(i,j))^8)*((32*r2(i,j)^3)+(25*r2(i,j)^2)+8*r2(i,j)+1);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.15 For BUH1 RBF---------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)=(1/6)-(2*r2(i,j).^2)+(16/3)*r2(i,j).^3 -(7/2)*r2(i,j).^4 +
  2*(r2(i,j).^4).*log(r2(i,j));
if  r2(i,j) == 0
Phi2(i,j)=(1/6)-(2*r2(i,j).^2)+(16/3)*r2(i,j).^3 -(7/2)*r2(i,j).^4;
end
end
end

%-------------- 9.16 For BUH2 RBF---------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
if 0<=r2(i,j) & r2(i,j)<=1
Phi2(i,j)=(112/45)*r2(i,j).^(9/2)+(16/3)*r2(i,j).^(7/2)-7*r2(i,j).^4-(14/15)*r2(i,j).^2+(1/9);
else
Phi2(i,j)=0;
end
end
end

%-------------- 9.17 For BUH3 RBF---------------------------------------------------------------------------
for i=1:N_vld
for j=1:M
r2(i,j)=sqrt((X_vld(1,i)-Xctr(1,j)).^2  + (Y_vld(1,i)-Xctr(2,j)).^2 );
Phi2(i,j)=(1/18)-r2(i,j).^2+(4/9)*r2(i,j).^3+0.5*r2(i,j).^4-(4/3)*r2(i,j).^3.*log(r2(i,j));
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
if  r2(i,j) == 0
Phi2(i,j)=(1/18)-r2(i,j).^2+(4/9)*r2(i,j).^3+0.5*r2(i,j).^4;
end
end
end

%*****************************************************************************************************
%--------------- Step 10:Construct the Zvldapx from Phi2 and weight 'w' --
%*****************************************************************************************************
Zvldapx=Phi2*w;

%*****************************************************************************************************
%--------------- Step 11:Computing the test error with MSE ------------------------------------
%*****************************************************************************************************
Z_vld = Z_vld';
Err_max_vld = max(abs(Z_vld-Zvldapx))
Err_RMSE_vld =sqrt((1/N_vld)*sum((Z_vld-Zvldapx).^2))
toc;
%*****************************************************************************************************
%--------------- Step 12:Find the condition number------------------------------------------------
%*****************************************************************************************************
con_G=cond(G)
con_Phi = cond(Phi)

%*****************************************************************************************************
%--------------- Step 13:Results Plotting ----------------------------------------------------------
%*****************************************************************************************************
for i=1 :N_trn
Solu_tr(i,1)=XYZ_trn(i,1);
Solu_tr(i,2)=XYZ_trn(i,2);
Solu_tr(i,3)=XYZ_trn(i,3);
Solu_tr(i,4)=Zappx(i,1);
End

for i=1 :N_vld
Solu_vld(i,1)=XYZ_vld(i,1);
Solu_vld(i,2)=XYZ_vld(i,2);
Solu_vld(i,3)=XYZ_vld(i,3);
Solu_vld(i,4)=Zvldapx(i,1);
End

figure;
hold on
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,3),'ro','markersize',3,'LineWidth',2)
plot3(Solu_tr(:,1),Solu_tr(:,2),Solu_tr(:,4),'ko','markersize',4,'LineWidth',2)
title('Data set')
xlabel('Input Variable (X)', 'FontSize',14);
ylabel('Input Variable (Y)', 'FontSize',14);
zlabel('Output Variable (Z)', 'FontSize',14);
legend({'Exact','Approximate'});
hold off
grid on

figure;
hold on
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,3),'ro','markersize',3,'LineWidth',2)
plot3(Solu_vld(:,1),Solu_vld(:,2),Solu_vld(:,4),'ko','markersize',4,'LineWidth',2)
title('Data set')
xlabel('Input Variable (X)', 'FontSize',14);
ylabel('Input Variable (Y)', 'FontSize',14);
zlabel('Output Variable (Z)', 'FontSize',14);
legend({'Exact','Approximate'});
hold off
grid on
%----------------------------------------- THIS IS THE END --------------------------------------------------------------memory
```

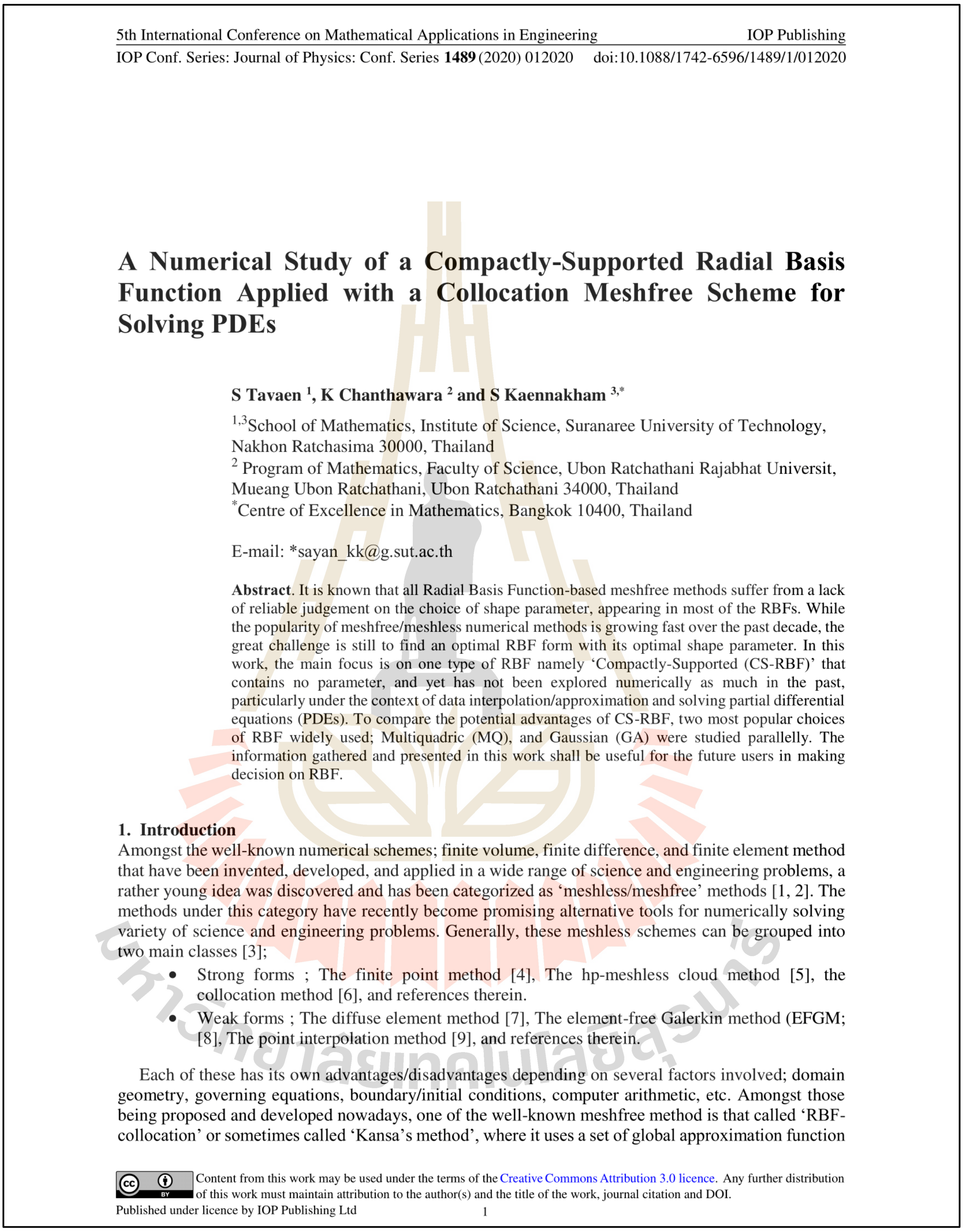
มหาวิทยาลัยเทคโนโลยีสุรนารี

APPENDIX B

PUBLICATIONS (SCOPUS-INDEXED)

มหาวิทยาลัยเทคโนโลยีสุรนารี

B.1 Experiment 1



5th International Conference on Mathematical Applications in Engineering IOP Publishing
IOP Conf. Series: Journal of Physics: Conf. Series **1489** (2020) 012020 doi:10.1088/1742-6596/1489/1/012020

A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs

S Tavaen [1], K Chanthawara [2] and S Kaennakham [3,*]

[1,3]School of Mathematics, Institute of Science, Suranaree University of Technology,
Nakhon Ratchasima 30000, Thailand
[2] Program of Mathematics, Faculty of Science, Ubon Ratchathani Rajabhat Universit,
Mueang Ubon Ratchathani, Ubon Ratchathani 34000, Thailand
[*]Centre of Excellence in Mathematics, Bangkok 10400, Thailand

E-mail: *sayan_kk@g.sut.ac.th

Abstract. It is known that all Radial Basis Function-based meshfree methods suffer from a lack of reliable judgement on the choice of shape parameter, appearing in most of the RBFs. While the popularity of meshfree/meshless numerical methods is growing fast over the past decade, the great challenge is still to find an optimal RBF form with its optimal shape parameter. In this work, the main focus is on one type of RBF namely 'Compactly-Supported (CS-RBF)' that contains no parameter, and yet has not been explored numerically as much in the past, particularly under the context of data interpolation/approximation and solving partial differential equations (PDEs). To compare the potential advantages of CS-RBF, two most popular choices of RBF widely used; Multiquadric (MQ), and Gaussian (GA) were studied parallelly. The information gathered and presented in this work shall be useful for the future users in making decision on RBF.

1. Introduction

Amongst the well-known numerical schemes; finite volume, finite difference, and finite element method that have been invented, developed, and applied in a wide range of science and engineering problems, a rather young idea was discovered and has been categorized as 'meshless/meshfree' methods [1, 2]. The methods under this category have recently become promising alternative tools for numerically solving variety of science and engineering problems. Generally, these meshless schemes can be grouped into two main classes [3];

- Strong forms ; The finite point method [4], The hp-meshless cloud method [5], the collocation method [6], and references therein.
- Weak forms ; The diffuse element method [7], The element-free Galerkin method (EFGM; [8], The point interpolation method [9], and references therein.

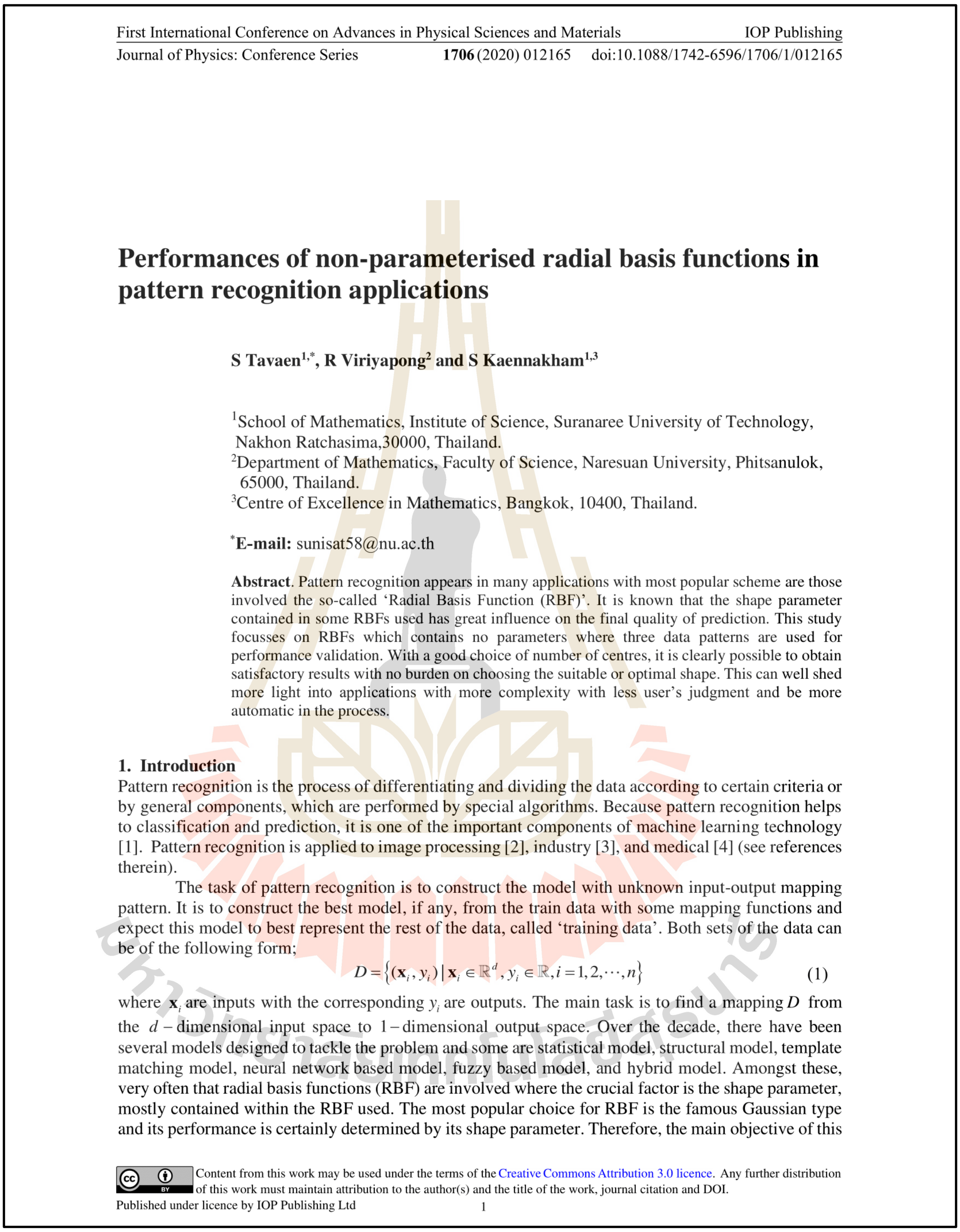
Each of these has its own advantages/disadvantages depending on several factors involved; domain geometry, governing equations, boundary/initial conditions, computer arithmetic, etc. Amongst those being proposed and developed nowadays, one of the well-known meshfree method is that called 'RBF-collocation' or sometimes called 'Kansa's method', where it uses a set of global approximation function

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

มหาวิทยาลัยเทคโนโลยีสุรนารี

Figure B.1 A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs.

B.2 Experiment 2



First International Conference on Advances in Physical Sciences and Materials IOP Publishing
Journal of Physics: Conference Series **1706** (2020) 012165 doi:10.1088/1742-6596/1706/1/012165

Performances of non-parameterised radial basis functions in pattern recognition applications

S Tavaen[1,*]**, R Viriyapong**[2] **and S Kaennakham**[1,3]

[1]School of Mathematics, Institute of Science, Suranaree University of Technology, Nakhon Ratchasima,30000, Thailand.
[2]Department of Mathematics, Faculty of Science, Naresuan University, Phitsanulok, 65000, Thailand.
[3]Centre of Excellence in Mathematics, Bangkok, 10400, Thailand.

[*]**E-mail:** sunisat58@nu.ac.th

Abstract. Pattern recognition appears in many applications with most popular scheme are those involved the so-called 'Radial Basis Function (RBF)'. It is known that the shape parameter contained in some RBFs used has great influence on the final quality of prediction. This study focusses on RBFs which contains no parameters where three data patterns are used for performance validation. With a good choice of number of centres, it is clearly possible to obtain satisfactory results with no burden on choosing the suitable or optimal shape. This can well shed more light into applications with more complexity with less user's judgment and be more automatic in the process.

1. Introduction

Pattern recognition is the process of differentiating and dividing the data according to certain criteria or by general components, which are performed by special algorithms. Because pattern recognition helps to classification and prediction, it is one of the important components of machine learning technology [1]. Pattern recognition is applied to image processing [2], industry [3], and medical [4] (see references therein).

The task of pattern recognition is to construct the model with unknown input-output mapping pattern. It is to construct the best model, if any, from the train data with some mapping functions and expect this model to best represent the rest of the data, called 'training data'. Both sets of the data can be of the following form;

$$D=\left\{(\mathbf{x}_i, y_i)\,|\,\mathbf{x}_i\in\mathbb{R}^d, y_i\in\mathbb{R}, i=1,2,\cdots,n\right\} \qquad (1)$$

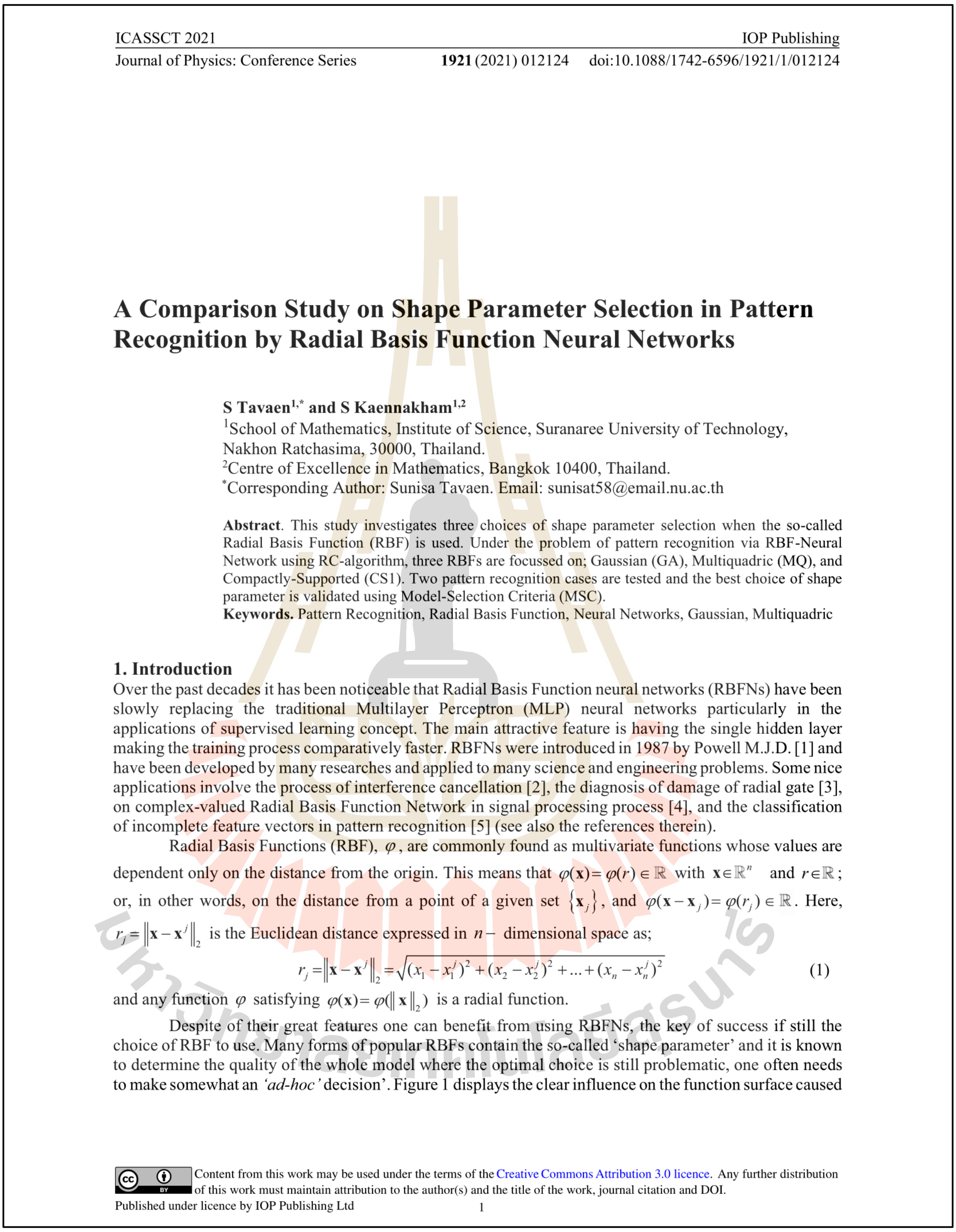
where $\mathbf{x}_i$ are inputs with the corresponding y_i are outputs. The main task is to find a mapping D from the $d-$dimensional input space to $1-$dimensional output space. Over the decade, there have been several models designed to tackle the problem and some are statistical model, structural model, template matching model, neural network based model, fuzzy based model, and hybrid model. Amongst these, very often that radial basis functions (RBF) are involved where the crucial factor is the shape parameter, mostly contained within the RBF used. The most popular choice for RBF is the famous Gaussian type and its performance is certainly determined by its shape parameter. Therefore, the main objective of this

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

มหาวิทยาลัยเทคโนโลยีสุรนารี

Figure B.2 Performances of Non-Parameterised Radial Basis Functions in Pattern Recognition Applications.

B.3 Experiment 3



ICASSCT 2021 IOP Publishing
Journal of Physics: Conference Series **1921** (2021) 012124 doi:10.1088/1742-6596/1921/1/012124

A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks

S Tavaen[1,*] **and S Kaennakham**[1,2]
[1]School of Mathematics, Institute of Science, Suranaree University of Technology, Nakhon Ratchasima, 30000, Thailand.
[2]Centre of Excellence in Mathematics, Bangkok 10400, Thailand.
[*]Corresponding Author: Sunisa Tavaen. Email: sunisat58@email.nu.ac.th

Abstract. This study investigates three choices of shape parameter selection when the so-called Radial Basis Function (RBF) is used. Under the problem of pattern recognition via RBF-Neural Network using RC-algorithm, three RBFs are focussed on; Gaussian (GA), Multiquadric (MQ), and Compactly-Supported (CS1). Two pattern recognition cases are tested and the best choice of shape parameter is validated using Model-Selection Criteria (MSC).

Keywords. Pattern Recognition, Radial Basis Function, Neural Networks, Gaussian, Multiquadric

1. Introduction

Over the past decades it has been noticeable that Radial Basis Function neural networks (RBFNs) have been slowly replacing the traditional Multilayer Perceptron (MLP) neural networks particularly in the applications of supervised learning concept. The main attractive feature is having the single hidden layer making the training process comparatively faster. RBFNs were introduced in 1987 by Powell M.J.D. [1] and have been developed by many researches and applied to many science and engineering problems. Some nice applications involve the process of interference cancellation [2], the diagnosis of damage of radial gate [3], on complex-valued Radial Basis Function Network in signal processing process [4], and the classification of incomplete feature vectors in pattern recognition [5] (see also the references therein).

Radial Basis Functions (RBF), φ, are commonly found as multivariate functions whose values are dependent only on the distance from the origin. This means that $\varphi(\mathbf{x})=\varphi(r)\in\mathbb{R}$ with $\mathbf{x}\in\mathbb{R}^n$ and $r\in\mathbb{R}$; or, in other words, on the distance from a point of a given set $\{\mathbf{x}_j\}$, and $\varphi(\mathbf{x}-\mathbf{x}_j)=\varphi(r_j)\in\mathbb{R}$. Here, $r_j=\left\|\mathbf{x}-\mathbf{x}^j\right\|_2$ is the Euclidean distance expressed in $n-$ dimensional space as;

$$r_j=\left\|\mathbf{x}-\mathbf{x}^j\right\|_2=\sqrt{(x_1-x_1^j)^2+(x_2-x_2^j)^2+\ldots+(x_n-x_n^j)^2} \qquad (1)$$

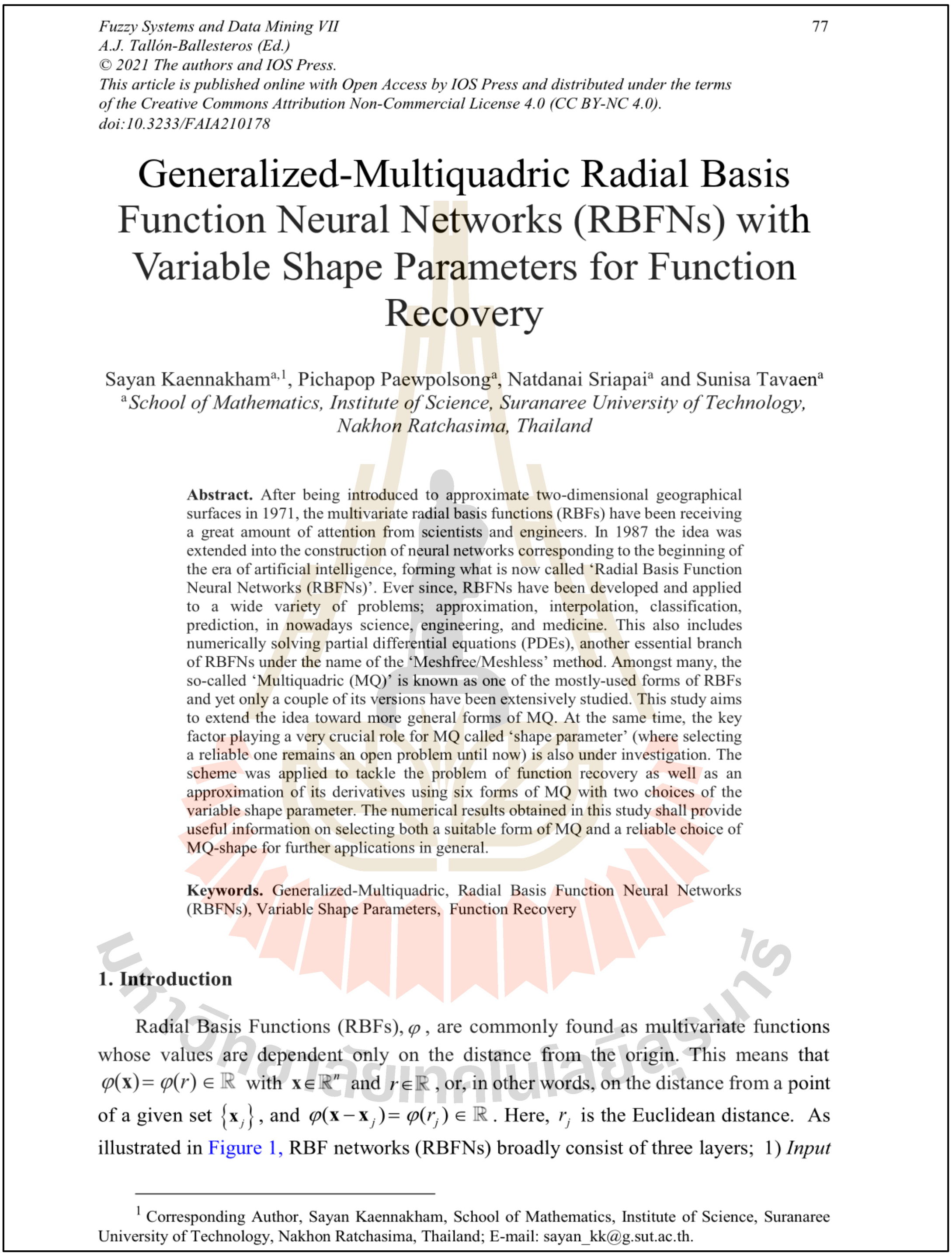
and any function φ satisfying $\varphi(\mathbf{x})=\varphi(\|\mathbf{x}\|_2)$ is a radial function.

Despite of their great features one can benefit from using RBFNs, the key of success if still the choice of RBF to use. Many forms of popular RBFs contain the so-called 'shape parameter' and it is known to determine the quality of the whole model where the optimal choice is still problematic, one often needs to make somewhat an *'ad-hoc'* decision'. Figure 1 displays the clear influence on the function surface caused

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

Figure B.3 A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks.

B.4 Experiment 4



Fuzzy Systems and Data Mining VII
A.J. Tallón-Ballesteros (Ed.)

© 2021 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA210178

Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery

Sayan Kaennakham[a,1], Pichapop Paewpolsong[a], Natdanai Sriapai[a] and Sunisa Tavaen[a]
[a] *School of Mathematics, Institute of Science, Suranaree University of Technology, Nakhon Ratchasima, Thailand*

Abstract. After being introduced to approximate two-dimensional geographical surfaces in 1971, the multivariate radial basis functions (RBFs) have been receiving a great amount of attention from scientists and engineers. In 1987 the idea was extended into the construction of neural networks corresponding to the beginning of the era of artificial intelligence, forming what is now called 'Radial Basis Function Neural Networks (RBFNs)'. Ever since, RBFNs have been developed and applied to a wide variety of problems; approximation, interpolation, classification, prediction, in nowadays science, engineering, and medicine. This also includes numerically solving partial differential equations (PDEs), another essential branch of RBFNs under the name of the 'Meshfree/Meshless' method. Amongst many, the so-called 'Multiquadric (MQ)' is known as one of the mostly-used forms of RBFs and yet only a couple of its versions have been extensively studied. This study aims to extend the idea toward more general forms of MQ. At the same time, the key factor playing a very crucial role for MQ called 'shape parameter' (where selecting a reliable one remains an open problem until now) is also under investigation. The scheme was applied to tackle the problem of function recovery as well as an approximation of its derivatives using six forms of MQ with two choices of the variable shape parameter. The numerical results obtained in this study shall provide useful information on selecting both a suitable form of MQ and a reliable choice of MQ-shape for further applications in general.

Keywords. Generalized-Multiquadric, Radial Basis Function Neural Networks (RBFNs), Variable Shape Parameters, Function Recovery

มหาวิทยาลัยเทคโนโลยีสุรนารี

1. Introduction

Radial Basis Functions (RBFs), φ, are commonly found as multivariate functions whose values are dependent only on the distance from the origin. This means that $\varphi(\mathbf{x}) = \varphi(r) \in \mathbb{R}$ with $\mathbf{x} \in \mathbb{R}^n$ and $r \in \mathbb{R}$, or, in other words, on the distance from a point of a given set $\{\mathbf{x}_j\}$, and $\varphi(\mathbf{x} - \mathbf{x}_j) = \varphi(r_j) \in \mathbb{R}$. Here, r_j is the Euclidean distance. As illustrated in Figure 1, RBF networks (RBFNs) broadly consist of three layers; 1) *Input*

---

[1] Corresponding Author, Sayan Kaennakham, School of Mathematics, Institute of Science, Suranaree University of Technology, Nakhon Ratchasima, Thailand; E-mail: sayan_kk@g.sut.ac.th.

Figure B.4 Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery.

CURRICULUM VITAE

NAME : Sunisa Tavaen **GENDER :** Female

EDUCATION BACKGROUND:

- Bachelor of Science Program in Mathematics, Naresuan University, Thailand, 2019

SCHOLARSHIP:

- Development and Promotion of Science and Technology Talents Project, Thai government scholarship

SELECTED PUBLICATIONS:

- **Sunisa Tavaen**, Krittidej Chanthawara and Sayan Kaennakham (2020) A Numerical Study of a Compactly-Supported Radial Basis Function Applied with a Collocation Meshfree Scheme for Solving PDEs, "Journal of Physics Conference Series", vol. 1489, no.1. doi: 10.1088/1742-6596/1489/1/012020.
- **Sunisa Tavaen**, Ratchada Viriyapong and Sayan Kaennakham (2020) Performances of non-parameterised radial basis functions in pattern recognition applications, "Journal of Physics Conference Series", vol. 1706, no.1. doi: 10.1088/1742-6596/1706/1/012165.
- **Sunisa Tavaen** and Sayan Kaennakham (2021) A Comparison Study on Shape Parameter Selection in Pattern Recognition by Radial Basis Function Neural Networks, "Journal of Physics: Conference Series: in First International Conference on Advances in Smart Sensor, Signal Processing and Communication Technology (ICASSCT 2021), 19-20, March 2021, Goa, India", vol. 1921.
- Sayan Kaennakham, Pichapop Paewpolsong, Natdanai Sriapai and **Sunisa Tavaen** (2021) Generalized-Multiquadric Radial Basis Function Neural Networks (RBFNs) with Variable Shape Parameters for Function Recovery. "Fuzzy Systems and Data Mining VII", vol. 340. pp. 77 - 85. doi: 10.3233/FAIA210178.
- Dusita Ritthison, **Sunisa Tavaen** and Sayan Kaennakham (2022) A Modified Local Distance-weighted (MLD) Method of Interpolation and Its Numerical Performances for Large Scattered Datasets. "Current Applied Science and Technology", vol. 22, no. 6.
- **Sunisa Tavaen** and Sayan Kaennakham (202x) Numerical Comparison of Shapeless Radial Basis Function Networks in Pattern Recognition. "Computers, Materials & Continua", vol. x. doi:10.32604/cmc.202x.xxxxxx. (Accepted)

มหาวิทยาลัยเทคโนโลยีสุรนารี