COMPARISON ANALYSIS BETWEEN MESHLESS RADIAL BASIS FUNCTION COLLOCATION METHOD AND FINITE ELEMENT METHOD

by

İsmet Karakan

B.Sc., Civil Engineering, Boğaziçi University, 2017

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

Graduate Program in Civil Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere appreciation to my thesis supervisor Prof. Cem Avcı for his patience and guidance. His supportive behaviour made the hardest part, which was the selection of the thesis subject, easier.

Besides my thesis supervisor, I would like to thank Assoc. Prof. Osman Börekçi for lecturing the marvelous meshless methods course. His courses with his wonderful insights widen my comprehension greatly.

Further, many thanks to my family members: Şevval, Naile, Zorer, Sevinç, and Harun for their unending support and encouragement. They were the ones who felt the lack of me in their life most throughout my years of study. Also, I want to thank my other family members and my friends for their companionship during the writing process of this thesis.

Last but not the least, I want to express my deepest gratitude to Assist. Prof. Ceren Gürkan, with whose continuous contributions this work comes into being. From the smallest obstacle to bigger troubles, she was there to help and explain patiently.

ABSTRACT

COMPARISON ANALYSIS BETWEEN MESHLESS RADIAL BASIS FUNCTION COLLOCATION METHOD AND FINITE ELEMENT METHOD

Comparison analysis between meshless Radial Basis Function Collocation Method and Finite Element Method is conducted in this work. Models for both steady and unsteady versions of Poisson and Stokes equation with various types of boundary conditions are built. The domains studied are square and L-shaped. The mesh, or the number of nodes inside these domains is gradually increased to observe the convergence properties of the methods. For the meshless method, a novel least square error calculation technique is presented to make the error comparison against Finite Element Method fair. Additionally, the shape factor optimization algorithm which minimizes the root mean square error is implemented for the meshless model to yield the most accurate results available. Then, the performances of both methods are compared considering several parameters. These parameters are chosen to be: accuracy by comparing least square error, root mean square error and maximum relative error; stability by comparing condition numbers; robustness by comparing convergence rates; computational cost by comparing runtimes; and ease of implementation.

ÖZET

AĞSIZ RADYAL BAZLI FONKSİYON EŞYERLEŞİM YÖNTEMİ VE SONLU ELEMAN YÖNTEMİNİN KARŞILAŞTIRMA ANALİZİ

Bu çalışmada ağsız Radyal Bazlı Foksiyon Eşyerleşim Yöntemi ile Sonlu Elemanlar Yöntemi arasında karşılaştırma analizi yapıldı. Poisson ve Stokes parçalı diferansiyel denklemlerinin durağan ve zamana bağlı örnekleri, değişik tipteki sınır şartları göz önüne alınarak modellendi. Denklemlerin sayısal olarak çözdürüldüğü bölgeler kare ve L-şekilli olacak şekilde seçildi. Bu iki yöntemin doğru sonuca yakınsama özelliklerini gözlemlemek için çözüm yapılan bölge içindeki çözüm noktaları sayısı gitgide artırıldı. Ağsız yöntemin sonuçlarının en küçük kareler hatası hesabı için yeni bir yöntem geliştirildi. Ayrıca, ağsız modelin en kesin sonucu vermesi için minimum ortalama karekök hatasını dikkate alarak en uygun şekil etkenini bulan bir algoritma da modellerde kullanıldı. Daha sonra, bu iki yöntemin performansları, çeşitli parametreler özelinde karşılaştırıldı. Bu parametreler, kesinlik için en küçük kareler hatası, ortalama karekök hatası ve en büyük bağıl hata; stabilite için matrislerin koşul sayısı; dayanıklılık için yakınsama oranı; sayısal maliyet için modellerin çözüm yapma süreleri; ve modellerin kolay uygulanabilirliği olmak üzere seçildi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	System matrix
b	Load vector
c	Shape factor
f	Source term
f_j	Source term values at jth node
K	Stifness matrix
M	Mass matrix
N	Number of nodes
p	Pressure
T_f	Final time step
u_{Aj}	Approximated value of the unknown u at jth node (RBFCM)
u_j	Approximated value of the unknown u at jth node (FEM)
u_x	Velocity in x-direction
u_y	Velocity in y-direction
z	Load vector
α	Coefficient vector of RBFCM
δh	Element size for FEM, or the distance between collocation nodes for RBFCM
δt	Time step size
ϕ_j	Global Shape function
φ_{ij}	Radial basis function at ith RBF center and jth collocation point
Φ	Matrix of radial basis functions
ψ_j	Local Shape function

LIST OF ACRONYMS/ABBREVIATIONS

CFES	Commercial Finite Element Software
CFES-P1	Commercial Finite Element Software with linear elements
CFES-P2	Commercial Finite Element Software with second order elements
FEM	Finite Element Method
FEM-P1	Finite Element Method with linear elements
FEM-P2	Finite Element Method with second order elements
LSE	Least Square Error
MQ	Multiquadrics
MQRBF	Multiquadric Radial Basis Function
MRE	Maximum Relative Error
RBF	Radial Basis Functions
RBFCM	Radial Basis Function Collocation Method
RBFCM-MQ	Radial Basis Function Collocation Method with Multiquadrics
RBFCM-MQ*	Radial Basis Function Collocation Method with Multiquadrics used in a domain where the collocation nodes are randomly distributed
RBFCM-TPS	Radial Basis Function Collocation Method with Thin Plate Splines
RBFCM-TPS*	Radial Basis Function Collocation Method with Thin Plate Splines used in a domain where the collocation nodes are randomly distributed
RMSE	Root Mean Square Error
SC	Symmetrical Collocation
TPS	Thin Plate Splines

1. INTRODUCTION

1.1. Background

Solving complex problems encountered in the field of science or engineering experimentally or analytically is getting less practical and more costly with the mind blowing advances in computer technology. Today, numerical modeling is among the hottest research topics and hence the theme of this thesis work.

Computers are very fast compared to humans at executing repetitive patterns. This is why at numerical modeling of complex solutions are sought by "discretization". These discretization processes, in fact, diverge the numerical solution away from the real solution, which, at the end, makes the necessity of the "approximation" apparent. Thus, all of these processes involve approximations which, if set up correctly, enforce the numerical solutions to get close enough to the real solutions.

1.2. Earlier Work

1.2.1. Finite Element Method

The early foundations of the Finite Element Method started at 1940's. McHenry [1] described a method which may be fit into almost all types of two-dimensional stress problems involving elastic materials. The method is proposed [2] by creating an analogy between real discrete elements and finite portions of the domain, and it is claimed that the method is adequate to yield numerical solutions to many problems which do not have simple solutions. Later, Turner *et al.* [3] enhanced the method further by calculating the stiffness influence coefficients of complex shell-type structures. They obtained the stiffness of the structure by summing the stiffnesses of individual units. However, the term "finite element" is first used by Clough [4] in 1960, and with the enormous growth of the computer power, the method thrived on solving various differ-

ential equations.

The first implementations of the finite element method to model fluid flows were around 1970's. Temam, in 1977, dealt with some of the theory and numerical analysis of the Navier-Stokes equations for viscous incompressible fluids. He utilized finite element method -as well as finite difference method- for linearized stationary case, non-linear stationary case, and fully non-linear time-dependent case, respectively [5]. More general derivations and discretizations, with an emphasis on finite element method rather than mathematics of Navier-Stokes equations, was provided by Thomasset in [6]. At those early times, the finite element method was well-established for structural analysis, however, the use of the method to model fluids was newly emerging. Later in [7–9], transonic flows with shocks, non-Newtonian fluids and flow in porous media are studied respectively.

Today FEM is very well studied for fluid modeling as well. Turbulent flow models [10, 11], complex fluid-structure interaction models [12] or even blood flow simulations [13] are among the very recent and exciting applications of FEM. Moreover, other than its common use in structural mechanics and fluid dynamics, the application of the finite element method ranges across different fields such as electromagnetics [14], noise propogation [15] and even biological systems [16, 17].

1.2.2. Meshless Radial Basis Functions Collocation Method

Unlike the finite element method, meshless methods are relatively new in the field of computational methods. There are bunch of numerical techniques which are named as "Meshless Methods". In meshless approach, the domain is decomposed into nodes rather than finite elements or volumes. These nodes do not need to be placed regularly, or uniformly since they are represented by radial basis functions.

In 1971, Hardy presented a new approach to the development of equations of topography and other irregular surfaces. Hardy's "Multiquadric analysis" finds the

equation of the surface which fits all the significant points exactly, in a simple manner [19]. Later, Franke evaluated some of the scattered data interpolation methods. Among those methods, Franke stated that the most successful one was Hardy's multiquadrics in terms of fitting ability and visual smoothness [21]. An alternative to MQ, the thin plate splines was firstly provided by Harder and Desmarais [22] and theoretical basis for the thin plate splines was studied by Meinguet [23].

Kansa, in 1990, modified multiquadrics scheme to apply it to partial derivative estimates along with surface approximations [24]. In the second paper of the series [25], Kansa applied the same scheme for parabolic, hyperbolic and elliptic partial differential equations. He presented solutions for the linear advection-diffusion problem using an implicit time-marching scheme on the Eulerian frame and compared it with finite difference scheme. Conclusion was that the MQ scheme was superior to FD scheme. Also in [25], he provided a solution of a two dimensional elliptic Poisson's equation subjected to dirichlet and Neumann boundary conditions and the solution was in a good agreement with the exact solution. Later, Kansa with Carlson [26] improved the accuracy of the scheme using variable shape parameters and with Hon [27] studied the ill-conditioning problem with MQRBF.

Since RBFCM is relatively new, there are bunch of studies going on to improve the use of it. In 2002, Sarler provided numerical implementation of global RBF for the non-linear Poisson's equation together with implicit time-discretization [28]. Later in 2004, Sarler *et al.* used Kansa's RBFCM to solve the steady-state natural convection in porous media [29]. In 2003, Lee *et al.* introduced *local* collocation, alternative to global collocation, mainly to overcome non-sparse matrices resulting from global collocation. The matrices generated using local collocation method are sparse because of the fact that the approximated value at a node is interpolated only using the nodes that fall into the domain of influence of that node rather than all the nodes in the domain, hence the approach was named as "local". Moreover, they stated that the local collocation technique is stable and it is able to yield accurate solutions regardless of the shape parameters used [30]. Following that, Sarler *et al.* used localized RBFCM to solve the

diffusion equation [31] and Darcy flow [32]. In addition, Siraj-ul-Islam *et al.* analysed one-dimensional advection, inviscid Burgers' equation and dimensionless form of two-dimensional Burgers' equation using localized RBFCM with explicit time stepping [33]. To have a general overview of various meshless methods and their applications, reader is referred to [34].

1.3. Novel Contributions and Outline of the Thesis

Other than a loose theoretical base provided by Hardy [35], a rigorous theoretical base for MQ is not provided yet because of the fact that its mathematical analysis is very difficult [36]. Therefore, it is not clear why MQ renders so well. Also, because of the same fact, the convergence property of MQ is not well known and thus, it is not possible to make an error analysis before conducting numerical experiments. Hence, comparisons between FEM and RBFCM is inevitable to have an idea about the accuracy properties of RBFCM. However, although many studies are carried on for both FEM and RBFCM, the direct comparison between these two methods have been done rather few.

Li *et al.* compared these two methods considering 2D Poisson's equation with over different domains. They used L-shaped and square shaped domains. Noting that the test function that they used was first order polynomial for the finite element analysis, they concluded that the RBFCM performed better even though the shape parameter, c, is arbitrarily selected [37]. Nonetheless, the model that they built was not using a time-marching scheme, thus yielding only a steady state solution. Another comparison was done by Gu and Liu, working over 1D and 2D convection-diffusion problems with several Peclet numbers. The comparison, however, was not done in regard of accuracy. Rather, the conclusion was that the instability problems arising from high Peclet numbers can be easily overcome by RBFCM with the techniques proposed in the paper, thus making RBFCM advantageous against FEM [38]. Later in 2012, Golbabai and Raibei studied Stokes eigenproblem in primitive variables of pressure and velocity. They provided an RBF formulation for the problem and compared the

results with FEM variations [39]. As well as other papers, the model built in the paper was time-independent. Likewise, Ozgener and Tanbay, made a comparison studying neutron diffusion equations. They also investigated the effect of shape parameter on accuracy, convergence rate and stability of RBFCM [40].

In this thesis work, an extensive comparison analysis between FEM and RBFCM is provided considering both two dimensional Poisson's equation and Stokes equation with Dirichlet and Neumann boundary conditions. For accuracy, relative error analysis and root mean square error analysis is supplied, keeping the costs proximate to each other. In the matter of stability, condition numbers of matrices for both cases are provided quantitatively. For speed, code run times are compared. And for ease of implementation, a qualitative conclusion is drawn. In the finite element analysis, the most common method of calculating global stiffness matrix, the Galerkin method of weighted residuals, is used. Linear elements along with higher order elements are utilized. In RBFCM part, multiquadric radial basis function is applied in general. And the radial basis functions are not localized, the global discretizations are used.

To sum up, the above-mentioned FEM vs RBFCM comparison papers lack some further analysis to get a better understanding of the relative performances hence this work focus on closing this gap. In this study, rather than working over time-independent models, time marching schemes are implemented. This work considers test functions of higher orders, not only linear ones for the finite element analysis. Moreover, the RBFCM model presented here optimizes the shape parameter for the minimum root mean square error, leading to a fair comparison.

The outline of this work is as follows: In Chapter 2, the very basics, numerical formulations, error calculations for Finite Element method is presented. In Chapter 3, the same analysis is done for Radial Basis Function Collocation Method. This chapter also includes the the technique used to optimize shape factor. The numerical examples for both Poisson and Stokes Equations are presented in Chapter 4. Specific results for each example can also be found in this chapter. The final words and comments on the

performances of both numerical methods are given in Chapter 5.

2. FINITE ELEMENT METHOD

Finite element method is one of the most general and widely used numerical methods which divides the domain of interest into sub-domains called *elements*. The calculations carried out in these finite elements then collected to generate an approximation of the solution over the whole domain. Countless studies including its numerical theory, error analysis, potential improvements, etc. are done in the literature[2,51, and *the references therein*]. In addition to the solid ideas behind the method, these studies surely made it very robust and well understood.

In order for FEM to approximate the solution, weak form of the problem must be defined beforehand[1] . The role of the weak form is to make continuous problem mathematically suitable to be treated in discrete form by introducing a so called test function (or approximation function), v, to the system. The solution is approximated using these test functions defined on each element (and hence on overall domain) and a numerical solution is obtained. The test functions are often chosen in polynomial form, so they are easily differentiable. They can be chosen first order, second order or higher orders depending on the goals of the study. Since the purpose of this work is to compare FEM and RBFCM, we used only up to second order polynomial test functions. Even though using second order polynomial test functions increase the degree of freedom, or the number unknown nodes, by a fine amount, meaning that there are more nodes to solve in the system matrix, we chose to let the reader to decide or evaluate the success of second order FEM against RBFCM.

To construct the approximation functions in 2-dimensions, we start with the triangular elements and a linear (first order) polynomial function of the form:

$$v_h = a_1 + a_2 x + a_3 y \tag{2.1}$$

[1] All the weak formulations for the differential equations used in this work are done in Appendix A.

The equation above defines a planar surface at each corner of a triangle, see Figure 2.1. When we solve the linear system of equations constructed by writing Equation 2.1 at three corners of the triangular element, and eliminating a_1, a_2 and a_3 from the system, we obtain the Equation 2.2, which defines the shape function over an element:

$$v_h^e(x,y) = v_1\psi_1^e(x,y) + v_2\psi_2^e(x,y) + v_3\psi_3^e(x,y) \tag{2.2}$$

where $\psi_i^e(x,y)$ are the first order local shape functions for the element and they are written as the following:

$$\psi_1^e(x,y) = \frac{1}{2A_e}[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y] \tag{2.3}$$

$$\psi_2^e(x,y) = \frac{1}{2A_e}[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y] \tag{2.4}$$

$$\psi_3^e(x,y) = \frac{1}{2A_e}[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y] \tag{2.5}$$

where A_e is the area of the triangular element.

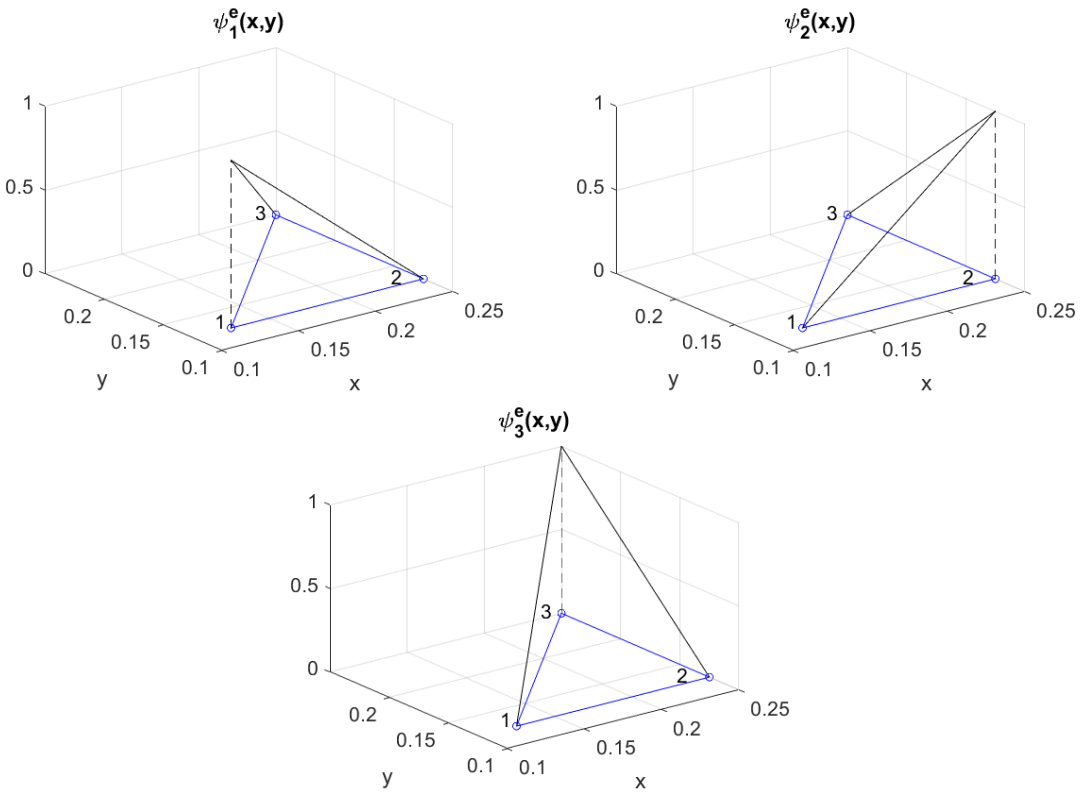

Figure 2.1. The first order shape functions, ψ_i^e, on an arbitrary element.

Second order shape functions ψ_2^e and ψ_4^e, on the other hand, are presented in Figure 2.2 only for visualization. The mathematical formulation of the second order shape functions is not included here since it is derived very similar to that of first order and it can be found in almost every textbook about FEM.

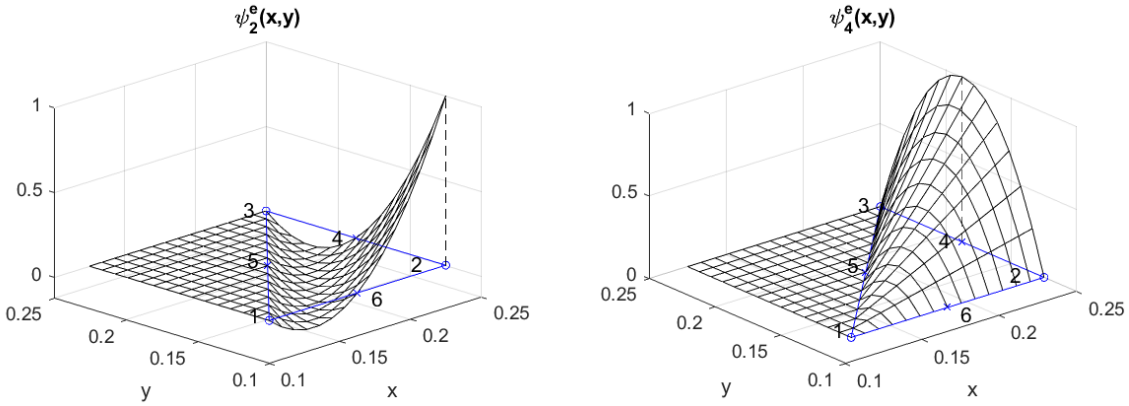

Figure 2.2. The second order shape functions ψ_2^e and ψ_4^e on an arbitrary element.

It is important to see the following property of the shape functions, which simply means that the shape functions satisfy the property of unity at every node in the triangle, they are equal to 1 at the node they are defined, and 0 at all other nodes.

$$\psi_i^e(x_j, y_j) = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

These local shape functions are then combined together to construct the "global" test function $\phi_j(x, y)$. This process is done by patching the local shape functions of adjacent elements. For a 1-dimensional element, this process is depicted in Figure 2.3. The local shape functions are shown on the top and the global test function defined at node 3 is shown in the bottom.

When we put the global test functions written at all the nodes together, we can write it as the multiplication of some coefficient and the test function defined at that node (see Equation 2.7). Similarly, we can approximate the unknown of our problem, say $u_h(x, y)$, by writing it as the sum of the multiplication of "coefficients" u_j and the

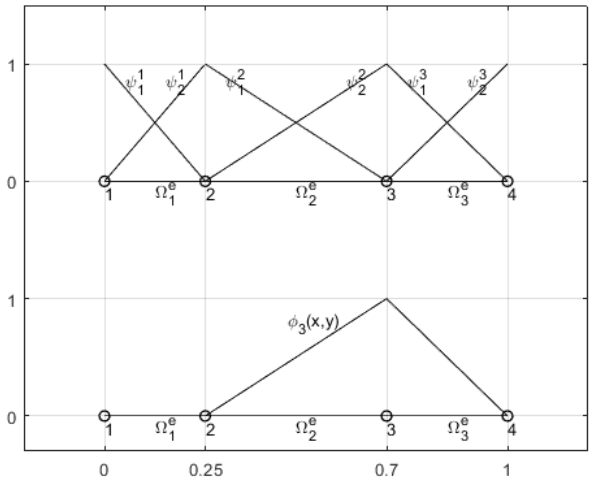

Figure 2.3. *Local* shape functions and a *global* test function in the domain.

global test functions ϕ_j defined at every node:

$$u_h(x,y) = \sum_{j=1}^{N} u_j \phi_j(x,y) \tag{2.6}$$

$$v_h(x,y) = \sum_{j=1}^{N} v_j \phi_j(x,y) \tag{2.7}$$

Following the weak forms defined in Appendix A, together with the discretized definition of unknown $u_h(x,y)$ defined in Equation 2.6, any linear differential equation can be written in the form as such:

$$\sum_{j=1}^{N} K_{ij} u_j = f_j \tag{2.8}$$

where, K_{ij} is the stiffness matrix and f_j is the load vector[2] .

As mentioned before, FEM starts with defining the weak form of the problem. These weak forms include integrals to be calculated. Therefore,to calculate the integrals numerically, we need numerical integration techniques. In this work, we used Gaussian

[2]This terminology is often used in the area of structural analysis. FEM is initially used in this area.

quadrature technique to deal with these integrals. For P1 (linear) elements, 12 points for the quadrature was enough to calculate the integral and for P2 (second order) elements, 25 points are used. The weights and the locations of these quadrature points are presented in Appendix B. A reference element is often used to calculate numerical integrals over. Since the domain is composed of many elements of different orientation, integration calculations are first done over a reference element and then transformed to a real element. The depiction of the reference element that we did our element transformations from is in the Figure 2.4.

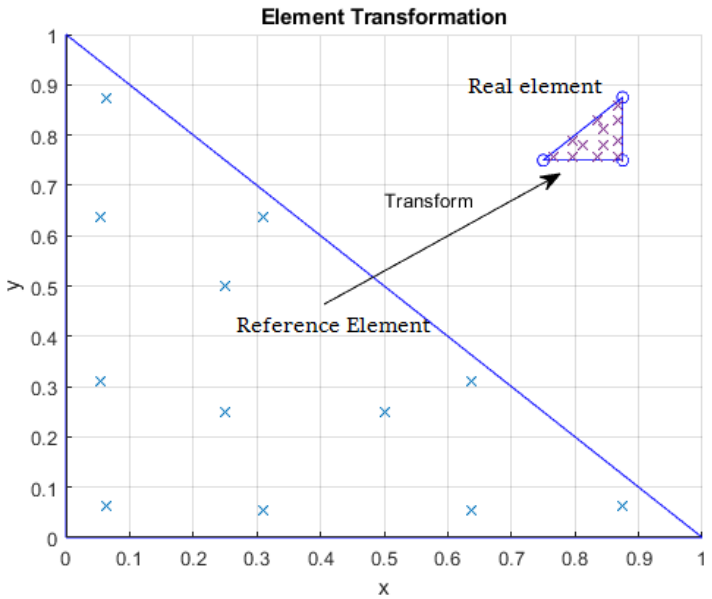

Figure 2.4. An example of element transformation.

2.1. Numerical Formulation

2.1.1. Steady Poisson Equation

Considering the strong form of steady Poisson equation[3] ,

$$\nabla \cdot (-k \nabla u) = f \tag{2.9}$$

[3]The physical meaning and the use of Poisson equation is discussed in Section 4.1.

This equation can be reduced to a weak form by multiplying it by a shape function v and taking the integral of both sides together with applying integration by parts rule[4] :

$$\int_{\Omega} k\nabla u \cdot \nabla v \; d\Omega = \int_{\Omega} fv \; d\Omega + \int_{\partial\Omega} gv \; dS \tag{2.10}$$

Equation 2.10 can be written for a 2-dimensional element as:

$$\int_{\Omega^e} \left[k \left(\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y} \right) \right] \; d\Omega = \int_{\Omega^e} f v_h \; d\Omega + \int_{\partial\Omega^e} g v_h \; dS \tag{2.11}$$

Replacing Equations 2.6 and 2.7 in 2.11, K_{ij} for just one element can be written as:

$$K_{ij}^e = \int_{\Omega^e} \left[k \left(\frac{\partial \psi_i}{\partial x}\frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial \psi_j}{\partial y} \right) \right] \; d\Omega \tag{2.12}$$

and also, the load vector f_j :

$$f_j^e = \int_{\Omega^e} f\psi_j \; d\Omega + \int_{\partial\Omega^e} g\psi_j \; dS \tag{2.13}$$

At this point, elemental forms of stifness matrices and the load vectors are assembled to create the global stifness matrix and the global load vector. The assembly process is presented in Figure 2.1.1. Each square block depicted in the figure is a local stiffness matrix of corresponding element.

After assembling the global stiffness matrix, K_{ij}, and the global load vector, f_j, of the system using the elemental ones, K_{ij}^e and f_j^e, K_{ij} and f_j are edited to apply the Dirichlet boundary condition. Since the values at the Dirichlet boundaries are known, the ith node corresponding to Dirichlet boundary is removed from both K_{ij} and f_j to reduce the global system size we will be solving. However, to remove the global load vector, the vector found by multiplication of each row of $K_{ndbc,dbc}$ and d_{dbc} is subtracted from f_{ndbc}, where subscript *ndbc* stands for *not Dirichlet boundary* and *dbc* stands for

[4]Detailed formulations of weak forms are given in Appendix A.

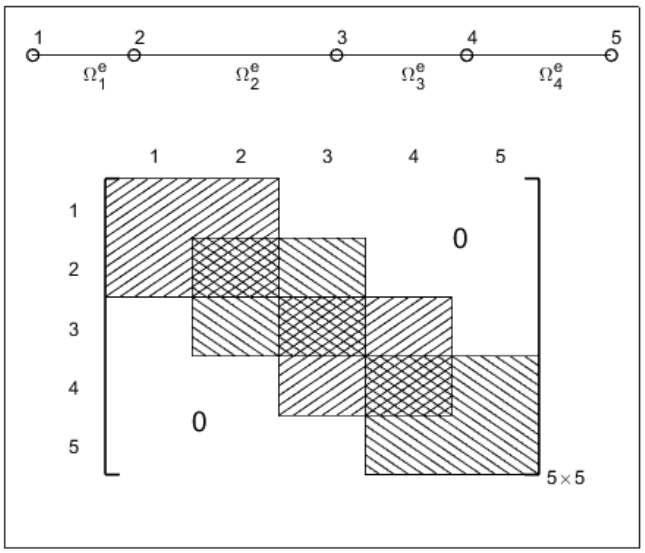

Figure 2.5. Assembly of local siffness matrices to global stifness matrix.

Dirichlet boundary. Finally, using Equation 2.8, the unknown at the non-Dirichlet nodes is found by:

$$u_{ndbc} = K^{-1}_{ndbc,ndbc} f_{ndbc} \tag{2.14}$$

The overall solution algorithm for steady Poisson equation is presented below:

1) Set input variables such as geometry properties, source term and boundary conditions.
2) Generate connectivity matrix.
3) Using Gaussian quadrature, create K^e_{ij} and f^e_j for each element.
4) Assemble global system matrix, K_{ij}, and load vector, f_j and reduce the known Dirichlet nodes.
5) Invert system matrix and solve for u_{ndbc} by using Equation 2.14.

2.1.2. Unsteady Poisson Equation

Considering the strong form of unsteady Poisson equation,

$$\frac{\partial u}{\partial t} + \nabla \cdot (-k \nabla u) = f \tag{2.15}$$

can be reduced to a weak form by following the same procedure in the previous section. Therefore, the weak form of unsteady Poisson equation can be defined as:

$$\int_T \int_\Omega \frac{\partial u_h}{\partial t} v_h \; d\Omega dt + \int_T \int_\Omega k \nabla u \cdot \nabla v \; d\Omega dt = \int_T \int_\Omega f v \; d\Omega dt + \int_T \int_{\partial\Omega} (k \nabla u \cdot \vec{n}\,) v \; dS dt \tag{2.16}$$

can be written for a 2-dimensional element as:

$$\int_T \int_{\Omega^e} \frac{\partial u_h}{\partial t} v_h \; d\Omega dt + \int_T \int_{\Omega^e} \left[k \left(\frac{\partial u_h}{\partial x} \frac{\partial v_h}{\partial x} + \frac{\partial u_h}{\partial y} \frac{\partial v_h}{\partial y} \right) \right] \; d\Omega dt = \int_T \int_{\Omega^e} f v_h \; d\Omega dt + \int_T \int_{\partial\Omega^e} g v_h \; dS dt \tag{2.17}$$

Note that without the time integration, the second term on the left hand side and the right hand side of Equation 2.17 is already discretized in Equations 2.12 and 2.13. Thus, Equation 2.17 can also be written as:

$$\int_T \int_{\Omega^e} \frac{\partial u_h}{\partial t} v_h \; d\Omega dt + \int_T K_{ij}^e u_j^e \; dt = \int_T f_j^e \; dt \tag{2.18}$$

Here, using first order accurate Backward Euler Method as a time integrator, Equation 2.18 becomes the following:

$$\int_{\Omega^e} \frac{u_h^n - u_h^{n-1}}{\delta t} v_h \; d\Omega + K_{ij}^e u_j^{e,n} = f_j^{e,n} \tag{2.19}$$

Now, we generate a matrix, M_{ij}, to keep the information of first term on the left hand side of Equation 2.19:

$$M_{ij}^{e} = \int_{\Omega^e} \psi_i \psi_j \, d\Omega^e \tag{2.20}$$

Then, 2.18 can be written as:

$$\frac{M_{ij}^{e}\left(u_j^{e,n} - u_j^{e,n-1}\right)}{\delta t} + K_{ij}^{e} u_j^{e,n} = f_j^{e,n} \tag{2.21}$$

or,

$$\left(M_{ij}^{e} + \delta t K_{ij}^{e}\right) u_j^{e,n} = \delta t f_j^{e,n} + M_{ij}^{e} u_j^{e,n-1} \tag{2.22}$$

This equation is for one element only. However, collecting all the terms coming from all the elements, the following can be written for the problem:

$$\left(M_{ij} + \delta t K_{ij}\right) u_j^{n} = \delta t f_j^{n} + M_{ij} u_j^{n-1} \tag{2.23}$$

Above equation is in the form $A_{i,j} u_j^n = b_j$, where,

$$A_{ij} = M_{ij} + \delta t K_{ij} \tag{2.24}$$

$$b_j = \delta t f_j^{n} + M_{ij} u_j^{n-1} \tag{2.25}$$

However, one last step must be done in order to account for Dirichlet boundary condition. The procedure done for K_{ij} and f_j in Section 2.1.1 is done exactly to A_{ij} and b_j. Inverting $A_{ndbc,ndbc}$ and multiplying this inversion with b_{ndbc} gives the results for the unknown at nth time step, u_{ndbc}^{n}. Therefore, finally:

$$u_{ndbc}^{n} = A_{ndbc,ndbc}^{-1} b_{ndbc} \tag{2.26}$$

The overall solution algorithm for unsteady Poisson equation is presented below:

1) Set input variables such as geometry properties, shape parameter, source term, boundary conditions and initial condition.
2) Generate connectivity matrix.
3) Using Gaussian quadrature, create K_{ij}^e and M_{ij}^e for each element, and assemble K_{ij} and M_{ij} for the system.
4) Create the system matrix A_{ij} using Equation 2.22.
5) Reduce system matrix to impose Dirichlet boundaries and invert the reduced system matrix, $A_{ndbc,ndbc}$.
6) Start time loop.
    i) Assemble load vector b_j using Equation 2.23.
    ii) Edit the load vector b_j and reduce it to b_{ndbc}.
    iii) Solve for the unknown, u_j^n, by using Equation 2.26.
    iv) Change u_j^{n-1} to u_j^n and return (i).

2.1.3. Steady Stokes Equation

Considering the strong form of the steady Stokes equation with incompressibility condition:

$$-\nabla^2 \mathbf{u} + \nabla p = \mathbf{0} \tag{2.27}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.28}$$

This form can be reduced to a weak form by multiplying the Stokes equation with second order shape function, v, and the incompressibility condition with first order shape function, q. Thus, the weak form can be defined as:

$$\int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega = \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} dS \tag{2.29}$$

$$\int_\Omega (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{2.30}$$

We let,

$$\mathbf{u}_h = \sum_{j=1}^{N_u} \mathbf{u}_j \vec{\phi}_j \tag{2.31}$$

and,

$$p_h = \sum_{k=1}^{N_p} p_k \phi_k \tag{2.32}$$

where N_u is the total number of velocity nodes and N_p is the total number of pressure nodes. It should be noted here that we used second order elements to approximate $\mathbf{u}_h$ and first order elements to approximate the unknown p_h in order to deal with the stability issues, that may arise otherwise, and also to be able to solve for the two unknowns of the problem simultaneously. Then, we can write the Stokes equation system as such:

$$\begin{pmatrix} K & 0 & -L_x \\ 0 & K & -L_y \\ L_x^T & L_y^T & 0 \end{pmatrix} \begin{bmatrix} u_x \\ u_y \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.33}$$

where, K matrix is for the Laplacian part of the Stokes equation which is exactly same as explained in the previous sections. However, matrices L_x and L_y represent the new matrices which are the weak derivatives:

$$L_x = \int_\Omega \phi_k \frac{\partial \phi_j}{\partial x} \tag{2.34}$$

$$L_y = \int_\Omega \phi_k \frac{\partial \phi_j}{\partial y} \tag{2.35}$$

Since we are using a mixed space (i.e. $P1$-$P2$ elements together here), it should be noted that the dimensions of the matrix K is $N_u \times N_u$, where N_u is the total number of nodes for the second order elements. Additionally, the dimensions of L_x and L_y are $N_u \times N_p$ where N_p is the total number of nodes for the first order elements.

Moreover, the matrix defined in Equation 2.33 must also be reduced if we have Dirichlet boundary conditions defined in our system. If we have Neumann boundary conditions defined in the problem, then there is no issue about the uniqueness of the pressure solution since the information about the behaviour of the pressure unknown at at least one of the boundary node is known. In that case, the boundary conditions are applied likewise in previous sections. However, if no Neumann boundary conditions are defined in the problem, and if we have only pure Dirichlet boundary conditions defined for velocity (and no information about pressure unknown at the boundaries), then pressure is not uniquely defined. This issue can be dealt with defining a *fake* Dirichlet boundary condition for the pressure at the last node of the domain just for the sake of mathematical well-definedness of the problem, even though it has no physical meaning.

The overall solution algorithm for steady Stokes equation is as follows:

1) Set input variables such as geometry properties, shape parameter, source term, boundary conditions and initial condition.
2) Generate connectivity matrices of the first and the second order elements.
3) Using Gaussian quadrature, create elemental forms of K, L_x and L_y and assemble them.
4) Create the system matrix (the first term in the left hand side of Equation 2.33) and load vector (the right hand side of the Equation 2.33).
5) Edit system matrix and load vector to impose Dirichlet boundaries as explained above and invert the reduced system matrix.
6) Solve for $\mathbf{u}_j$ and p_j.

2.1.4. Unsteady Stokes Equation

Considering the strong form of the unsteady Stokes equation with the incompressibility condition is as follows:

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \tag{2.36}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.37}$$

Above strong form can be reduced to a weak form by applying the procedure done in Section 2.1.3:

$$\int_T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \; d\Omega dt + \int_T \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega dt - \int_T \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega dt$$
$$= \int_T \int_\Omega \mathbf{f} \cdot \mathbf{v} \; d\Omega dt + \int_T \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} dS dt \tag{2.38}$$

$$\int_\Omega (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{2.39}$$

Likewise in the steady case, let,

$$\mathbf{u}_h^n = \sum_{j=1}^{N_u} \mathbf{u}_j^n \vec{\phi}_j \tag{2.40}$$

and,

$$p_h^n = \sum_{k=1}^{N_p} p_k^n \phi_k \tag{2.41}$$

where superscript n denotes the time step. If we write the last two terms of the left hand side of Equation 2.38 as,

$$\int_T \left[\int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega \right] dt$$

then we can use the matrix in the Equation 2.33 for the term in the square brackets. Also for the right hand side of of Equation 2.38 can be dicretized as following without time integration:

$$\begin{bmatrix} \int_\Omega f_x\phi_j \ d\Omega + \int_{\partial\Omega} g_x\phi_j \ dS \\ \int_\Omega f_y\phi_j \ d\Omega + \int_{\partial\Omega} g_y\phi_j \ dS \\ 0 \end{bmatrix} = \begin{bmatrix} f_{x,j} \\ f_{y,j} \\ 0 \end{bmatrix}$$

where $(\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}) = [g_x, g_y]$. Now, using the Backward Euler Method, the first term on the left hand side of Equation 2.38 becomes,

$$\int_\Omega \left[\frac{\mathbf{u}^n - \mathbf{u}^{n-1}}{\delta t}\mathbf{v}\right] \ d\Omega$$

and discretizing it in space,

$$\frac{1}{\delta t}\begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{bmatrix} \mathbf{u}^n - \mathbf{u}^{n-1} \\ 0 \end{bmatrix}$$

where $M_{ij} = \int_\Omega \phi_i\phi_j \ d\Omega$. Finally, the total system can be written as such:

$$\begin{pmatrix} M+\delta tK & 0 & -\delta tL_x \\ 0 & M+\delta tK & -\delta tL_y \\ L_x^T & L_y^T & 0 \end{pmatrix}\begin{bmatrix} u_x \\ u_y \\ p \end{bmatrix}^n = \begin{bmatrix} \delta t f_{x,j} \\ \delta t f_{y,j} \\ 0 \end{bmatrix}^n + \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{bmatrix} u_x \\ u_y \\ p \end{bmatrix}^{n-1} \tag{2.42}$$

The discussion on the boundary conditions are exactly the same as in the previous sections.

The overall solution algorithm for unsteady Stokes equation with incompressibility condition is presented below:

1) Set input variables such as geometry properties, source term, boundary conditions and initial condition.
2) Generate connectivity matrices of the first and the second order elements.
3) Using Gaussian quadrature, create elemental forms of M, K, L_x and L_y and assemble them.
4) Using matrices created in previous step, create the global system matrix (the matrix at the left hand side of Equation 2.42), reduce the system matrix to apply Dirichlet boundary conditions and invert the reduced system matrix.
5) Start time loop.
   i) Create load vector (the right hand side of Equation 2.42).
   ii) Edit the load vector .
   iii) Solve for solution vector(the vector at the left hand side of Equation 2.42) s^n by using Equation 2.42.
   iv) Change s^{n-1} to s^n and return (i).

2.2. Error Calculation

Since the aim of this thesis is to compare FEM with meshless RBFCM method, one of the very important performance criteria is how those methods perform considering the well-known type of errors. Therefore in this section, least square error (LSE), root mean square error (RMSE) and maximum relative error (MRE) calculations for FEM is explained. Moreover, a brief explanation about h-convergence is given at the end of LSE discussion. In general, least square error analyses are done for FEM in the literature and RMSE analyses are done for RBFCM. Thus, this study compares FEM and RBFCM with respect to these three types of errors.

First of all, LSE is a type of error which indicates the proximity of the analytical surface and the approximated surface (see Figure 3.2). Considering a 2-dimensional

element, the formulation of LSE is as follows:

$$LSE = \sum_{k=1}^{E} LSE_k^e \tag{2.43}$$

where E is the total number of elements and,

$$LSE^e = \left(\int_{\Omega_e} (u_h(x,y) - u(x,y))^2 \, \partial\Omega \right)^{0.5} \tag{2.44}$$

for an arbitrary element. To evaluate the integral appearing in Equation 2.44 numerically, the Gaussian quadrature technique is implemented. Thus, LSE^e is calculated as such:

$$LSE^e = \left(\sum_{p=1}^{P} (u_j \phi_j(x_p, y_p) - u(x_p, y_p))^2 \right)^{0.5} \tag{2.45}$$

where P is the total number of quadrature points, $u(x_p, y_p)$ is the analytical solution at pth quadrature point and $u_j \phi_j(x_p, y_p)$ is the approximated solution at that quadrature point.

The solutions obtained by FEM must become more accurate by increasing the number of elements in the domain, or in other words, by using finer mesh, and hence more nodes. This means that the solution must converge to a single correct solution and since this is done by refining the mesh, it is called *h-convergence*[5] . Moreover, the h-convergence of the solution must be a exponential function governed by the equation below:

$$\log(LSE) = m \log(\delta h) + b \tag{2.46}$$

where m is the slope of the curve in the logarithmic scale. If the solution is error-free from other parameters such as the insufficiency of time integrator or the insufficiency

[5]In FEM literature the letter h is often used to refer to the size of the element in the mesh, hence the name, *h-convergence*

of the order of Gaussian quadrature, the slope, m, must be an order higher than we have used for approximation. This means m must be equal to 3 for FEM-P2 and 2 for FEM-P1. It is possible for reader to see these slope values in all of examples in Chapter 4, confirming the accuracy of our models.

RMSE, on the other hand, is the standard deviation of the errors in the domain. Thus, rather than indicating the proximity of the approximated surface and the analytical surface, RMSE indicates how close the approximated solution at each node to the analytical solution at that node. After solving the differential equations, the vector u_j keeps the approximated values at the solution nodes. Therefore, the RMSE calculation is as follows:

$$RMSE = \sum_{j=1}^{N} \sqrt{\frac{(u_j - u(x_j, y_j))^2}{N}} \tag{2.47}$$

Lastly, the maximum relative error indicates the error at the node where the approximated value deviates most from the analytical solution (see Figure 3.3), and it is calculated by the equation below:

$$MRE = max\left(\left|\frac{u_j - u(x_j, y_j)}{u(x_j, y_j)}\right|\right) \tag{2.48}$$

3. RADIAL BASIS FUNCTION COLLOCATION METHOD

Radial Basis Function Collocation Method is a true meshless numerical method which utilizes points distributed uniformly, non-uniformly or randomly throughout the domain rather than utilizing a mesh and a mesh connection. This nature of RBFCM makes it distinct from other classical methods such as FDM, FVM or FEM. However, there are some disadvantages mainly due to poor-condition of the system matrices generated by RBFs [34]. In spite of these disadvantages, algorithms dealing with the matrix inversion are getting better day by day. Additionally, there are a lot of studies trying to improve the poor-conditioning of RBF based methods such as the Localized Collocation Meshless Method(LCMM) [31–33] or Radial Basis Function Finite Difference(RBF-FD) [46]. Besides, only by using Gaussian RBF, for example, one can improve the condition of the matrix since the effect of distant nodes diminishes to almost 0. However, "RBFCM" in this work only refers to Kansa's method (also called globalized RBFCM in the literature) and multiquadrics are used in general.

Examples of commonly used type of RBFs:

- Multiquadric: $\varphi(r) = (r^2 + c^2)^{\beta/2},\ \beta > 0 \beta \epsilon 2N + 1$
- Inverse Multiquadric: $\varphi(r) = (r^2 + c^2)^{-\beta/2},\ \beta > 0, \beta \epsilon 2N + 1$
- Splines: $\varphi(r) = r^{\beta},\ \beta > 0, \beta \epsilon 2N + 1$
- Thin Plate Splines: $\varphi(r) = r^{\beta} \ln r,\ \beta > 0, \beta \epsilon 2N$
- Gaussian: $\exp -c^2 r^2$

where r is the Euclidean distance between two points and c is the shape parameter.

Hardy in [19,20] used only multiquadrics in his basic scheme. However, the scheme can be written for all RBFs. Let $u(x)$ be any function, then $u(x)$ can be written as a

collection of N continuously differentiable RBFs, $\varphi_j(x)$, as such:

$$u(x) = \sum_{j=1}^{N} \alpha_j \varphi_j(x - x_j) \tag{3.1}$$

where α_js are the coefficients for each RBF $\varphi_j(x)$ and they are found by solving the set of linear equations.[6]

Kansa, in [24, 25], expanded this basic scheme to approximate partial derivatives and partial differential equations. Partial derivatives of any function $u(x, y)$ can be written as:

$$\frac{\partial^n}{\partial x^n}[u(x,y)] = \sum_{j=1}^{N} \alpha_j \frac{\partial^n}{\partial x^n}[\varphi_j(x - x_j, y - y_j)] \tag{3.2}$$

$$\frac{\partial^n}{\partial y^n}[u(x,y)] = \sum_{j=1}^{N} \alpha_j \frac{\partial^n}{\partial y^n}[\varphi_j(x - x_j, y - y_j)] \tag{3.3}$$

$$\frac{\partial^{n+m}}{\partial x^n \partial y^m}[u(x,y)] = \sum_{j=1}^{N} \alpha_j \frac{\partial^{n+m}}{\partial x^n \partial y^m}[\varphi_j(x - x_j, y - y_j)] \tag{3.4}$$

where n and m are positive integers and N is the total number of nodes. Further, the reader may refer to Appendix C for the full list of partial derivatives of MQ and TPS. These two types of RBFs are the ones we generally use in this work.

Now, any linear differential equation can be interpreted in a matrix form as such:

$$A\alpha = b \tag{3.5}$$

or,

$$\alpha = A^{-1}b \tag{3.6}$$

[6]Various iterative techniques dealing with non-linear equations can also be implemented in this scheme.

where A_{ij} is $N \times N$ system matrix[7] , α_j is $N \times 1$ coefficient vector and b_j is $N \times 1$ load vector. After solving the system for α, the unknown can be found by:

$$u_j = \Phi_{ij}\alpha_j \tag{3.7}$$

where,

$$\Phi = \begin{pmatrix} \varphi_{11} & \varphi_{21} & \cdots & \varphi_{N1} \\ \varphi_{12} & \varphi_{22} & \cdots & \varphi_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{1N} & \varphi_{2N} & \cdots & \varphi_{NN} \end{pmatrix}$$

and also, φ_{ij} for MQ($\beta = 1$) and TPS respectively:

$$\varphi_{ij} = (r_{ij}^2 + c^2)^{\beta/2} \tag{3.8}$$

$$\varphi_{ij} = r_{ij}^{\beta} \ln r_{ij} \tag{3.9}$$

where $r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

3.1. Numerical Formulation

3.1.1. Steady Poisson Equation

Considering 2-dimensional steady Poisson Equation:

$$\nabla \cdot (-k\nabla u) = f \text{ in } \Omega, \tag{3.10}$$

$$u = u_D \text{ on } \Gamma_D, \tag{3.11}$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N \tag{3.12}$$

[7]The system matrix varies with differential equation and boundary conditions. Assemble details of system matrix is explained in Section 3.1.

In partial form, Equation 3.10 can be written as:

$$-k(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = f \text{ in } \Omega \tag{3.13}$$

assuming k is a constant.

Plugging suitable one of the Equations 3.1-3.4 in Equation 3.13:

$$-k(\sum_{j=1}^{N} \alpha_j \frac{\partial^2}{\partial x^2}[\varphi_j(x - x_j, y - y_j)] + \sum_{j=1}^{N} \alpha_j \frac{\partial^2}{\partial y^2}[\varphi_j(x - x_j, y - y_j)]) = f \tag{3.14}$$

Finally, the system matrix A, and load vector b can be assembled as:

$$A = \begin{pmatrix} \Phi_{I\times N} \\ \Phi^{\vec{n}}_{J\times N} \\ -k(\Phi^{xx} + \Phi^{yy})_{K\times N} \end{pmatrix}, b = \begin{pmatrix} u_D \\ g \\ f \end{pmatrix} \begin{matrix} \text{on } \Gamma_D \\ \text{on } \Gamma_N \\ \text{in } \Omega \end{matrix}$$

where I, J, K are the number of points on Dirichlet boundaries, Neumann boundaries and inside the domain respectively and they add up to total number of points, N. Also, derivatives are denoted as superscripts.

The solution algorithm for steady Poisson equation is presented below:

1) Set input variables such as geometry properties, shape parameter, source term and boundary conditions.
2) Generate RBF matrix Φ and its derivatives.
3) Assemble system matrix and load vector.
4) Invert system matrix and solve for α by using Equation 3.6.
5) Solve for unknown u_j by using Equation 3.7.

3.1.2. Unsteady Poisson Equation

Considering 2-dimensional unsteady Poisson equation:

$$\frac{\partial u}{\partial t} + \nabla \cdot (-k\nabla u) = f \text{ in } \Omega \times (0,T], \tag{3.15}$$

$$u = u_D \text{ on } \Gamma_D \times (0,T], \tag{3.16}$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N \times (0,T], \tag{3.17}$$

$$u = u_i \text{ at } t = 0. \tag{3.18}$$

Assuming k is a constant, Equation 3.15 can be written in partial form:

$$\frac{\partial u}{\partial t} - k(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = f \tag{3.19}$$

Using first order accurate Backward Euler Method, Equation 3.19 can be discretized as:

$$\frac{u^n - u^{n-1}}{\delta t} - k(\frac{\partial^2 u^n}{\partial x^2} + \frac{\partial^2 u^n}{\partial y^2}) = f^n \tag{3.20}$$

where δt is the time step and superscript n denotes the time step. Further, Equation 3.20 can be discretized as:

$$\Phi_{ij}(\alpha_j^n - \alpha_j^{n-1}) - \delta tk(\Phi_{ij}^{xx} + \Phi_{ij}^{yy})\alpha_j^n = \delta t f_j^n \tag{3.21}$$

$$(\Phi_{ij} - \delta tk(\Phi_{ij}^{xx} + \Phi_{ij}^{yy}))\alpha_j^n = \delta t f_j^n + \Phi_{ij}\alpha_j^{n-1} \tag{3.22}$$

Note that $\Phi_{ij}\alpha_j^{n-1} = u^{n-1}$. Finally, we have the form $A\alpha^n = b$ where,

$$A = \begin{pmatrix} \Phi_{I\times N} \\ \Phi^{\vec{n}}_{J\times N} \\ (\Phi - \delta tk(\Phi^{xx} + \Phi^{yy}))_{K\times N} \end{pmatrix}, b = \begin{pmatrix} u_D^n \\ g^n \\ \delta t f^n + u^{n-1} \end{pmatrix} \begin{matrix} \text{on } \Gamma_D \\ \text{on } \Gamma_N \\ \text{in } \Omega \end{matrix}$$

The overall solution algorithm for unsteady Poisson equation is presented below:

1) Set input variables such as geometry properties, shape parameter, source term, boundary conditions and initial condition.
2) Generate RBF matrix Φ and its derivatives.
3) Assemble system matrix A and invert it.
4) Start time loop.
   i) Assemble load vector b by known u_D^n, g^n, f^n and u^{n-1} values.
   ii) Solve for α^n by using 3.6.
   iii) Solve for unknown, u^n, by using Equation 3.7.
   iv) Change u^{n-1} to u^n and return (i).

3.1.3. Steady Stokes Equation

Considering 2-dimensional steady Stokes Equation with incompressibility condition:

$$-\nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \tag{3.23}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.24}$$

the partial form can be written as:

$$-\frac{\partial^2 u_x}{\partial x^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial p}{\partial x} = f_x \tag{3.25}$$

$$-\frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial p}{\partial y} = f_y \tag{3.26}$$

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \tag{3.27}$$

Plugging in the Equations 3.1-3.4 in Equations 3.25-3.27:

$$-\left(\sum_{j=1}^{N}\alpha_j^{u_x}\frac{\partial^2\varphi_j}{\partial x^2}+\sum_{j=1}^{N}\alpha_j^{u_x}\frac{\partial^2\varphi_j}{\partial y^2}\right)+\sum_{j=1}^{N}\alpha_j^{p}\frac{\partial\varphi_j}{\partial x}=f_x \tag{3.28}$$

$$-\left(\sum_{j=1}^{N}\alpha_j^{u_y}\frac{\partial^2\varphi_j}{\partial x^2}+\sum_{j=1}^{N}\alpha_j^{u_y}\frac{\partial^2\varphi_j}{\partial y^2}\right)+\sum_{j=1}^{N}\alpha_j^{p}\frac{\partial\varphi_j}{\partial y}=f_y \tag{3.29}$$

$$\sum_{j=1}^{N}\alpha_j^{u_x}\frac{\partial\varphi_j}{\partial x}+\sum_{j=1}^{N}\alpha_j^{u_y}\frac{\partial\varphi_j}{\partial y}=0 \tag{3.30}$$

Here, assuming the solution vector, s, and coefficient matrix, α, as:

$$s_{3N\times1}=\left(\begin{array}{c}u_x\\\hline u_y\\\hline p\end{array}\right),\ \alpha_{3N\times1}=\left(\begin{array}{c}\alpha^{u_x}\\\hline \alpha^{u_y}\\\hline \alpha^{p}\end{array}\right)$$

the first order RBF matrix can be written as such:

$$\Phi_{3N\times3N}=\left(\begin{array}{c|c|c}\Phi_{N\times N}&0&0\\\hline 0&\Phi_{N\times N}&0\\\hline 0&0&\Phi_{N\times N}\end{array}\right)$$

Now, the RBFCM discretization like Equation 3.7 can be written for the solution vector:

$$s=\Phi_{3N\times3N}\alpha \tag{3.31}$$

To find the coefficient matrix α we need to assemble the system matrix, A, and load vector, b. Using Equations 3.28-3.30, A and b before applying the boundary conditions can be assembled as such:

$$A_{3N\times3N}=\left(\begin{array}{c|c|c}-(\Phi^{xx}+\Phi^{yy})_{N\times N}&0&\Phi^{x}_{N\times N}\\\hline 0&-(\Phi^{xx}+\Phi^{yy})_{N\times N}&\Phi^{y}_{N\times N}\\\hline -(\Phi^{x}_{N\times N})^T&-(\Phi^{y}_{N\times N})^T&0\end{array}\right),\ b=\left(\begin{array}{c}f_x\\\hline f_y\\\hline 0\end{array}\right)$$

Of course, it is important to edit A and b to impose boundary conditions. In the case of existence of Neumann boundary conditions, A and b is edited very much like in Section 3.1.3. However, if no Neumann boundary condition is defined in the problem, we impose an artificial Dirichlet boundary condition for pressure to define pressure uniquely.

An example of a system matrix assembled for 3 nodes is presented below for the reader to capture the structure of the matrix better. Assuming at the nodes 1 and 2, Dirichlet boundary conditions and natural boundary conditions are defined respectively, and unit normal, $\vec{n} = [1, 0]$ at the natural boundary:

$$A = \left(\begin{array}{ccc|ccc|ccc}
\varphi_{11} & \varphi_{12} & \varphi_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\
\varphi^x_{21} & \varphi^x_{22} & \varphi^x_{23} & 0 & 0 & 0 & -\varphi_{21} & -\varphi_{22} & -\varphi_{23} \\
-\varphi^{xx}_{31} - \varphi^{yy}_{31} & -\varphi^{xx}_{32} - \varphi^{yy}_{32} & -\varphi^{xx}_{33} - \varphi^{yy}_{33} & 0 & 0 & 0 & \varphi^x_{31} & \varphi^x_{32} & \varphi^x_{33} \\
\hline
0 & 0 & 0 & \varphi_{11} & \varphi_{12} & \varphi_{13} & 0 & 0 & 0 \\
0 & 0 & 0 & \varphi^x_{21} & \varphi^x_{22} & \varphi^x_{23} & 0 & 0 & 0 \\
0 & 0 & 0 & -\varphi^{xx}_{31} - \varphi^{yy}_{31} & -\varphi^{xx}_{32} - \varphi^{yy}_{32} & -\varphi^{xx}_{33} - \varphi^{yy}_{33} & \varphi^x_{31} & \varphi^x_{32} & \varphi^x_{33} \\
\hline
-\varphi^x_{11} & -\varphi^x_{21} & -\varphi^x_{31} & -\varphi^y_{11} & -\varphi^y_{21} & -\varphi^y_{31} & 0 & 0 & 0 \\
-\varphi^x_{12} & -\varphi^x_{22} & -\varphi^x_{32} & -\varphi^y_{12} & -\varphi^y_{22} & -\varphi^y_{32} & 0 & 0 & 0 \\
-\varphi^x_{13} & -\varphi^x_{23} & -\varphi^x_{33} & -\varphi^y_{13} & -\varphi^y_{23} & -\varphi^y_{33} & 0 & 0 & 0
\end{array}\right)$$

and also the load vector, b, is:

$$b = \left(\begin{array}{c}
u_{xD} \\ 0 \\ \hline f_x \\ \hline u_{yD} \\ 0 \\ \hline f_y \\ \hline 0 \\ 0 \\ 0
\end{array}\right)$$

The solution algorithm for steady Stokes equation with incompressibility condition is explained below:

1) Set input variables such as geometry properties, shape parameter, source term and boundary conditions.
2) Generate RBF matrix Φ and its derivatives.
3) Assemble system matrix, A, and load vector, b.
4) Invert system matrix and solve for α by using Equation 3.6.
5) Solve for solution vector s by using Equation 3.31.

3.1.4. Unsteady Stokes Equation

Consider 2-dimensional unsteady Stokes equation with incompressibility condition:

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \tag{3.32}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3.33}$$

the partial form can be written as:

$$\frac{\partial u_x}{\partial t} - \frac{\partial^2 u_x}{\partial x^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial p}{\partial x} = f_x \tag{3.34}$$

$$\frac{\partial u_y}{\partial t} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial p}{\partial y} = f_y \tag{3.35}$$

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \tag{3.36}$$

Using Backward Euler Method, the first terms of Equations 3.34 and 3.35 discretized as:

$$\frac{\Phi \alpha^{u_x,n} - \Phi \alpha^{u_x,n-1}}{\delta t} \tag{3.37}$$

$$\frac{\Phi \alpha^{u_y,n} - \Phi \alpha^{u_y,n-1}}{\delta t} \tag{3.38}$$

Let M be a matrix for time discretization of the system. Then:

$$M_{3N\times 3N} = \left(\begin{array}{c|c|c} \Phi_{N\times N} & 0 & 0 \\ \hline 0 & \Phi_{N\times N} & 0 \\ \hline 0 & 0 & 0 \end{array}\right)$$

holds the time information of Equations 3.37 and 3.38.

The $-\nabla^2\mathbf{u}+\nabla p$ part is already discretized in the previous section. Using $A_{3N\times 3N}$ matrix and $b_{3N\times 1}$ vector, the Stokes system can be written as such:

$$(M + \delta t A)\alpha^n = \delta t b^n + M\alpha^{n-1} \tag{3.39}$$

Now, Equation 3.39 can be written in the form of Equation 3.6:

$$\alpha^n = K^{-1}z \tag{3.40}$$

where $K = M + \delta t A$ and $z = \delta t b^n + M\alpha^{n-1}$. After solving for α, we can finally find the solution vector of the next time step by:

$$s^n = \Phi_{(3N\times 3N)}\alpha^n \tag{3.41}$$

where first, second and third N entries of s^n are the u_x, u_y and p values respectively at nth time step.

The overall solution algorithm for unsteady Stokes equation with incompressibility condition is presented below:

1) Set input variables such as geometry properties, shape parameter, source term, boundary conditions and initial condition.
2) Generate RBF matrix Φ and its derivatives.

3) Assemble space discretization matrix, A, time discretization matrix M.
4) Assemle the system matrix K using A and M, and invert it.
5) Start time loop.
   i) Assemble load vector z.
   ii) Solve for α^n by using Equation 3.40.
   iii) Solve for solution vector, s^n, by using Equation 3.41.
   iv) Change s^{n-1} to s^n and return (i).

3.2. Error Calculation

Error calculation plays a major role in this work since the main goal of this work is to compare FEM and RBFCM. Therefore in this section, least squares error, root mean square error and maximum relative error calculations for RBFCM is explained. Generally, root mean square analyses are done for RBFCM approximations in the literature. However, least square error analyses occupy most of the studies in FEM. Therefore, it is more convenient to do both LSE and RMSE analyses for both numerical methods.

Since LSE analyses lack in RBFCM literature, we develop a calculation method which imitates the LSE analysis done for FEM. In this method, novel *artificial elements* that corresponds to real elements in FEM are generated in the RBFCM domain (see Figure 3.2). For each artifical element, volume integral for analytical surface and approximated surface is calculated using Gaussian quadratures. Later, square differences of these volumes are summed for each artificial elements and the summation is square rooted to calculate LSE. The reader may refer to Figure 3.2 to visualize these volumes.

Volume integral of approximated surface is calculated by using quadrature points as RBF centers. Assuming a quadrature point has coordinates (x_p, y_p), then approxi-

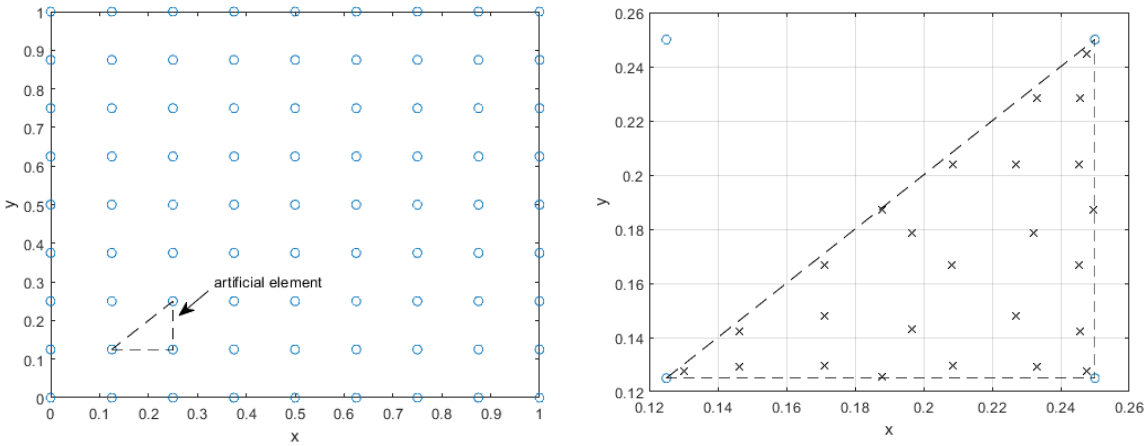

Figure 3.1. An example of artificial element in RBFCM domain (left), Gaussian quadrature points (right), $\delta h = 0.125$.

mated function $u_A(x_p, y_p)$ is:

$$u_A(x_p, y_p) = \sum_{j=1}^{N} \alpha_j \varphi_j(x_p - x_j, y_p - y_j) \tag{3.42}$$

Then, volume integral of the artificial element is calculated by:

$$Volume = \sum_{k=1}^{E} w_k u_{Ak} \tag{3.43}$$

where E, is the total number of quadrature points and w_k is the Gaussian weights. After finding the volumes, standard LSE calculation procedure is followed.

LSE shows whether the approximated surface is close to analytical surface or not rather than showing whether the results approximated are close to analytical solution at the solution nodes like RMSE.

After solving the differential equations, solution vectors holds the values for the approximated solutions at the RBF centers. In all examples of the next chapter, RBF centers are chosen such that they coincide with the collocation nodes. Hence, solution vectors keep the approximated solutions for each point in the domain. Square

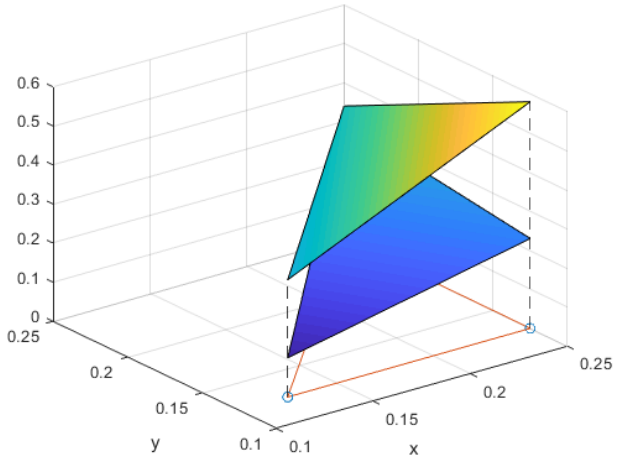

Figure 3.2. An example of analytic surface and approximated surface.

differences of solution vectors and analytical solutions at the nodes are calculated. Then these differences summed, divided by total number of nodes and square rooted to calculate RMSE. The equation for RMSE calculation is:

$$RMSE = \sum_{j=1}^{N} \sqrt{\frac{(u_{Aj} - u(x_j, y_j))^2}{N}} \tag{3.44}$$

where N is the total number of nodes[8] , u_A is the approximate solution, $u(x, y)$ is the analytical solution.

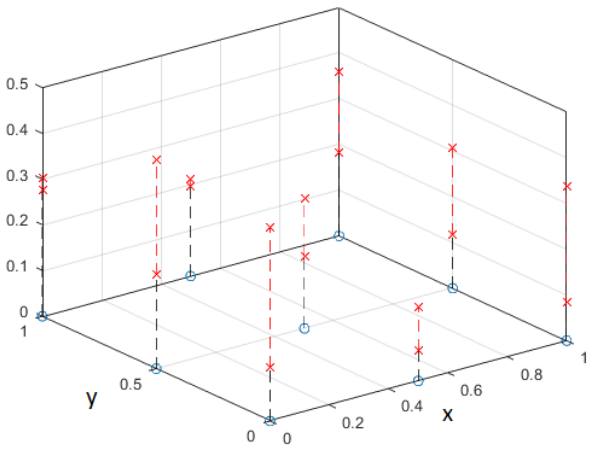

Figure 3.3. Examples of $|u_{Aj} - u(x_j, y_j)|$ (Dashed red lines).

[8]RMSE calculations include boundary nodes.

Finally, relative errors are calculated using the equation:

$$RelativeError = \left| \frac{u_{Aj} - u(x_j, y_j)}{u(x_j, y_j)} \right| \tag{3.45}$$

and maximum is found by comparing each relative error for all the nodes except boundaries.

3.3. Shape Factor Optimization

The shape parameter, c, is an arbitrary constant which has a major role in the results obtained by RBFCM-MQ [19,20]. It appears in the formulations of several type of RBFs such as Multiquadrics or Gaussian RBF[9] . Since the approximated solution of, say RBFCM-MQ, mainly depends on the shape factor used, there are a lot of studies dealing with the optimization of it. However, the optimum shape parameter depends on many parameters of the problem such as properties of geometry, domain and the differential equation itself, making the optimum shape parameter hard to find. In [47] for example, tedious mathematical work is done to estimate local errors a priori using Fourier transforms on RBFs, and thus, having an idea about the behaviour of the optimum shape parameter. However, the mathematical advancements in this topic is not enough to determine the optimum shape parameter before solving the problem. Indeed, it is claimed in [67, 68] that the calculus based schemes to optimize shape parameter do not guarantee the global minima. Additionally, there are various shape parameters proposed in [19, 42, 43]. Unfortunately, these shape parameters are not necessarily yield the best results, even though they have a good estimate on analytical solution. Therefore, to make the comparison more fair, getting the best approximation out of RBFCM-MQ by determinining the optimum shape parameter was one of the crucial parts of this work.

Before getting into the algorithm that seeks for the optimum shape parameter used in this work, it is impotant to state some facts about the relation of error and the

[9]The other RBFs, such as TPS, are not dependent on shape factor.

shape parameter. Firstly, it is known the error is inversely proportional to the shape parameter and the condition number of the system matrix. This inverse proportionality is called "Schaback's uncertainty relation" and proven in [48]. Schaback's uncertainty relation dictates the shape parameter to be a relatively large number. However, choosing the shape parameter too large also increases the error in the solution approximation due to the errors appeared in the system matrix inversion. An immediate result of this is that the error decreases up to a point where the inversion errors are tolerable. Then, the error goes up because of the inversion errors[10] . Thus, it can be claimed that there exists a global minima in the error function of shape parameter.

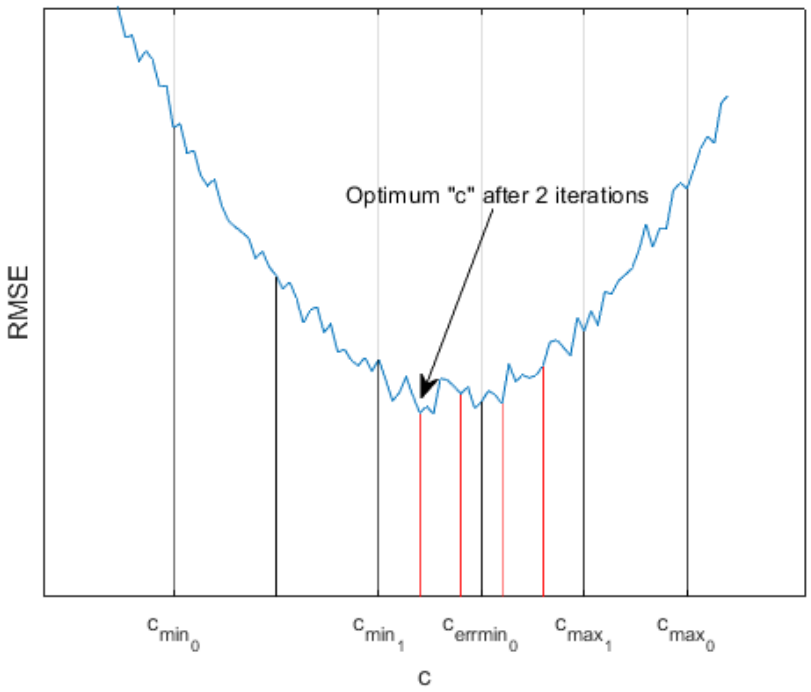

Figure 3.4. Plot of root mean square error function vs. shape parameter.

To determine the optimum shape parameter, we seek the global minimum of the error function. Firstly, c_{min} and c_{max} is determined such that the global minimum lies between these values. The interval between these values are divided to P smaller intervals to create middle c values. Then, a solution is found for each c_i where $c_0 = c_{min}$ and $c_P = c_{max}$. c_i value which gives the minimum RMSE results is chosen and the interval between $c_{err_{min}+1}$ and $c_{err_{min}-1}$ is also divided to P smaller intervals. This process is

[10]The error function of the shape parameter is not a smooth curve. There exists local fluctiations additionally.

followed up to point where going further made not much difference in accuracy.

Unfortunately, it is more likely that this optimization technique will find the local minima rather than the global minima since the error function is not always convex even if we omit the local fluctiations. We found that the optimized shape factor that this technique gives is dependant on c_{min} and c_{max} values, meaning that the optimized shape factor is likely to be a local minima. This problem is also persisted in [26], where Kansa and Carlson tried to optimize the shape factor using RMSE. However, we observed from our solutions, in general, that changing c_{min} and c_{max} value changed the RMSE output inconsiderably.

The solution algorithm used to find the optimum shape parameter is explained below:

1) Determine c_{min} and c_{max} such that the global minimum lies between c_{min} and c_{max}.
2) Divide the interval between c_{min} and c_{max} P smaller intervals.
3) Solve the problem for each c_i and find $c_{err_{min}}$.
4) If going further,
    i) does not improve accuracy, then stop.
    ii) improves accuracy, let $c_{max} = c_{err_{min}+1}$, also let $c_{min} = c_{err_{min}-1}$ and return (2).

4. NUMERICAL EXAMPLES

Performances of both FEM and RBFCM are tested over various Poisson and Stokes examples including Dirichlet and Neumann type boundary conditions. The domains used are generally unit square. However, the features of the domains are given explicitly for each case. In Examples 4.1.1 through 4.1.3, the steady Poisson Equation is taken into consideration, whereas for Examples 4.1.4, through 4.1.7 time dependent modelling is done for the same equation. Examples 4.2.1 and 4.2.2 are the problems of steady Stokes equation with incompressibility condition and Examples 4.2.3 and 4.2.4 are the time dependent cases for the same equation system. At the end of each section, the results for each comparison parameters are discussed.

For the finite element approximation, linear and second order test functions are used. For RBFCM, multiquadric radial basis functions are used in general[11] and the shape parameter is optimized for minimum root mean square error (see Section 3.2). Since analytical solution is essential to optimize the shape factor, Example 4.1.7, which lacks such an analytical solution, does not include the optimal results for RBFCM model. Therefore, widely used shape factors proposed in [19, 42, 43] for MQRBF are applied. Because of the same reason, the error analyses are not included in that example. Rather, it includes the results obtained from a commercial finite element software and the reader may visually confirm the proximity of all the results in Figure 4.17. An improved version of RBFCM, called symmetrical collocation (RBFCM(SC)), is implemented in the Example 4.1.2. Additionally, to show the true meshless nature of RBFCM, in the examples where the L-shaped domain is used include non-uniform, randomly distributed nodes.

Starting from $\delta h = 1/4$ down to $\delta h = 1/32$, mesh is refined four times in all examples which the analytical solution is available to observe h-convergence[12] for each

[11] Thin plate splines are also utilized as radial basis functions in most of the examples.
[12] h-convergence is explain in Section 2.2.

method.[13] It was not possible to refine the mesh further due to the both memory(RAM) and runtime reasons. H-convergence results for FEM also serves as validation of our in house Python code. Unfortunately for the RBFCM-MQ, studies focusing on convergence properties of MQ do not have decisive conclusions.

Some remarks before analysing the results for examples must be given here, since these issues persist in all of the examples. First of all, the degree of freedom, or number of nodes we solve for, for the same δh value is higher for FEM-P2 approximations compared to other methods. The increase in the degrees of freedom, as δh gets smaller, is also bigger for FEM-P2. An immediate result of this situation is that FEM-P2 has the best convergence rate. However, having more degrees of freedom has its cost in runtime. Therefore, the runtime of FEM-P2 is generally higher compared to other methods. The second remark is about the runtime of RBFCM-MQ. Because of the optimizing algorithm explained in the Section 3.3, the code for RBFCM-MQ runs many times more than the others. Nevertheless, using prescribed c values, rather than implementing an algorithm for optimization, would yield much smaller runtime for RBFCM-MQ. In fact, using prescribed c values would end up very close runtime to our model using thin plate splines as RBFs (RBFCM-TPS) rather than using multiquadrics, since the difference between the two is only the generation of matrices of RBFs and its derivatives. One last remark must be made about the condition numbers. As indicated in the Chapter 3, the condition numbers of the system matrices generated with multiquadrics is always higher than the others. Even though having such a higher condition number, RBFCM-MQ yielded the best results on average after FEM-P2.

4.1. Poisson Equation

Poisson equation is a second order partial differential equation. It has a wide range of use in physical problems. It can be used to model gravitational field of Newtonian gravity, heat transfer in fluid mechanics and electrical potential in electrostatics. Each of these problems have different coefficients in front of the Laplacian operator.

[13] Interval lengths in x-direction, δx, and in y-direction, δy, are set to equal and referred as δh.

However, the mathematics and the behaviour of the variables in these problems are analogous with each other. Poisson equation is one of the simplest example of elliptic partial differential equation.

The problem definition in the strong form is given in each example in the upcoming sections, since the problems slightly differ from each other.

4.1.1. Steady Poisson Equation with Dirichlet Boundary Conditions

For the domain $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega = \Gamma_D$, the strong form of Poisson Equation is:

$$\nabla \cdot (-k\nabla u) = f \text{ in } \Omega, \tag{4.1}$$

$$u = u_D \text{ on } \Gamma_D, \tag{4.2}$$

where Ω is a unit square and all boundaries are of Dirichlet type. And the weak form is:

$$\int_\Omega k\nabla u \cdot \nabla v \; d\Omega = \int_\Omega f v \; d\Omega \tag{4.3}$$

For this example, the analytical equation is set to a simple trigonometric function:

$$u(x, y) = \sin(\pi x)\cos(\pi y/2), \tag{4.4}$$

and therefore, by setting $k = 1$, the source term f becomes:

$$f = \frac{5}{4}\pi^2 \sin(\pi x)\cos(\pi y/2)$$

To build the RBFCM model on the other hand, the strong form is reduced to partial form:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f, \tag{4.5}$$

$$u|_{x,y=0} = \sin(\pi x) \text{ and } u|_{\Gamma_D/\{(x,0)\}} = 0. \tag{4.6}$$

The computational domains for both FEM and RBFCM are shown in Figure 4.1.1. In Figure 4.2, the plot of analytical solution of Equation 4.4 is shown.

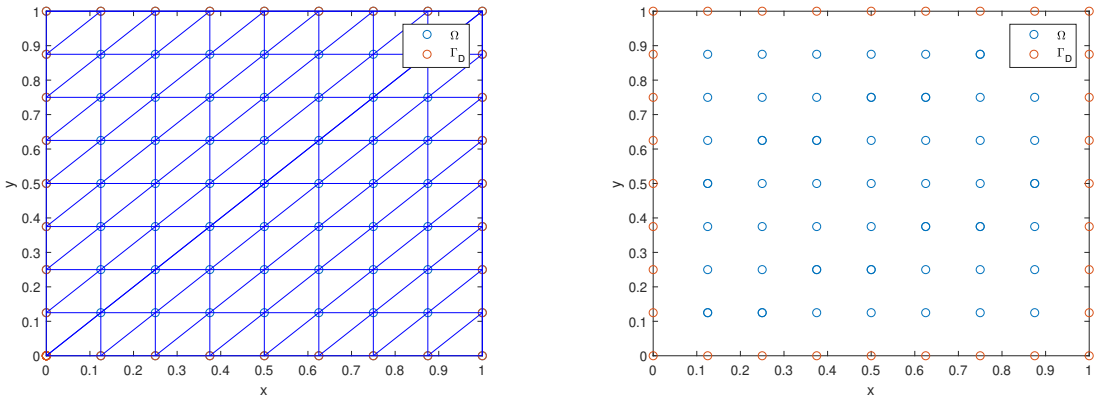

Figure 4.1. Computational domains of the example 4.1.1: first order FEM(P1) elements (left), RBFCM nodes (right), $\delta h = 0.125$.

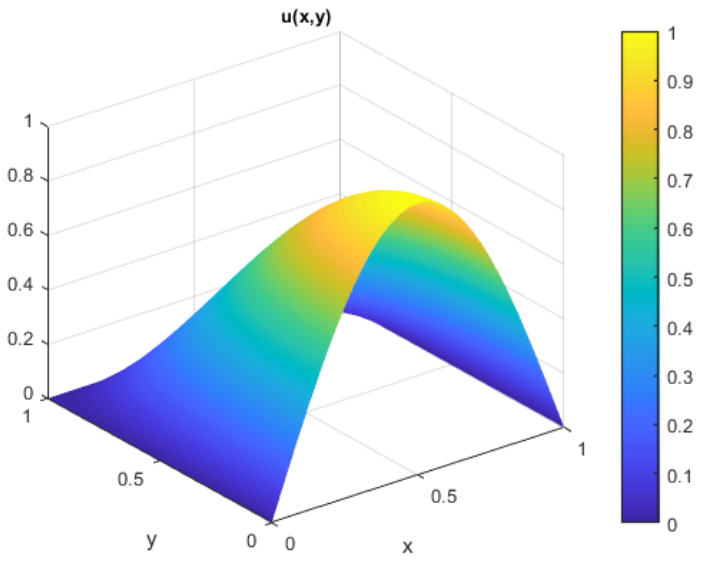

Figure 4.2. Plot of analytic solution of the example 4.1.1.

The least square error results of this example (see Figure 4.3 and Table 4.2) indicates firstly that both RBFCM-MQ and RBFCM-TPS perform better than FEM-P1 for every δh. The convergence rate (the slopes in the Figure 4.3) of RBFCM-MQ is the worst among all of the methods. However, since the shape factor, c, is optimized, it yields the best result at $\delta h = 1/4$. Despite having a low convergence rate, the only method that outruns RBFCM-MQ was FEM-P2 at $\delta h = 1/32$. Considering the fact that FEM-P2 has much higher degrees of freedom and its degree of freedom increases more than the other methods as δh getting smaller and smaller, FEM-P2 outrunning the other methods can easily be expected at smaller δh, *i.e.* for a finer mesh. Additionally, the degree of freedom of RBFCM-TPS is less than that of FEM-P2. Despite this fact, the convergence of RBFCM-TPS is very close to FEM-P2.

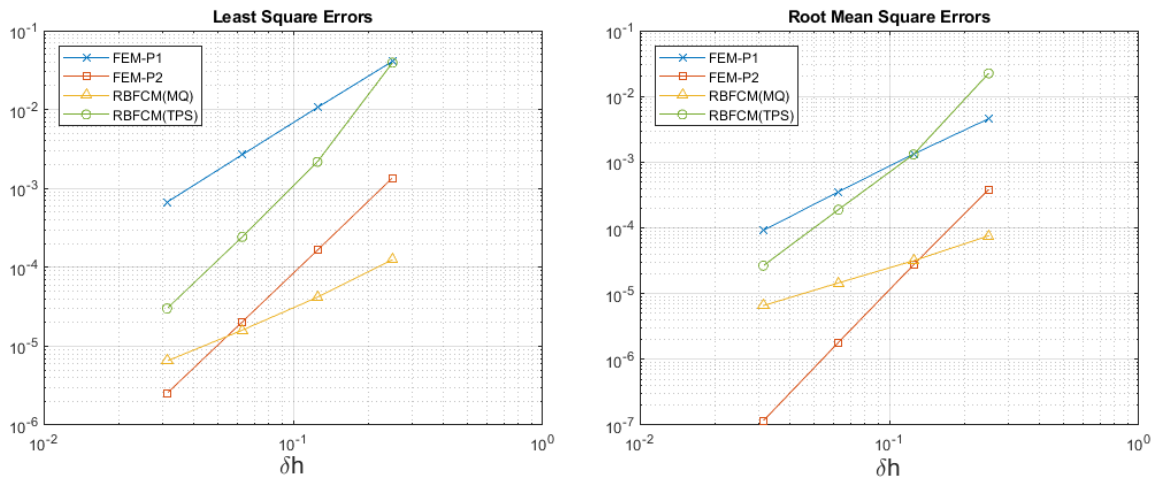

Figure 4.3. Least square error (left), and root mean square error (right) of the Example 4.1.1.

An interesting result of this example is that RBFCM-MQ achieved a clear h-convergence, which is not usual [50]. Such a convergence is very sensible to the shape factor. However, the analytical solution of the unknown $u(x, y)$ has only a harmonic solution. Thus, we think that the biharmonic nature of multiquadrics was able to capture the behaviour of $u(x, y)$ easily, independent from shape factor.

Table 4.1. Error results for the example 4.1.1.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	4.0913e-02	4.5785e-03	3.0996e-02
	FEM-P2	1.3607e-03	3.8258e-04	3.1752e-03
	RBFCM-MQ	1.2474e-04	7.5694e-05	3.2256e-04
	RBFCM-TPS	3.9215e-02	2.2392e-02	1.1867e-01
1/8	FEM-P1	1.0610e-02	1.3259e-03	1.3702e-02
	FEM-P2	1.6449e-04	2.7347e-05	7.3156e-04
	RBFCM-MQ	4.1730e-05	3.1698e-05	5.2630e-04
	RBFCM-TPS	2.1718e-03	1.3086e-03	1.6877e-02
1/16	FEM-P1	2.6783e-03	3.5420e-04	5.0984e-03
	FEM-P2	2.0341e-05	1.8088e-06	1.7848e-04
	RBFCM-MQ	1.5987e-05	1.4551e-05	6.0970e-04
	RBFCM-TPS	2.4298e-04	1.8919e-04	1.0450e-02
1/32	FEM-P1	6.7122e-04	9.1434e-05	1.7098e-03
	FEM-P2	2.5353e-06	1.1598e-07	4.4330e-05
	RBFCM-MQ	6.5102e-06	6.5645e-06	4.4268e-03
	RBFCM-TPS	3.0099e-05	2.6610e-05	9.9368e-03

Table 4.2. Comparisons for the example 4.1.1.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	1.0096e+01	0.37	
	FEM-P2	3.6373e+01	5.97	
	RBFCM-MQ	3.0615e+13	1.32	3.2690
	RBFCM-TPS	5.0316e+03	0.03	
1/8	FEM-P1	2.7294e+01	0.90	
	FEM-P2	1.3872e+02	24.04	
	RBFCM-MQ	1.5651e+14	13.84	1.0852
	RBFCM-TPS	6.7667e+05	0.42	
1/16	FEM-P1	1.0409e+02	3.45	
	FEM-P2	5.5299e+02	98.31	
	RBFCM-MQ	3.8185e+14	178.59	0.4521
	RBFCM-TPS	5.2737e+07	3.36	
1/32	FEM-P1	4.1485e+02	16.46	
	FEM-P2	2.2127e+03	458.54	
	RBFCM-MQ	2.9054e+15	2595.66	0.2184
	RBFCM-TPS	3.5464e+09	41.40	

4.1.2. Steady Poisson Equation with Dirichlet and Neumann Boundary Conditions

In this example, different from the previous one, both Dirichlet and Neumann boundary conditions are considered. For the domain $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$, the strong form of Poisson's Equation is:

$$\nabla \cdot (-k\nabla u) = f \text{ in } \Omega, \tag{4.7}$$

$$u = u_D \text{ on } \Gamma_D = \{(0, y) \cup (x, 0)\} \subseteq \partial\Omega, \tag{4.8}$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N = \{(x, 1) \cup (1, y)\} \subseteq \partial\Omega. \tag{4.9}$$

and the weak form is:

$$\int_\Omega k\nabla u \cdot \nabla v \; d\Omega = \int_\Omega fv \; d\Omega + \int_{\Gamma_N} kgv \; dS \tag{4.10}$$

Trigonometric *sinus* and *cosinus* functions with a polynomial function is set as an analytical solution this time:

$$u(x, y) = \cos(\pi x)\sin(\pi y) - xy^2, \tag{4.11}$$

The plot of analytical solution above is presented in Figure 4.5. Moreover, by setting $k = 1$, the variables f, u_0 and g become:

$$f = 2\pi^2 \cos(\pi x)\sin(\pi y) + 2x$$

$$u_D(0, y) = 0 \text{ and } u_D(x, 0) = \sin(\pi x)$$

$$g = \begin{bmatrix} -\pi \sin(\pi x)\sin(\pi y) - y^2 \\ \pi \cos(\pi x)\cos(\pi y) - 2xy \end{bmatrix} \cdot n$$

Now, for the RBFCM discretization we simply do,

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f \text{ in } \Omega \tag{4.12}$$

$$u|_{x=0,y} = 0 \text{ and } u|_{x,y=0} = \sin(\pi x) \text{ on } \Gamma_D \tag{4.13}$$

$$\frac{\partial u}{\partial y}|_{(x,y=1)} = -2x, \frac{\partial u}{\partial x}|_{(x=1,y)} = -\pi \cos(\pi y) - y^2 \text{ on } \Gamma_N \tag{4.14}$$

Computational domains are the same as the example 4.1.1. The only difference is the Neumann boundaries. Computational domains are presented in Figure 4.4

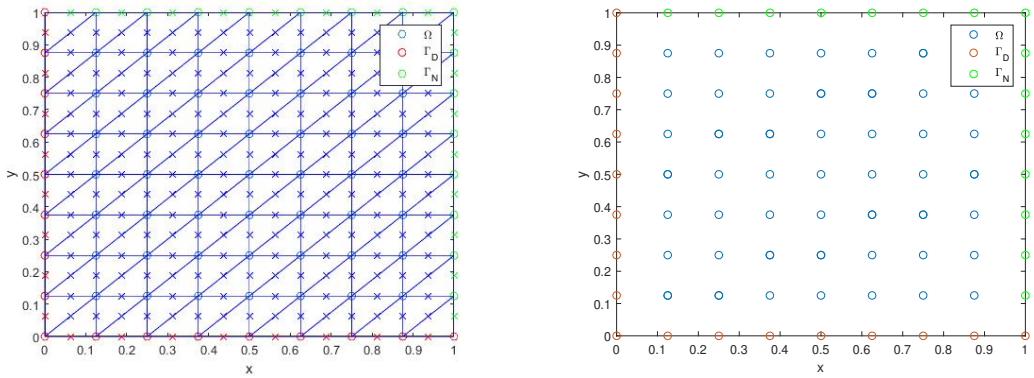

Figure 4.4. Computational domains for the example 4.1.2: second order FEM elements (left), RBFCM nodes (right), $\delta h = 0.125$.

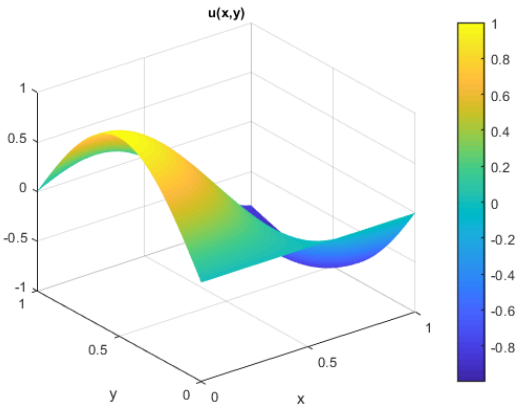

Figure 4.5. Plot of analytic solution of the example 4.1.2.

The major difference of this example from other examples is that we used symmetrical collocation here. It involves fourth order derivations of RBFs and solving these derivatives is very time consuming. Even though symmetrical collocated model has a tedious mathematical work beforehand, the improvement achieved by it happened to be insufficient to outperform FEM-P2 for the δh values we use. However, the h-convergence achieved by RBFCM-TPS(SC) is slightly better than that of FEM-P2. This is a promising outcome considering the small runtime of RBFCM-TPS(SC). We should note here, again, the fact that the FEM-P2 has more degrees of freedom in its solution. Therefore, RBFCM-TPS(SC) can be regarded as an efficient method considering, additionally, its relatively low condition number.

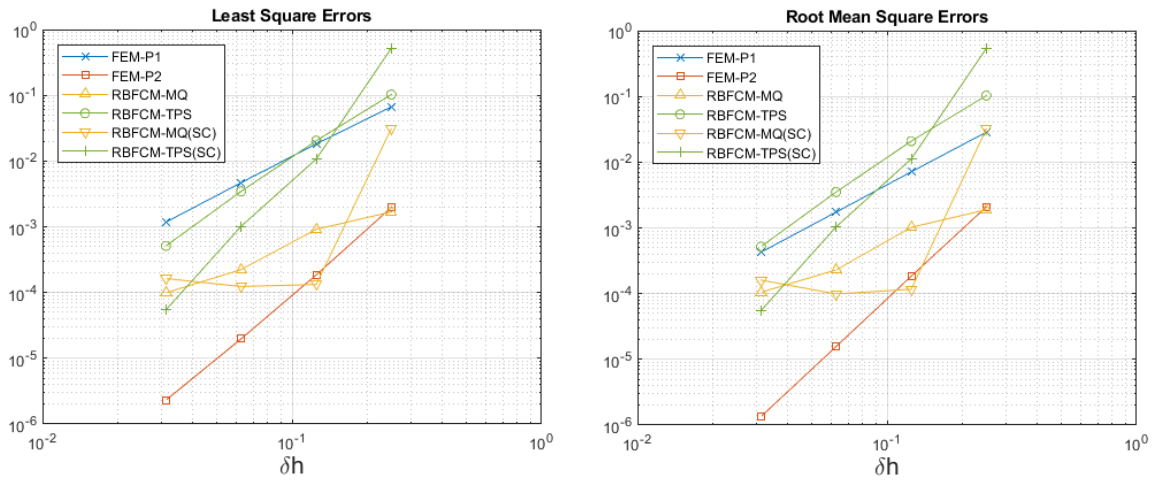

Figure 4.6. Least square error (left), and root mean square error (right) of the example 4.1.2.

Additionally, likewise the previous example, RBFCM-MQ achieved a satisfactory result for the biggest δh value. However, FEM-P2 results achieved far better results because of the good convergence that the method is able to succeed. Also it is important to mention that both FEM-P1 and FEM-P2 behave in a robust, predictable manner, which is very nice to be able to foreseen beforehand by the user.

Table 4.3. Error results for the Example 4.1.2.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	6.6548e-02	2.8595e-02	1.6574e-01
	FEM-P2	1.9713e-03	2.0920e-03	5.9226e-02
	RBFCM-MQ	1.6821e-03	1.8901e-03	1.4837e-02
	RBFCM-TPS	1.0260e-01	1.0286e-01	2.4686e-00
	RBFCM-MQ(SC)	3.1194e-02	3.2110e-02	5.7762e-01
	RBFCM-TPS(SC)	5.1996e-01	5.3758e-01	6.2197e-00
1/8	FEM-P1	1.8265e-02	7.1199e-03	2.8353e-01
	FEM-P2	1.8554e-04	1.8385e-04	1.9726e-02
	RBFCM-MQ	9.1635e-04	1.0196e-03	1.2390e-01
	RBFCM-TPS	2.0409e-02	2.0864e-02	8.0468e-01
	RBFCM-MQ(SC)	1.3263e-04	1.1660e-04	2.4124e-02
	RBFCM-TPS(SC)	1.0714e-02	1.1167e-02	2.1227e-01
1/16	FEM-P1	4.6832e-03	1.7358e-03	1.1307e+00
	FEM-P2	1.9774e-05	1.5660e-05	5.2236e-03
	RBFCM-MQ	2.2364e-04	2.2935e-04	1.8787e-01
	RBFCM-TPS	3.4551e-03	3.5116e-03	5.4010e-01
	RBFCM-MQ(SC)	1.2360e-04	9.8731e-05	6.1737e-02
	RBFCM-TPS(SC)	1.0174e-03	1.0374e-03	2.1303e-01
1/32	FEM-P1	1.1784e-03	4.2632e-04	1.2470e+01
	FEM-P2	2.3148e-06	1.3461e-06	1.3206e-03
	RBFCM-MQ	9.9145e-05	1.0480e-04	6.9503e-01
	RBFCM-TPS	5.1353e-04	5.1848e-04	4.4301e-00
	RBFCM-MQ(SC)	1.6306e-04	1.6036e-04	1.5940e-00
	RBFCM-TPS(SC)	5.4559e-05	5.5048e-05	8.1248e-01

Table 4.4. Comparisons for the Example 4.1.2.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	3.8528e+01	0.26	
	FEM-P2	1.7217e+02	5.82	
	RBFCM-MQ	1.0380e+13	1.39	3.0477
	RBFCM-TPS	1.7324e+04	0.04	
	RBFCM-MQ(SC)	1.0594e+10	2.66	2.4921
	RBFCM-TPS(SC)	8.6540e+04	0.07	
1/8	FEM-P1	1.2952e+02	1.09	
	FEM-P2	6.2128e+02	23.07	
	RBFCM-MQ	4.1429e+18	13.63	1.7752
	RBFCM-TPS	2.0877e+07	0.25	
	RBFCM-MQ(SC)	2.7147e+18	68.82	3.2085
	RBFCM-TPS(SC)	3.8574e+08	1.26	
1/16	FEM-P1	4.6665e+02	3.79	
	FEM-P2	2.3502e+03	92.00	
	RBFCM-MQ	2.5521e+20	176.01	2.3188
	RBFCM-TPS	7.5561e+09	3.32	
	RBFCM-MQ(SC)	6.0351e+21	1151.32	4.7549
	RBFCM-TPS(SC)	2.1578e+13	31.31	
1/32	FEM-P1	1.7635e+03	16.21	
	FEM-P2	9.1286e+03	436.11	
	RBFCM-MQ	5.3153e+20	2405.89	3.1656
	RBFCM-TPS	2.1175e+12	43.55	
	RBFCM-MQ(SC)	5.2492e+20	24982.49	4.5599
	RBFCM-TPS(SC)	1.5476e+17	483.65	

4.1.3. Steady Poisson Equation with Dirichlet and Neumann Boundary Conditions 2

In this last example of Steady Poisson Equation, the domain is chosen to be L-shaped rather than unit square. The boundary conditions applied at the bottom and left walls are of Dirichlet type, and for the rest, Neumann Boundary condition is applied. Additionally, in order to show true meshless nature of RBFCM, a second domain, in which the nodes are randomly generated, is created along with meshing uniformly throughout the domain. The number of the nodes used in this non-uniform domains is the same with its uniform counterpart. The results obtained by using these non-uniform domain are marked with asterisk sign *(e.g. RBFCM-MQ*)*. The domain plots are shown in Figure 4.7.

The problem definition for this example is:

$$\nabla \cdot (-k\nabla u) = f \text{ in } \Omega, \tag{4.15}$$

$$u = u_D \text{ on } \Gamma_D = \{(0, y) \cup (x, 0)\} \subseteq \partial\Omega, \tag{4.16}$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N = \partial\Omega \setminus \Gamma_D. \tag{4.17}$$

and the weak form is exactly same with the Equation 4.10.

The analytical equation is an exponential function with trigonometric function, which reads as:

$$u(x, y) = ye^{-x^2} + \sin(\pi x)\cos(\pi y), \tag{4.18}$$

Thus, by setting $k = 1$, the variables f, u_D and g become:

$$f = ye^{-x^2}(2 - 4x^2) + 2\pi^2 \sin(\pi x)\cos(\pi y)$$

$$u_D(0, y) = y \text{ and } u_D(x, 0) = \sin(\pi x)$$

$$g = \begin{bmatrix} -2xye^{-x^2} + \pi\cos(\pi x)\cos(\pi y) \\ e^{-x^2} - \pi\sin(\pi x)\sin(\pi y) \end{bmatrix} \cdot n$$

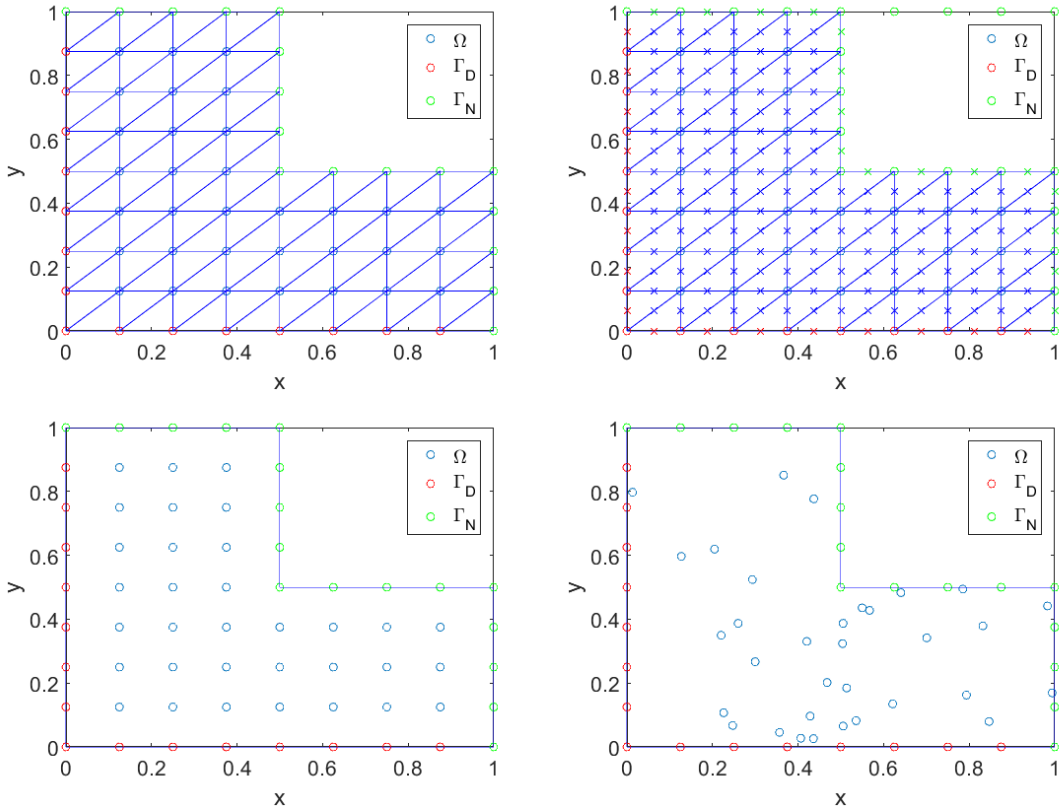

Figure 4.7. Computational domains for the Example 4.1.3: first order FEM elements (upper left), second order FEM elements (upper right), uniform RBFCM nodes (lower left), randomly generated RBFCM nodes (lower right) $\delta h = 0.125$.

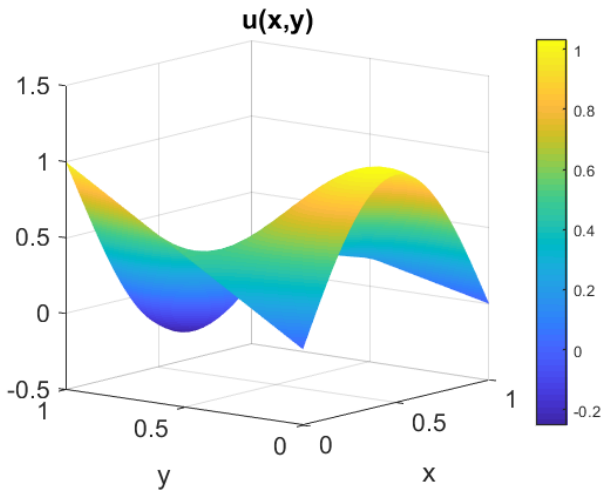

Figure 4.8. Plot of analytic solution of the Example 4.1.3.

Considering the LSE results, the convergence achieved by both RBFCM-TPS and RBFCM-TPS* is remarkable[14] . This good convergence of RBFCM-TPS* allowed it to outrun FEM-P2 and RBFCM-TPS to catch up with FEM-P2. Also, it is obvious that randomizing the locations of the collocation points made a good improvement on RBFCM-TPS. We were unable to give a satisfactory reason to this happening.

The LSE results of RBFCM-MQ, on the other hand, performed worse considering the performance it delivered in the previous examples. This may be due to the extremely large condition number of the system matrices generated by the RBFCM-MQ. Hence, it can be said that the inversion errors worsened the approximation.

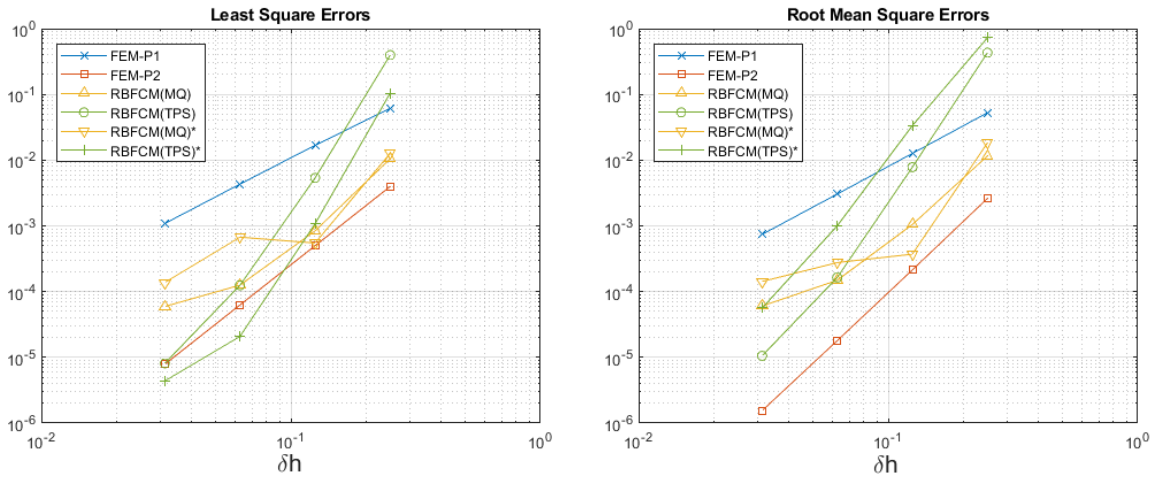

Figure 4.9. Least square errors (left), and root mean square errors (right) of the Example 4.1.3.

[14]It is worth to mention here that these are not symmetrically collocated like the previous example.

Table 4.5. Error results for the Example 4.1.3.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	6.1314e-02	5.2674e-02	4.9549e-01
	FEM-P2	3.9733e-03	2.6220e-03	4.5113e-02
	RBFCM-MQ	1.0482e-02	1.1598e-02	2.5729e-01
	RBFCM-TPS	3.9608e-01	4.3068e-01	6.5524e-00
	RBFCM-MQ*	1.2702e-02	1.8341e-02	3.1080e-01
	RBFCM-TPS*	1.0455e-01	7.2773e-01	1.0916e+01
1/8	FEM-P1	1.6813e-02	1.2653e-02	6.5368e-00
	FEM-P2	4.9848e-04	2.1394e-04	4.4954e-02
	RBFCM-MQ	8.4011e-04	1.0493e-03	7.8007e-03
	RBFCM-TPS	5.3602e-03	7.8091e-03	1.7169e-00
	RBFCM-MQ*	5.4772e-04	3.7115e-04	6.3870e-03
	RBFCM-TPS*	1.0738e-03	3.3464e-02	6.8322e-01
1/16	FEM-P1	4.3166e-03	3.0485e-03	1.6217e+00
	FEM-P2	6.2625e-05	1.7972e-05	1.4341e-02
	RBFCM-MQ	1.2543e-04	1.4906e-04	2.2941e-02
	RBFCM-TPS	1.2470e-04	1.6389e-04	7.9943e-02
	RBFCM-MQ*	6.7277e-04	2.7652e-04	1.2967e-02
	RBFCM-TPS*	2.0777e-05	1.0013e-03	2.0719e-01
1/32	FEM-P1	1.0868e-03	7.4206e-04	9.7824e-01
	FEM-P2	7.8558e-06	1.5462e-06	5.9083e-04
	RBFCM-MQ	5.9462e-05	6.0289e-05	5.9806e-02
	RBFCM-TPS	8.1155e-06	1.0539e-05	4.8547e-03
	RBFCM-MQ*	1.3766e-04	1.4223e-04	9.7173e-02
	RBFCM-TPS*	4.3778e-06	5.6193e-05	6.8266e-03

Table 4.6. Comparisons for the Example 4.1.3.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	8.5473e+16	0.17	
	FEM-P2	2.2710e+16	3.20	
	RBFCM-MQ	9.7932e+38	1.29	2.0453
	RBFCM-TPS	1.3816e+04	0.01	
	RBFCM-MQ*	1.0259e+15	1.20	3.9142
	RBFCM-TPS*	1.9752e+04	0.01	
1/8	FEM-P1	4.6499e+16	0.82	
	FEM-P2	3.1471e+16	13.99	
	RBFCM-MQ	2.8510e+76	12.09	4.9847
	RBFCM-TPS	2.8867e+08	0.16	
	RBFCM-MQ*	1.2396e+12	12.88	0.7111
	RBFCM-TPS*	1.1667e+08	0.15	
1/16	FEM-P1	8.9694e+16	2.54	
	FEM-P2	2.3842e+16	55.33	
	RBFCM-MQ	5.6765e+131	176.01	3.3479
	RBFCM-TPS	5.8608e+11	1.68	
	RBFCM-MQ*	6.2932e+20	158.07	2.8187
	RBFCM-TPS*	1.5451e+12	0.04	
1/32	FEM-P1	7.5725e+16	12.09	
	FEM-P2	1.0653e+17	272.45	
	RBFCM-MQ	5.8934e+164	2130.74	2.1268
	RBFCM-TPS	7.0240e+14	25.31	
	RBFCM-MQ*	8.75314e+16	2115.69	0.1815
	RBFCM-TPS*	8.2372e+15	23.66	

4.1.4. Unsteady Poisson Equation with Dirichlet Boundary Conditions

The domain in this example is chosen to be the same as the Example 4.1.1, since these examples serve as test cases for more realistic examples. Also, all the boundaries are of Dirichlet type. A simple Backward Euler Method is used as a time integrator. Time step size, δt, is chosen to be 0.01. All the results and figures related to this example are of final time step, $T_f = 50$.

The strong form of the equation, the boundary conditions and the initial conditions are:

$$\frac{\partial u}{\partial t} + \nabla \cdot (-k \nabla u) = f \text{ in } \Omega \times (0, T], \tag{4.19}$$

$$u(x_{\Gamma_D}, y_{\Gamma_D}, t) = u_D(t) \text{ on } \Gamma_D \times (0, T], \tag{4.20}$$

$$u(x, y, 0) = u_i(x, y) \text{ at } t = 0, \tag{4.21}$$

and the weak form is obtained by multiplying Equation 4.19 with test function v and applying integration by parts is written as follows:

$$\int_T \int_\Omega \frac{\partial u}{\partial t} \cdot v \; d\Omega dt + \int_T \int_\Omega k \nabla u \cdot \nabla v \; d\Omega dt = \int_T \int_\Omega f v \; d\Omega dt \tag{4.22}$$

Here we defined a simple analytical solution with a first order time variable, to be able not to see the time errors, since otherwise, with a first order time integration scheme as BE, the time errors would dominate the overall error. Therefore, the analytic solution then is defined as follows:

$$u(x, y, t) = \exp\left(\frac{-x}{y+1}\right) + 0.8t \tag{4.23}$$

Therefore, by setting $k = 1$, the source term f becomes:

$$f = 0.8 - \frac{\exp\left(\frac{-x}{y+1}\right)}{(y+1)^2} - \frac{x \exp\left(\frac{-x}{y+1}\right)(x - 2y - 2)}{(y+1)^4}$$

For the RBFCM discretization, the partial form is defined as:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f \text{ in } \Omega \tag{4.24}$$

Here, the reader may refer to Figure 4.1.1 to see the computational domains. To have a better grasp of this example, a plot of the solution for a *given time* is provided in Figure 4.10. The results obtained with FEM and RBFCM are summarized in Table 4.7.

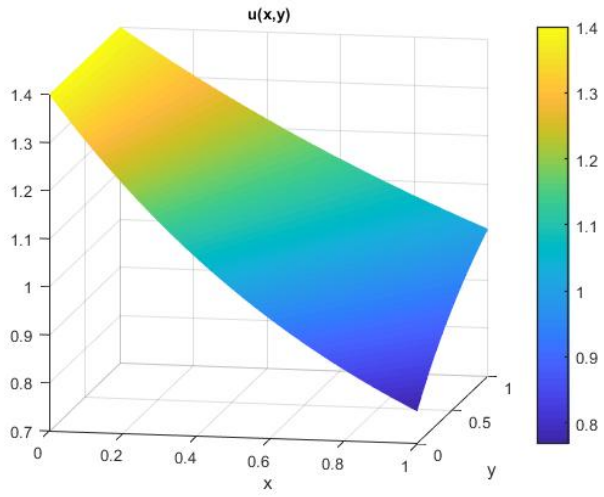

Figure 4.10. Plot of analytic solution of the example 4.1.4 at time step 50.

Considering both LSE and RMSE results of this example, it can be claimed that the FEM-P2 errors are far better than any other method especially for small δh values. However, likewise the previous examples, the convergence rate of RBFCM-TPS is noteworthy that it might best even FEM-P2 errors for lower δh values. Moreover, the runtime of RBFCM-TPS is far smaller than that of FEM-P2, but the condition number gets significantly higher as the mesh gets finer, meaning that the solution is

less and less robust. Unfortunately, refining mesh further, or using more nodes inside the node was not doable because of both memory(RAM) and runtime(GPU) reasons.

Additionally, this example is an unsteady one. Therefore, it is possible that the RBFCM approximations tend to have higher error when the solution includes the time discretization.

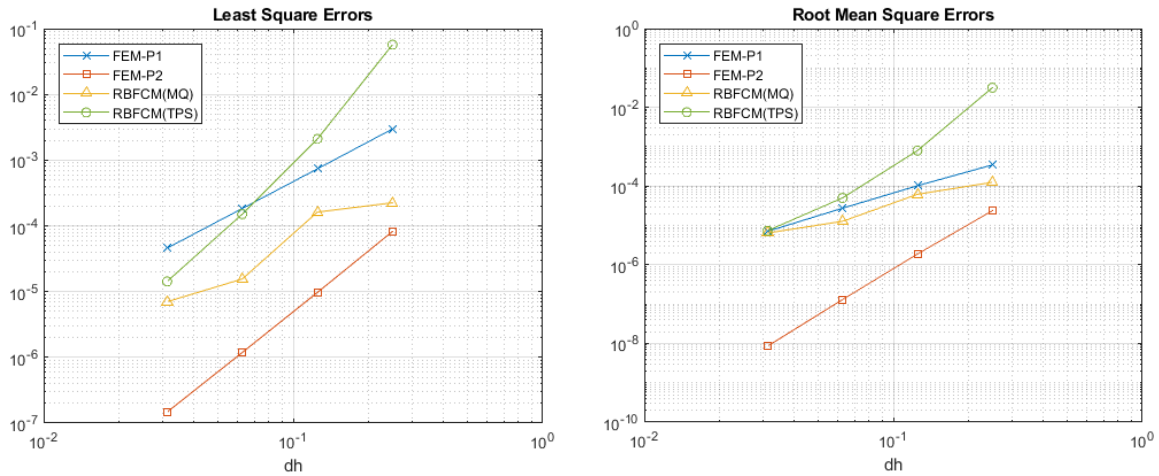

Figure 4.11. Least square error (left), and root mean square error (right) of the example 4.1.4 at the final time step.

Table 4.7. Error results for the example 4.1.4.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	2.9807e-03	3.4385e-04	6.7155e-04
	FEM-P2	8.2270e-05	2.3670e-05	6.7724e-05
	RBFCM-MQ	2.2434e-04	1.2448e-04	3.4047e-04
	RBFCM-TPS	5.7000e-02	3.1522e-02	5.8299e-02
1/8	FEM-P1	7.3728e-04	1.0176e-04	1.8990e-04
	FEM-P2	9.7053e-06	1.8735e-06	7.0307e-06
	RBFCM-MQ	1.6202e-04	6.1755e-05	1.9620e-04
	RBFCM-TPS	2.0935e-03	7.9506e-04	1.5508e-03
1/16	FEM-P1	1.8366e-04	2.7314e-05	4.8728e-05
	FEM-P2	1.1855e-06	1.3038e-07	6.0283e-07
	RBFCM-MQ	1.5548e-05	1.2669e-05	4.0090e-05
	RBFCM-TPS	1.5074e-04	4.9898e-05	2.1140e-04
1/32	FEM-P1	4.5868e-05	7.0589e-06	1.2223e-05
	FEM-P2	1.4716e-07	8.5568e-09	4.4584e-08
	RBFCM-MQ	7.0277e-06	6.4226e-06	2.5192e-05
	RBFCM-TPS	1.4284e-05	7.3250e-06	6.9690e-05

Table 4.8. Comparisons for the Example 4.1.4.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	1.4723e+00	0.31	
	FEM-P2	6.4474e+00	7.46	
	RBFCM-MQ	9.2692e+13	3.86	3.1083
	RBFCM-TPS	5.6832e+02	0.03	
1/8	FEM-P1	4.5962e+00	1.37	
	FEM-P2	2.3494e+01	31.94	
	RBFCM-MQ	4.5153e+19	19.10	3.7903
	RBFCM-TPS	2.7334e+04	0.25	
1/16	FEM-P1	1.7413e+01	6.81	
	FEM-P2	9.1896e+01	155.66	
	RBFCM-MQ	1.0099e+12	181.25	0.3624
	RBFCM-TPS	1.7672e+06	2.94	
1/32	FEM-P1	6.8722e+01	56.49	
	FEM-P2	3.6555e+02	1164.50	
	RBFCM	1.5840e+12	2322.39	0.1676
	RBFCM-TPS	1.1475e+08	41.96	

4.1.5. Unsteady Poisson Equation with Dirichlet Boundary Conditions 2

This example is solved to compare the FEM code we develop from scratch with a commercial finite element software. The domain setup and time integration parameters are all the same as in Example 4.1.4. Therefore, reader may refer to strong, weak and partial forms described in Equations 4.19, 4.22 and 4.24 respectively. However, only

difference here is the analytical solution, which is:

$$u(x, y, t) = 1 + x^2 + 3y^2 + 1.2t \tag{4.25}$$

and hence the source term, f, is calculated as:

$$f = -6.8$$

Here, spatial variables are of second order and the time variable is of first order. Therefore, both FEM-P2 in house code and the commercial software's results computed with second order elements are expected to reach machine precision. For this reason, we only have shown the second order convergence obtained using linear finite elements in Figure 4.14 and Table 4.10.

As mentioned above, this example serves as validation of our finite element code with the help of a commercial software. Hence, the percent difference between our FEM-P2 solution and the solution of commercial software[15] for each node is presented in Figure 4.13. It can be seen from the same figure that the maximum percent difference is of order 10^{-12}, meaning that the solutions are almost identical.

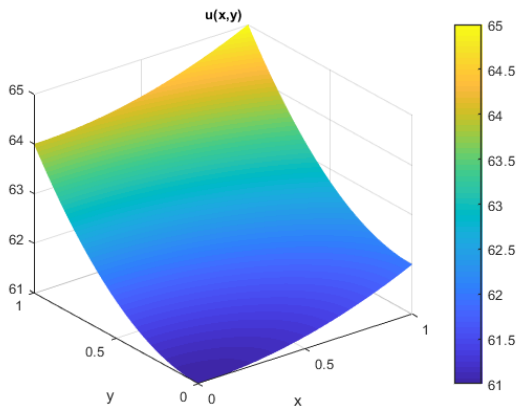

Figure 4.12. Plot of analytic solution of the Example 4.1.5 at time step 50.

[15]The commercial software used here is *FEniCS*.

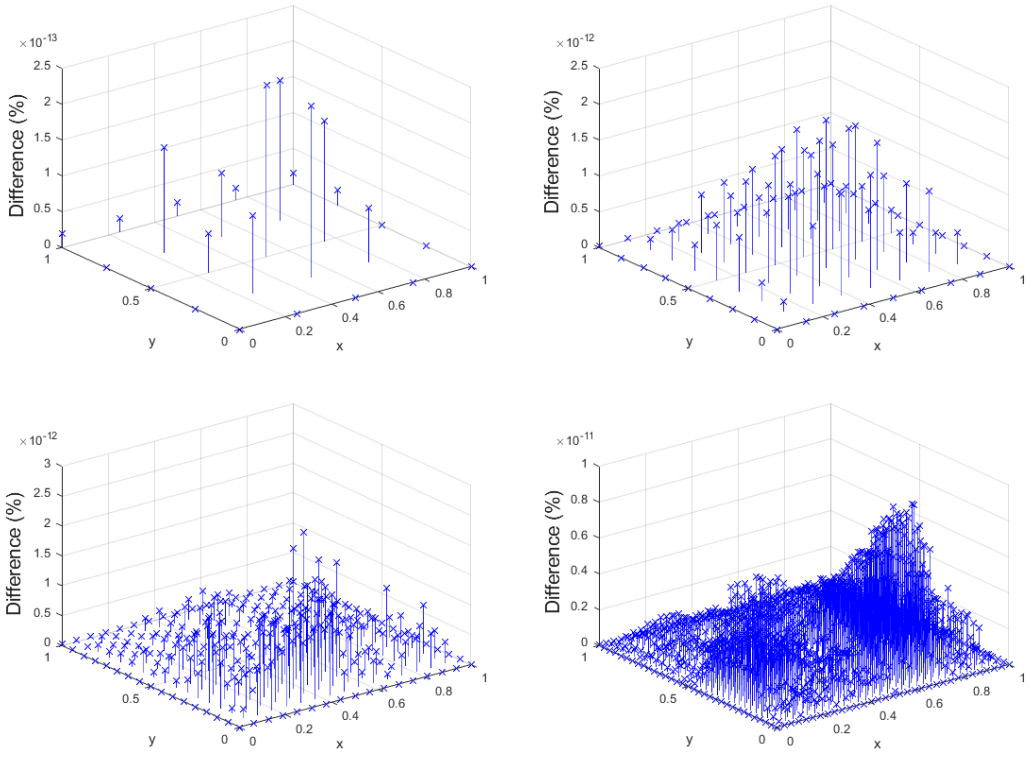

Figure 4.13. Percent difference between FEM-P1 solutions of ours and commercial software's for each δh.

Considering LSE results of this example, RBFCM-TPS achieves an h-convergence likewise the previous examples. However, its approximation for bigger δh values is not better than FEM-P1. Moreover, even though RBFCM-MQ does not achieve a h-convergence, its approximation is the best excluding FEM-P2. An immediate result of RBFCM-TPS having a h-convergence unlike RBFCM-MQ is that RBFCM-TPS needs more nodes in order to approximate better, whereas RBFCM-MQ does not necessarily need more nodes inside the domain. This status between two RBFs is also present in the previous examples and can be seen in Figure 4.14 for this example.

On the other hand, FEM-P1 achieved a machine precision for RMSE. We have observed, in general, that if the analytical solution is one order higher than the order of shape function used in finite element approximation, the RMSE of approximated solution achieves a machine precision.

Table 4.9. Error results for the example 4.1.5.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	4.4194e-02	4.7749e-15	6.7155e-04
	RBFCM-MQ	2.4440e-05	1.4020e-05	3.4047e-04
	RBFCM-TPS	1.3689e-01	7.4916e-02	6.8904e-02
1/8	FEM-P1	1.1048e-02	1.4741e-14	1.8990e-04
	RBFCM-TPS	1.4284e-05	7.3250e-06	6.9690e-05
	RBFCM-MQ	5.1141e-03	1.9228e-03	1.5841e-03
1/16	FEM-P1	2.7621e-03	4.0227e-14	4.8728e-05
	RBFCM-MQ	7.7948e-05	6.5415e-05	4.0090e-05
	RBFCM-TPS	3.8338e-04	1.5437e-04	3.5372e-04
1/32	FEM-P1	6.9053e-04	4.3011e-14	1.2223e-05
	RBFCM-MQ	4.2960e-05	3.9559e-05	2.5192e-05
	RBFCM-TPS	1.3164e-05	2.2349e-05	9.9747e-05

Table 4.10. Comparisons for the example 4.1.5.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	1.4723e+00	0.31	
	RBFCM-MQ	9.2692e+13	3.86	3.1083
	RBFCM-TPS	5.6832e+02	0.03	
1/8	FEM-P1	4.5962e+00	1.37	
	RBFCM-MQ	4.5153e+19	19.10	3.7903
	RBFCM-TPS	2.7334e+04	0.25	
1/16	FEM-P1	1.7413e+01	6.81	
	RBFCM-MQ	1.0099e+12	181.25	0.3624
	RBFCM-TPS	1.7672e+06	2.79	
1/32	FEM-P1	6.8722e+01	56.49	
	RBFCM-MQ	1.5840e+12	2322.39	0.1676
	RBFCM-TPS	1.1475e+08	39.84	

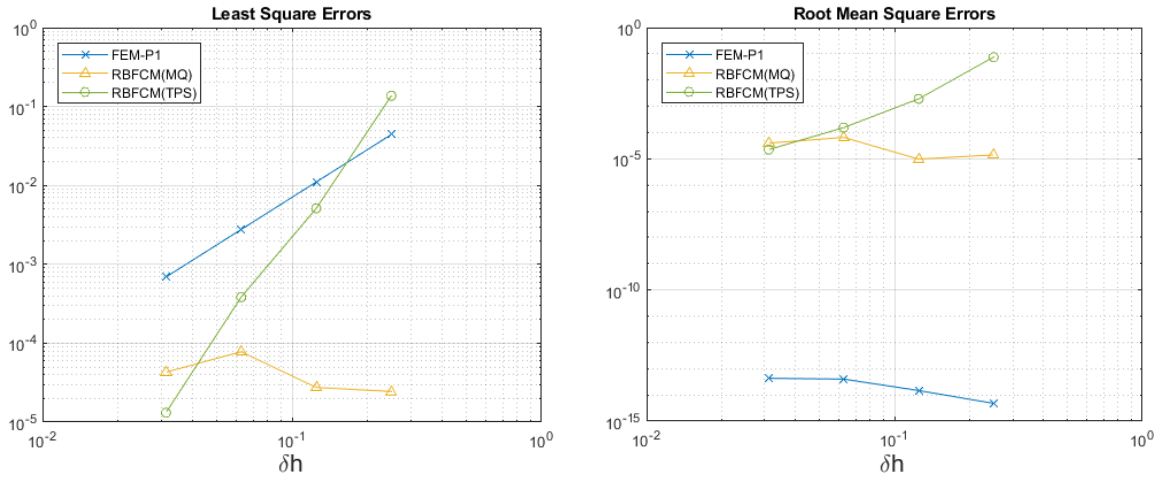

Figure 4.14. Least square error (left), and root mean square error (right) of the Example 4.1.5 at the final time step.

4.1.6. Unsteady Poisson Equation with Dirichlet and Neumann Boundary Conditions

In this example, time dependent Poisson Equation is solved. The domain created here is L-shaped and it is the same as the domain defined in Example 4.1.3. The domain can be seen in Figure 4.7. Likewise in Example 4.1.3, a domain with randomly generated nodes is also created to show the true meshless nature of RBFCM in this unsteady case. The results obtained using these non-uniform domains are marked with a asterisk sign *(e.g. RBFCM-MQ*)*.

Strong form of the equation and the problem definition are:

$$\frac{\partial u}{\partial t} + \nabla \cdot (-k \nabla u) = f \text{ in } \Omega \times (0, T], \tag{4.26}$$

$$u(x_{\Gamma_D}, y_{\Gamma_D}, t) = u_D(t) \text{ on } \Gamma_D \times (0, T], \tag{4.27}$$

$$\nabla u \cdot n = g \text{ on } \Gamma_N \times (0, T], \tag{4.28}$$

$$u(x, y, 0) = u_i(x, y) \text{ at } t = 0, \tag{4.29}$$

and the weak form is as follows:

$$\int_T \int_\Omega \frac{\partial u}{\partial t} \cdot v \; d\Omega dt + \int_T \int_\Omega k \nabla u \cdot \nabla v \; d\Omega dt = \int_T \int_\Omega f v \; d\Omega dt + \int_T \int_{\Gamma_N} k g v \; dS \quad (4.30)$$

Here, the analytical solution is chosen to be:

$$u(x, y, t) = y \cos^2(2\pi x) + xy \sin(\pi y/2) + 0.5t, \quad (4.31)$$

and thus, setting $k = 1$ in Equation 4.15, the source term, initial and boundary conditions become:

$$f = 8\pi^2 y(\cos^2(2\pi x) - \sin^2(2\pi x)) + x\pi(\frac{\pi}{4} y \sin(\pi y/2) - \cos(\pi y/2)) + 0.5$$

$$u_D(0, y, t) = y + 0.5t \text{ and } u_D(x, 0, t) = 0.5t$$

$$g = \begin{bmatrix} y(\sin(\pi y/2) - 4\pi \cos(2\pi x) \sin(2\pi x)) \\ \cos^2(2\pi x) + x(\sin(\pi y/2) + \frac{\pi}{2} y \cos(\pi y/2)) \end{bmatrix} \cdot n$$

$$u_i(x, y, 0) = y \cos^2(2\pi x) + xy \sin(\pi y/2)$$

The analytical solution of this example at the initial time instant is given in Figure 4.15. Note that the surface depicted in the figure will shift upwards (in z-direction).

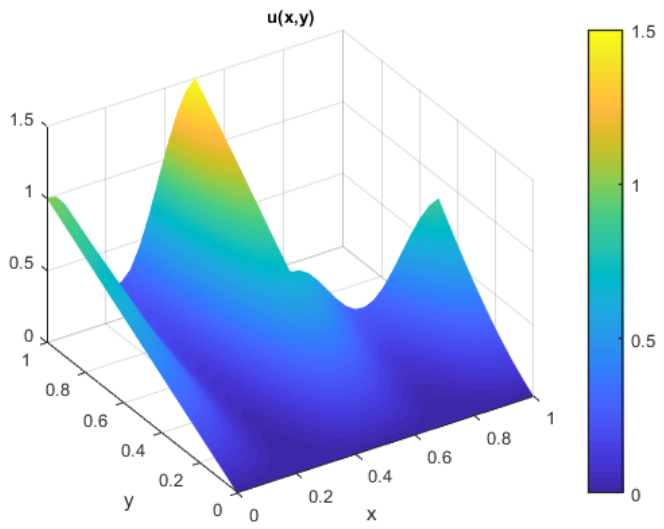

Figure 4.15. Plot of analytic solution of the Example 4.1.6 at $t = 0$.

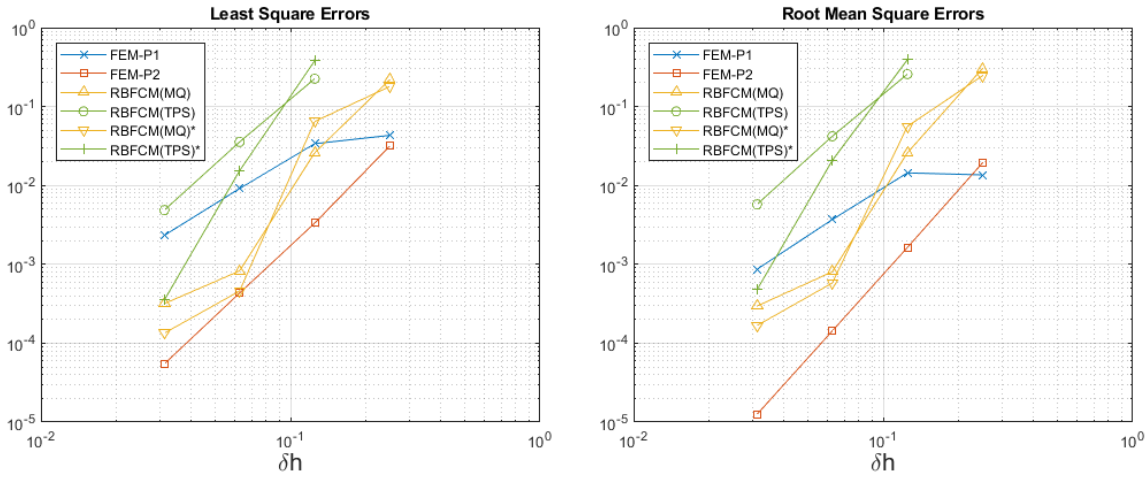

Figure 4.16. Least square errors (left), and root mean square errors (right) of the Example 4.1.6.

For $\delta h = 1/4$, RBFCM-TPS was not able to converge to the real solution. However for smaller δh values, it converges. One interesting outcome of this example is the better convergence rate of RBFCM-TPS* than that of RBFCM-TPS despite that the only difference is distributing nodes randomly. The best approximation is done by FEM-P2. Considering the previous examples, RBFCM-MQ was able to approximate the real solution better for large δh values. As can be seen from Table 4.12, the condition number of RBFCM-MQ is very high compared to other methods. The inversion errors due to the high condition number may takeover the RMSE and LSE errors of RBFCM-MQ.

Table 4.11. Error results for the example 4.1.6.

δh		**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM-P1	4.3136e-02	1.3596e-02	6.8642e-02
	FEM-P2	3.1645e-02	1.9303e-02	4.8326e-02
	RBFCM-MQ	2.1948e-01	3.0416e-01	5.3092e-01
	RBFCM-TPS	-	-	-
	RBFCM-MQ*	1.7967e-01	2.3906e-01	6.5204e-01
	RBFCM-TPS*	-	-	-
1/8	FEM-P1	3.3809e-02	1.4415e-02	4.7364e-02
	FEM-P2	3.3259e-03	1.6333e-03	8.7680e-03
	RBFCM-MQ	2.5986e-02	2.5900e-02	1.2377e-01
	RBFCM-TPS	2.2504e-01	2.5718e-01	9.5100e-01
	RBFCM-MQ*	6.4619e-02	4.9836e-03	3.3697e-02
	RBFCM-TPS*	3.7964e-01	5.7491e-01	1.5558e-00
1/16	FEM-P1	9.1263e-03	3.6902e-03	1.8492e-02
	FEM-P2	4.3212e-04	1.4296e-04	9.7149e-04
	RBFCM-MQ	8.2429e-04	8.0403e-04	4.2117e-03
	RBFCM-TPS	3.5533e-02	4.1969e-02	1.6628e-01
	RBFCM-MQ*	4.5798e-04	8.6724e-04	5.3061e-03
	RBFCM-TPS*	1.5318e-02	2.0688e-02	8.7538e-02
1/32	FEM-P1	2.3344e-03	1.2768e-02	5.9405e-03
	FEM-P2	5.4762e-05	1.2585e-05	1.3858e-04
	RBFCM-MQ	3.1808e-04	2.9621e-04	3.3921e-03
	RBFCM-TPS	4.8726e-03	5.7666e-03	2.4396e-02
	RBFCM-MQ*	1.3438e-04	2.4708e-04	1.3438e-03
	RBFCM-TPS*	3.5415e-04	2.5616e-03	1.3324e-02

Table 4.12. Comparisons for the example 4.1.6.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM-P1	4.2580e+00	0.20	
	FEM-P2	1.4282e+01	4.76	
	RBFCM-MQ	9.2605e+48	4.17	111.1500
	RBFCM-TPS	-	-	
	RBFCM-MQ*	3.5199e+20	5.75	109.7731
	RBFCM-TPS*	-	-	
1/8	FEM-P1	9.0496e+00	0.94	
	FEM-P2	3.3544e+01	18.19	
	RBFCM-MQ	1.2052e+73	21.29	1.6013
	RBFCM-TPS	5.9543e+06	0.22	
	RBFCM-MQ*	1.6964e+18	21.81	1.3919
	RBFCM-TPS*	2.7514e+07	0.23	
1/16	FEM-P1	2.4168e+01	4.48	
	FEM-P2	1.1005e+02	89.09	
	RBFCM-MQ	9.6607e+130	181.21	0.9145
	RBFCM-TPS	6.8422e+08	2.15	
	RBFCM-MQ*	5.3205e+13	234.06	0.3187
	RBFCM-TPS*	1.2206e+10	2.15	
1/32	FEM-P1	8.2067e+01	33.82	
	FEM-P2	4.1185e+02	640.09	
	RBFCM-MQ	1.3203e+163	2290.32	1.1611
	RBFCM-TPS	1.1167e+11	30.89	
	RBFCM-MQ*	1.1525e+20	2428.36	1.0345
	RBFCM-TPS*	2.1596e+12	27.67	

4.1.7. Diffusion of a Gaussian Hill

In this example, the so called "Gaussian Hill" problem is solved. The domain is a simple unit square and in the center of the domain, the problem starts with a "Hill". All the boundaries are of Dirichlet type and because of the Dirichlet boundaries held constant with the value of 0, the hill diffuses near 0 everywhere as time goes on. The Gaussian Hill is modeled using time dependent Poisson Equation. Therefore, the reader may refer from Equations 4.19 to 4.22 and Equation 4.24 for the strong, weak and the partial forms.

The necessary parameters; the source term, initial condition and boundary conditions are:

$$f = 0$$

$$u_D = 0$$

$$u_i = exp(-5(x-0.5)^2 - 5(y-0.5)^2)$$

This initial condition forces the hill to start at the middle for the unit square type domain.

Contour plots of the numerical solutions obtained with in-house codes along with the commercial finite element software at the final time instant is shown in Figure 4.17 for reader to visually confirm the reliability of the in-house codes.

Additionally, it is worth to mention that there is no analytical solution of this problem. Therefore, it is not possible to draw error comparison tables related to this example.

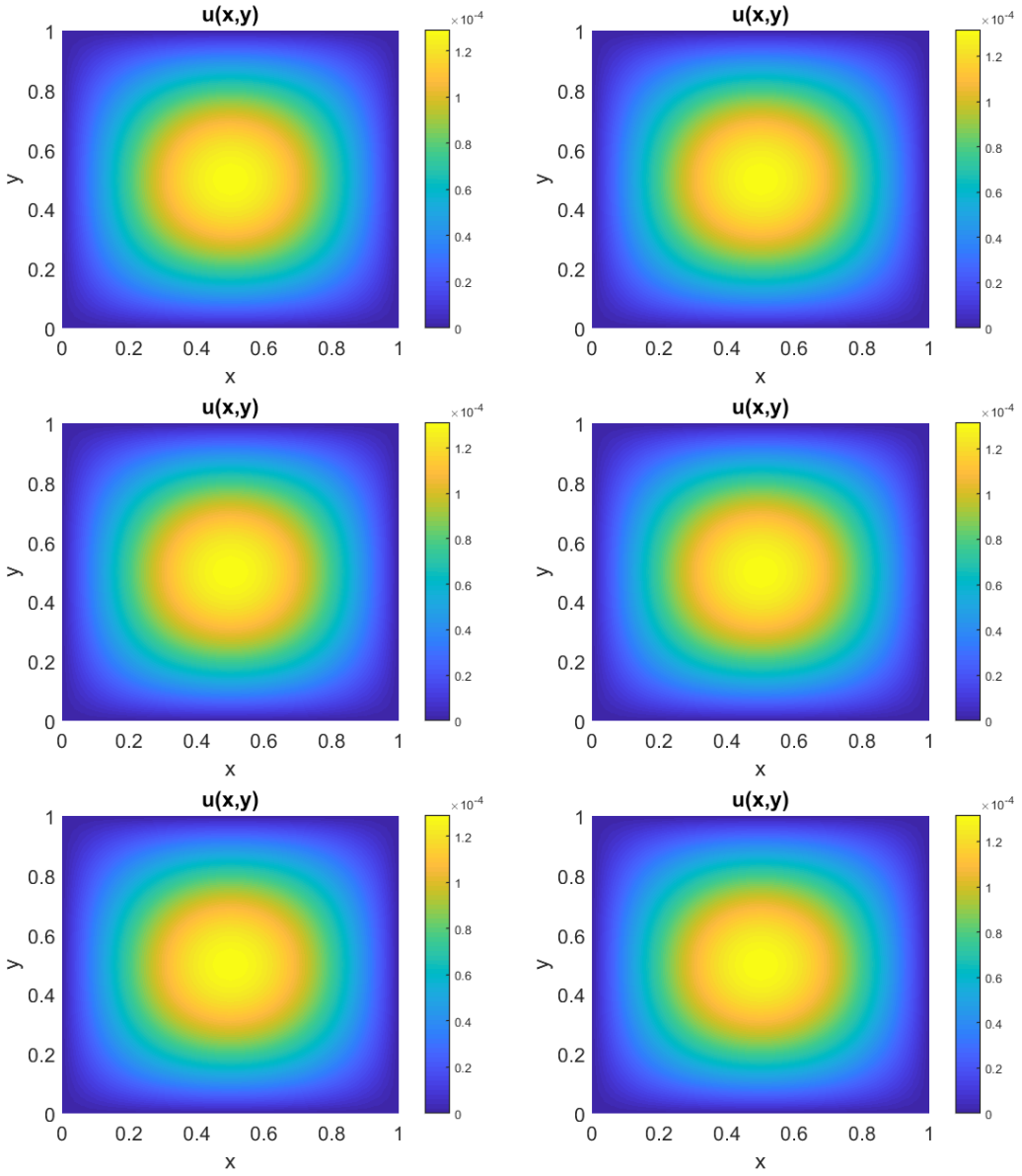

Figure 4.17. Contour plot of the solution of Example 4.1.7 at final time instant; FEM-P1 (upper left), FEM-P2 (upper right), RBFCM-MQ (middle left), RBFCM-TPS (middle right), CFES-P1 (lower left) and CFES-P2 (lower right).

4.2. Stokes Equation

The Stokes Equation are the equations which governs the fully viscous flow in which the convective behaviuors are neglected. In order to neglect the convective terms, the velocity of the flow must be very small such as in the real physical examples of

groundwater flow and creeping flow. These type of flows are also called Darcy flow. Neglecting convective terms leads to the inertial forces being zero in the flow. Immediate result of this is Reynold's number, Re, also being zero, since $Re = \frac{InertialForces}{ViscousForces}$. Also, it is much easier to model Stokes Equations than Navier-Stokes Equations since the non-linearity arising from convective terms are not present [41].

Strong form of the Stokes Equation with incompressibility condition is as follows:

$$-\nabla^2 \mathbf{u} + \nabla p = \mathbf{0} \tag{4.32}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{4.33}$$

where $\mathbf{u}$ is the velocity vector and p is the pressure. And the weak form is defined as[16] :

$$\int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega = \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} \; dS \tag{4.34}$$

$$\int_\Omega (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{4.35}$$

where $\mathbf{v}$ is the vector of the test function for velocity and q is the test function for pressure unknown. Since we defined the weak form for FE solution, next we define the partial form of Stokes equation for meshless RBFCM model is:

$$-\frac{\partial^2 u_x}{\partial x^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial p}{\partial x} = 0 \tag{4.36}$$

$$-\frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial p}{\partial y} = 0 \tag{4.37}$$

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \tag{4.38}$$

[16] See Appendix A.3 for full weak formulation of the Stokes equations.

4.2.1. Poiseuille Flow

Poiseuille Flow is an easy flow to model using Stokes equation since the analytic velocity and pressure can be calculated and hence error analysis can be done. In this flow, Dirichlet boundary conditions are defined on the inlet, Neumann boundary conditions are defined on the outflow and no slip boundary conditions are defined on the rest of the boundaries. The domain $\Omega = (-1,1) \times (-1,1)$ and the boundaries $\Gamma_N = \{(1,y) \subset \partial\Omega\}$, $\Gamma_D = \partial\Omega \backslash \Gamma_N$. The flow is moving from inlet to outlet very slowly. Analytical solution for velocity and pressure is set to be:

$$u_x = 1 - y^2,\ u_y = 0,\ p = -2x + constant \tag{4.39}$$

Here, the *constant* for pressure is assigned to 2 in order to satisfy natural boundary conditions. Since for the given domain:

$$(\frac{\partial u}{\partial x} - p) = 2x - 2 = 0,\ \frac{\partial v}{\partial x} = 0\ on\ \Gamma_N, \tag{4.40}$$

and thus:

$$(\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) = \mathbf{0} \tag{4.41}$$

Since we defined natural boundary conditions at the outlet (Equation 4.41) we can determine both velocity and pressure simultaneously using finite element discretization with P1-P2 Taylor-Hood elements. The streamlines for analytical solution of the velocity and pressure are presented in Figure 4.2.1.

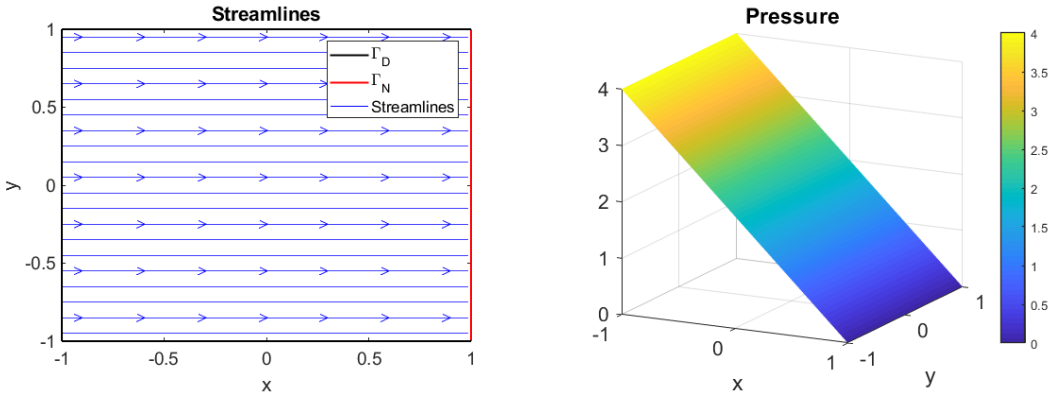

Figure 4.18. Streamlines (on the left) and pressure field (on the right) of analytical solutions of Poiseuille Flow.

By using P1 approximation for pressure and P2 for velocity, FEM approximates both unknowns exactly, since analytic velocity and pressure are of order 2 and 1, respectively[17] . Therefore least square errors of both unknowns for FEM goes to machine precision so FEM errors are not presented in the errors table.

It is important to note that since we model Stokes equation here, two unknowns are present, unlike Poisson equation. In Poisson equation, the shape factor optimization was done considering the unknown $u(x, y)$ only. However, here in Stokes equation, the shape factor optimization is done considering all of the unknowns. We chose to give more importance to velocity compared to pressure. This is done because the model of FEM uses first order elements for pressure, and hence, the order of pressure error generally appears to be higher than velocity error.

Considering the machine precision achieved by FEM, the results obtained by RBFCM-MQ can be regarded as unsatisfactory.

[17]P1-P2 elements utilize second order shape functions for velocity and first order shape functions for pressure.

Table 4.13. Error results of the example 4.2.1.

δh			**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	RBFCM-MQ	u_x	1.9794e-06	1.7657e-06	2.5033e-06
		u_y	4.7929e-07	4.2519e-07	-
		p	1.4437e-05	1.3921e-05	9.8402e-06
1/8	RBFCM-MQ	u_x	1.2291e-05	1.2148e-05	3.9415e-05
		u_y	3.3508e-06	3.6430e-06	-
		p	5.9386e-05	6.4579e-05	1.8062e-05
1/16	RBFCM-MQ	u_x	4.2913e-07	4.5572e-07	2.1308e-06
		u_y	2.5804e-06	2.5861e-06	-
		p	5.1830e-06	5.2088e-06	1.7346e-05
1/32	RBFCM-MQ	u_x	5.3798e-06	6.2864e-06	1.9774e-04
		u_y	8.0867e-06	8.2964e-06	-
		p	4.6760e-05	5.0949e-05	1.2435e-03

Table 4.14. Comparisons for the example 4.2.1.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM	1.3601e+03	4.72	
	RBFCM-MQ	8.0236e+22	5.85	344.3360
1/8	FEM	5.6106e+03	21.81	
	RBFCM-MQ	2.8682e+23	57.91	126.2500
1/16	FEM	2.2626e+04	96.06	
	RBFCM-MQ	1.7708e+22	735.29	24.8759
1/32	FEM	9.0695e+04	1627.93	
	RBFCM-MQ	9.4179e+22	12739.50	16.6113

The exact value of the velocity in the y-direction, u_y, is 0 everywhere, therefore, relative error calculation for u_y is meaningless.

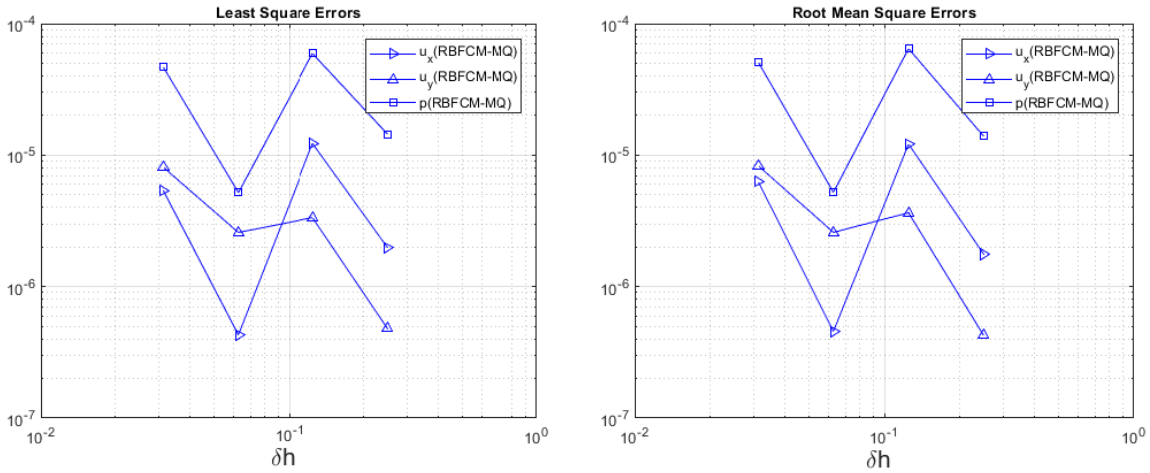

Figure 4.19. Least square errors (left), and root mean square errors (right) of the Example 4.2.1.

4.2.2. Colliding Flow

In the Colliding Flow, the boundary conditions are different than the Poiseuille Flow explained above even though the domain is the same ($\Omega = (-1,1) \times (-1,1)$). Rather than using natural boundary conditions at the outflow, Dirichlet boundary conditions are defined for velocity at all boundaries ($\partial\Omega = \Gamma_D$). This leads to a situation where pressure is determined only up to a constant. And hence, the pressure is not uniquely defined [41]. Because of this problem, a special attention must be given to pressure. A couple of techniques to overcome this problem is available in the literature [44].

We chose to deal with this problem as such: Even though it has no physical meaning, rather than satisfying continuity at the upper right node(x, y=(1,1)), exact value of the pressure is forced as a Dirichlet boundary condition in order to obtain a unique numerical solution for pressure unknown. This technique is used for both FEM and RBFCM. And as a result, the pressure field is uniquely determined where it is compatible with the analytical solution, rather than shifting up or down around the analytical solution.

Analytical solutions chosen for the unknowns of Colliding Flow are:

$$u_x = 20xy^3,\ u_y = 5x^4 - 5y^4,\ p = 60x^2y - 20y^3 + constant \tag{4.42}$$

There is no *necessary choice* for the *constant* in pressure since no Neumann boundaries are defined. Thus, to make things easier, the constant is assigned to 0. The immediate result of this is $p(1,1) = 40$ and this value is inserted as a boundary value for pressure as explained above.

The streamlines for analytical solution of the velocity and pressure of Colliding Flow are presented in Figure 4.2.2.

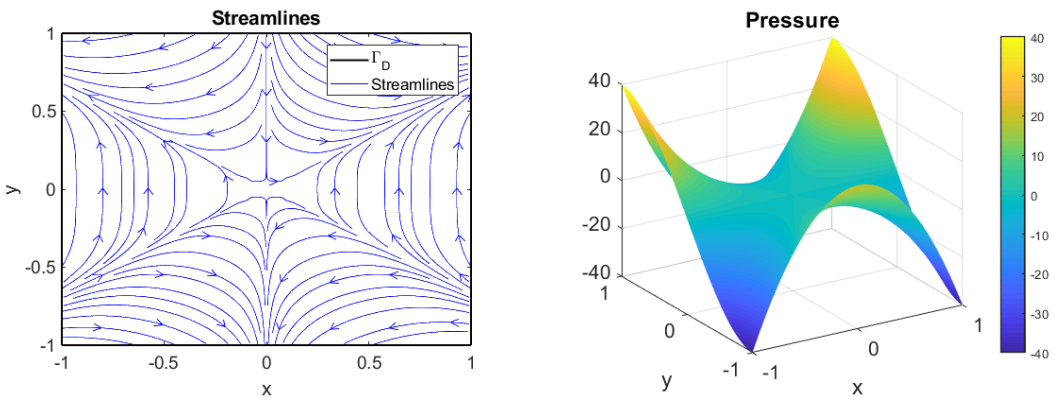

Figure 4.20. Streamlines (on the left) and pressure field (on the right) of analytical solutions of Colliding Flow.

Colliding flow includes higher order terms in analytical solutions compared to Poiseuille flow. That is why FEM approximation is not able to achieve a machine precision. In Table 4.15, the error values are presented quantitatively. Considering LSE results, the general observation which also appeared most of the examples before, RBFCM-MQ has better convergence for bigger δh values. As δh gets smaller, FEM approximation outruns RBFCM-MQ because of the better convergence rate.

Table 4.15. Error results of Example 4.2.2.

δh			**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM	u_x	1.9972e-01	4.4310e-02	6.2741e-01
		u_y	1.4875e-01	2.0329e-00	1.3267e-01
		p	3.3812e-00	3.2249e-00	4.97610e-01
	RBFCM-MQ	u_x	4.6942e-03	4.2300e-03	6.1035e-03
		u_y	4.2896e-03	4.6183e-03	1.6503e-02
		p	3.7017e-02	4.5724e-02	1.0283e-02
1/8	FEM	u_x	2.4987e-02	3.2143e-03	5.2571e-01
		u_y	1.7962e-02	1.8344e-00	1.1705e-01
		p	7.7134e-01	5.2419e-01	3.8031e-01
	RBFCM-MQ	u_x	3.6885e-03	4.2093e-03	4.3749e-02
		u_y	2.8917e-03	3.1027e-03	3.3828e-01
		p	1.0464e-01	1.0427e-01	3.4701e-01
1/16	FEM	u_x	3.1175e-03	2.1891e-04	5.2466e-01
		u_y	2.2150e-03	1.7410e-00	1.1679e-01
		p	1.8330e-01	1.0462e-01	3.3492e-01
	RBFCM-MQ	u_x	1.6588e-03	1.7104e-03	3.0781e-01
		u_y	1.4713e-03	1.9796e-03	1.0000e-00
		p	5.4580e-02	5.5262e-02	1.3650e-00
1/32	FEM	u_x	3.8929e-04	1.4446e-05	5.2485e-01
		u_y	2.7564e-04	1.6895e-00	1.1733e-01
		p	4.6266e-02	2.4312e-02	9.8171e-01
	RBFCM-MQ	u_x	6.1429e-04	6.0686e-04	1.4422e-00
		u_y	1.1515e-03	1.2767e-03	3.5562e-00
		p	7.3298e-02	7.3353e-02	1.5131e+01

Table 4.16. Comparisons for the example 4.2.1.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM	2.9031e+04	4.67	
	RBFCM-MQ	1.1580e+20	5.41	34.9777
1/8	FEM	3.9191e+05	19.73	
	RBFCM-MQ	1.5021e+21	60.00	31.7811
1/16	FEM	5.6720e+06	97.59	
	RBFCM-MQ	2.1011e+22	405.94	7.5164
1/32	FEM	8.5846e+07	1175.59	
	RBFCM-MQ	7.1881e+21	7542.8467	2.0341

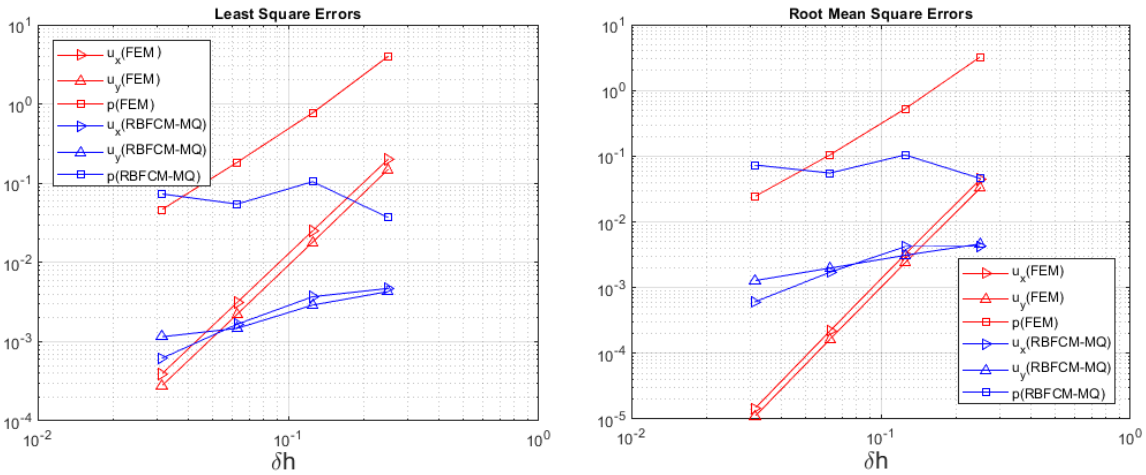

Figure 4.21. Least square errors (left), and root mean square errors (right) of the Example 4.2.2.

4.2.3. Unsteady Stokes Equation with Natural Boundary Conditions

A time dependent Stokes Equation is modelled in this example. The domain is chosen to be L-shaped likewise in the Example 4.1.3 and 4.1.6. The RBFCM solutions for randomized nodes are also present here. Backward Euler Method is used as a time integrator. Time step size, δt, is chosen to be 0.01 and all the results are obtained at the final time step. The natural boundary condition is applied at the right boundary

($x = 1$) and rest of the boundaries are Dirichlet type. This example has very basic analytical solutions and serves as a step between steady Stokes Equation examples and the next example. Only time dependent variable here is the velocity in the x-direction, u_x.

The unsteady Stokes Equation with the incompressibility condition are as follows:

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{0} \tag{4.43}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{4.44}$$

and thus, the weak form of Equations 4.43 and 4.44 are:

$$\int_T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot v \; d\Omega dt + \int_T \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega dt - \int_T \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega dt = \int_T \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} \; dS dt \tag{4.45}$$

$$\int_\Omega (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{4.46}$$

and the partial forms for RBFCM discretization are:

$$\frac{\partial u_x}{\partial t} - \frac{\partial^2 u_x}{\partial x^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial p}{\partial x} = 0 \tag{4.47}$$

$$\frac{\partial u_y}{\partial t} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial p}{\partial y} = 0 \tag{4.48}$$

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \tag{4.49}$$

Analytical solutions chosen for this example are as follows:

$$u_x = t + 1 - y^3, \; u_y = -x^3 + 3x^2 - 3x, \; p = -6xy - x + 6y + constant \tag{4.50}$$

Here, the constant term in the pressure is set to 1 to apply natural boundary conditions at the boundary $x = 1$. By doing this, the term on the right hand side of the Equation 4.45 becomes 0 at $x = 1$.

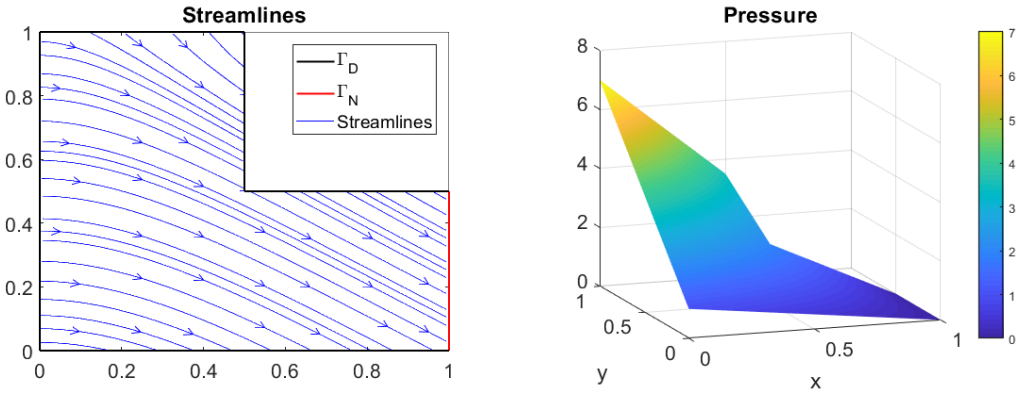

Figure 4.22. Streamlines (on the left) and pressure field (on the right) of analytical solutions of Example 4.2.3 at the final time step, $T_f = 50$.

This example additionally includes a RBFCM-MQ model with non-uniform, randomized nodes in the domain, RBFCM-MQ*. However, there is not a significant improvement considering the error results. In general, the approximation of FEM-P2 was better in the previous unsteady problems and it converged better in all δh values. In this example, RBFCM-MQ and RBFCM-MQ* was able to approximate the analytical solution better for the biggest δh value. Anyhow, considering the overall performance, FEM was better to approximate the analytical solution.

Even though RBFCM-MQ model includes a shape factor optimization algorithm, the time of this model to run lasted shorter than that of FEM, excluding $\delta h = 1/16$. This is not observed in the previous examples because in this example, we updated our optimization algorithm such that to find the optimized value, the model run for less number of shape factors.

Table 4.17. Error results of the example 4.2.3 at the final time step, T_f.

δh			**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM	u_x	4.6628e-04	1.8759e-04	2.3959e-04
		u_y	4.7669e-04	1.8331e-04	1.2356e-03
		p	2.1147e-02	3.5392e-02	4.97610e-01
	RBFCM-MQ	u_x	1.8012e-05	1.7232e-05	1.8468e-05
		u_y	1.4943e-04	1.7032e-04	3.3508e-04
		p	1.2197e-03	1.8030e-03	7.3884e-03
	RBFCM-MQ*	u_x	4.0589e-05	5.1709e-05	5.9294e-05
		u_y	5.6369e-05	6.0874e-05	1.7609e-04
		p	1.6318e-03	2.1545e-03	3.3389e-03
1/8	FEM	u_x	5.8171e-05	1.8849e-05	3.1795e-05
		u_y	5.9125e-05	1.8206e-05	1.6212e-04
		p	5.2391e-03	8.0887e-03	7.5903e-02
	RBFCM-MQ	u_x	1.7702e-05	1.9720e-05	6.0884e-05
		u_y	3.6500e-05	2.4775e-05	1.6177e-04
		p	2.6182e-04	4.3492e-04	9.2685e-04
	RBFCM-MQ*	u_x	1.7900e-05	2.0393e-05	4.6186e-05
		u_y	2.1737e-05	3.7042e-05	9.4831e-04
		p	5.5126e-04	6.6899e-04	1.9989e-02
1/16	FEM	u_x	7.2780e-06	1.7412e-06	3.5158e-06
		u_y	7.3471e-06	1.7134e-06	2.0932e-05
		p	1.3098e-03	1.9713e-03	3.7170e-02
	RBFCM-MQ	u_x	9.8270e-05	1.1409e-04	2.2447e-04
		u_y	2.7878e-05	1.2289e-04	7.1115e-04
		p	9.8471e-04	1.1958e-03	3.5241e-03
	RBFCM-MQ*	u_x	2.3109e-05	2.5442e-05	7.2811e-05
		u_y	1.0539e-04	3.1263e-05	9.7626e-04
		p	1.2230e-03	1.4015e-03	4.0332e-02

Table 4.17. Error results of the Example 4.2.3 at the final time step, T_f. (cont.)

δh			**LS Error**	**RMS Error**	**Max. Rel. Error**
1/32	FEM	u_x	9.1060e-07	1.5642e-07	3.9530e-07
		u_y	9.1523e-07	1.5588e-07	2.6379e-06
		p	3.2752e-04	4.8946e-04	1.8494e-02
	RBFCM-MQ	u_x	1.5661e-04	1.8096e-04	3.2589e-04
		u_y	1.4859e-04	1.7138e-04	2.2407e-03
		p	4.0034e-03	4.7539e-03	2.0112e-02
	RBFCM-MQ*	u_x	9.6380e-04	1.1053e-03	2.3545e-03
		u_y	3.2839e-04	3.9629e-04	2.7625e-02
		p	8.7968e-04	1.0716e-03	5.9108e-01

Table 4.18. Comparisons for the example 4.2.3.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM	4.2349e+04	5.35	
	RBFCM-MQ	4.3688e+20	1.11	35.3288
	RBFCM-MQ*	6.2620e+20	1.92	31.9055
1/8	FEM	1.2923e+05	25.85	
	RBFCM-MQ	3.4924e+21	7.5	30.8750
	RBFCM-MQ*	3.5862e+20	11.98	33.6399
1/16	FEM	4.4145e+05	183.15	
	RBFCM-MQ	2.6432e+22	830.82	38.4791
	RBFCM-MQ*	1.4787e+22	222.77	32.3054
1/32	FEM	1.6226e+06	2449.58	
	RBFCM-MQ	6.2344e+22	1862.99	1.1354
	RBFCM-MQ*	1.1646e+24	3130.00	6.8409

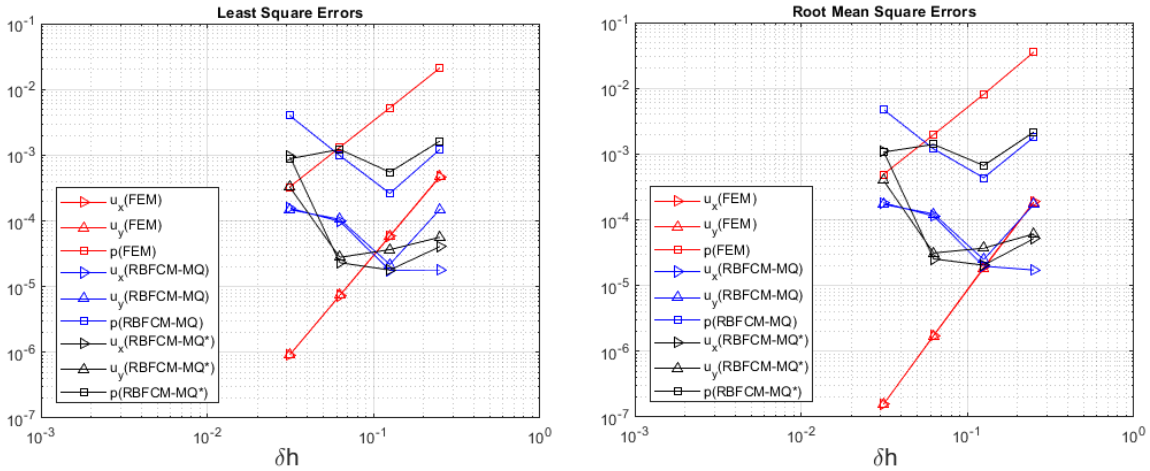

Figure 4.23. Least square errors (left), and root mean square errors (right) of the Example 4.2.3 at the final time step, $T_f = 50$.

4.2.4. Unsteady Stokes Equation with Dirichlet Boundary Conditions

A time dependent Stokes Equation only with Dirichlet boundary conditions is modelled in this example. The domain is a unit square. The reader may refer to Figure 4.1.1 to visualize the domain used here. Likewise other unsteady examples, a simple Backward Euler Method is used here. Time interval, δt, is set to be 0.00675 and all the results are obtained at time step 100. Since all the boundaries are of Dirichlet type, the problem about the uniqueness of the pressure explained in Example 4.2.2 is also present in this example. The same technique used in Example 4.2.2 is used here to overcome the problem for finite element discretization. Additionally, this example includes a source term distinct from previous Stokes examples.

The unsteady Stokes Equation with a source and the incompressibility condition are as follows:

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \tag{4.51}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{4.52}$$

and thus, the weak form of Equations 4.51 and 4.52 are:

$$\int_T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot v \; d\Omega dt + \int_T \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega dt - \int_T \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega dt$$
$$= \int_T \int_{\partial\Omega} \mathbf{f} \cdot \mathbf{v} \; d\Omega dt + \int_T \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} \; dS dt \tag{4.53}$$
$$\int_\Omega (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{4.54}$$

$\mathbf{f} \cdot \mathbf{v}$ on the right hand side is a vector rather than a scalar. The result is a matrix multiplication of $\mathbf{f}$ and transpose of $\mathbf{v}$. The partial forms for RBFCM discretization are:

$$\frac{\partial u_x}{\partial t} - \frac{\partial^2 u_x}{\partial x^2} - \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial p}{\partial x} = f_x \tag{4.55}$$
$$\frac{\partial u_y}{\partial t} - \frac{\partial^2 u_y}{\partial x^2} - \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial p}{\partial y} = f_y \tag{4.56}$$
$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0 \tag{4.57}$$

Analytical solutions for this example is set in to following:

$$u_x = 3xy^2 t, \; u_y = -y^3 t, \; p = xy \tag{4.58}$$

Then the source term $\mathbf{f}$ is given by $\mathbf{f} = \frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p$:

$$\mathbf{f} = \begin{bmatrix} 3xy^2 - 6xt + y \\ -y^3 + 6yt + x \end{bmatrix}$$

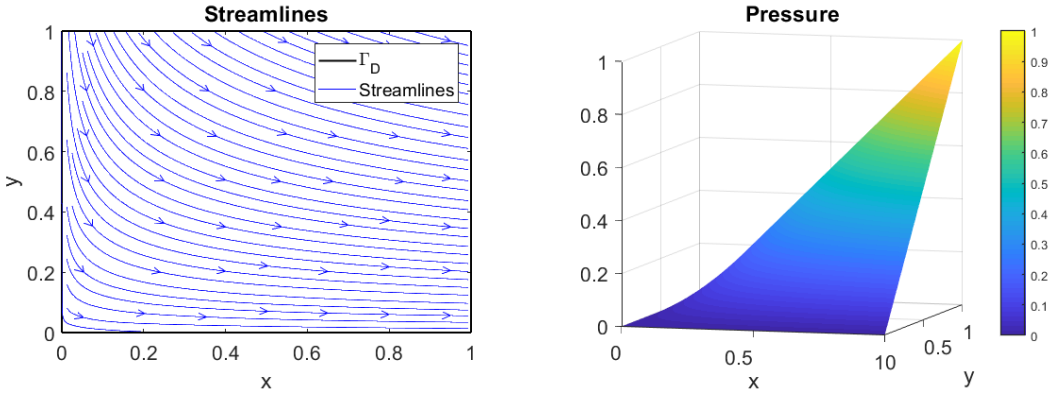

Figure 4.24. Streamlines (on the left) and pressure field (on the right) of analytical solutions of Example 4.2.4 at the final time step, $T_f = 100$.

Considering the error Figure 4.25, the convergence behaviour of both methods is almost the same with the previous unsteady Stokes problem. This means that the sensitivity of the methods do not differ switching from Neumann boundary condition to only Dirichlet boundary condition.

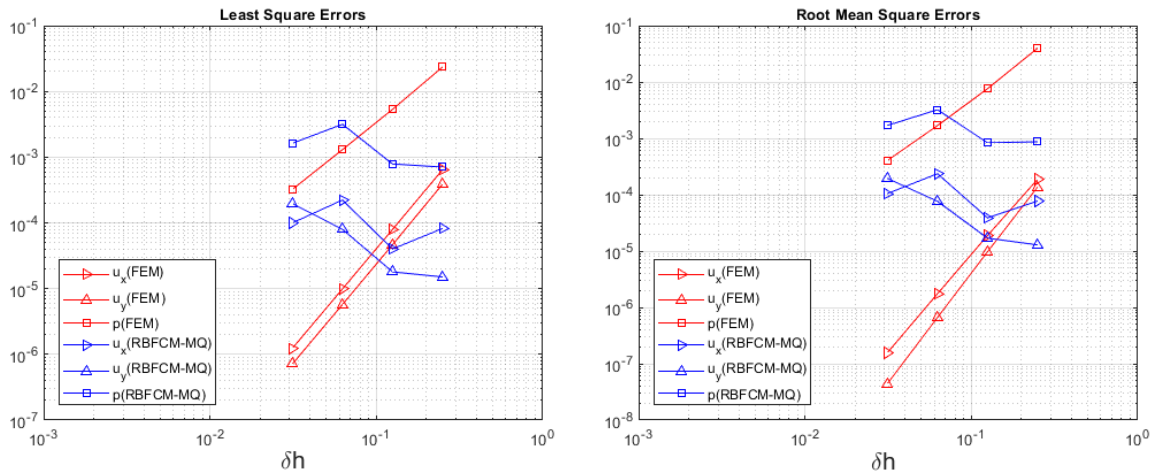

Figure 4.25. Least square errors (right), and root mean square errors (left) of the Example 4.2.4 at the final time step, $T_f = 100$.

Table 4.19. Error results of the example 4.2.4 at the final time step, $T_f = 100$.

δh			**LS Error**	**RMS Error**	**Max. Rel. Error**
1/4	FEM	u_x	6.4507e-04	1.8759e-04	7.9539e-02
		u_y	3.8894e-04	1.3675e-04	5.0000e-01
		p	2.3473e-02	3.9753e-02	4.2700e-01
	RBFCM-MQ	u_x	8.2281e-05	7.7071e-05	1.4624e-03
		u_y	1.5040e-05	1.2843e-05	5.1951e-04
		p	7.0599e-04	8.6863e-04	1.0931e-02
1/8	FEM	u_x	7.9339e-05	1.8849e-05	7.9634e-02
		u_y	4.6437e-05	9.6878e-06	4.9999e-01
		p	5.3242e-03	7.6087e-03	4.2019e-01
	RBFCM-MQ	u_x	4.0064e-05	3.8748e-05	1.1285e-02
		u_y	1.7997e-05	1.6974e-05	6.2360e-03
		p	7.8658e-04	8.3425e-04	5.7940e-02
1/16	FEM	u_x	9.8682e-06	1.7412e-06	7.9630e-02
		u_y	5.7202e-06	6.6058e-07	4.9999e-01
		p	1.2988e-03	1.6822e-03	4.1963e-01
	RBFCM-MQ	u_x	2.2267e-04	2.3605e-04	6.1136e-01
		u_y	7.9747e-05	7.6357e-05	3.6352e-01
		p	3.1598e-03	3.2237e-03	2.1383e-01
1/32	FEM	u_x	1.2317e-06	1.5642e-07	7.9629e-02
		u_y	7.1195e-07	4.4128e-08	5.0000e-01
		p	3.2263e-04	4.0344e-04	4.1948e-01
	RBFCM-MQ	u_x	9.9887e-05	1.0453e-04	2.6081e-01
		u_y	1.9680e-04	1.9448e-04	1.4239e-01
		p	1.6239e-03	1.6925e-03	5.1358e-01

Table 4.20. Comparisons for the example 4.2.4.

δh		**Cond. No**	**Runtime (sec)**	**Opt. Shape Factor**
1/4	FEM	2.3245e+05	82.18	
	RBFCM-MQ	1.7164e+24	2.05	36.6162
1/8	FEM	1.9987e+06	335.55	
	RBFCM-MQ	1.1053e+25	13.20	34.7811
1/16	FEM	2.4462e+07	1589.55	
	RBFCM-MQ	1.1567e+27	144.31	22.4962
1/32	FEM	3.5100e+08	11464.21	
	RBFCM-MQ	4.9034e+26	4408.80	13.3694

5. CONCLUSION

This work is prepared aiming to present a fair comparison between a meshless (RBFCM) and a mesh-needed (FEM) numeric models in the context of Poisson and Stokes equations. As it has been presented in Introduction, meshless Radial Basis Function Collocation Method (RBFCM) is relatively new in the area of numerical modelling. In order to have an idea about the performance of this *new* meshless method, it is imperative to compare it with other well-studied and widely used methods such as Finite Difference Method, Finite Volume Method or Finite Element Method. While surveying the literature, it has been observed that the direct and detailed comparison between RBFCM and FEM is not present. This thesis aims to fill this gap by comparing two methods directly, considering various parameters in detail, quantitatively including errors, condition numbers, and runtime. Other qualitative comparison parameters are also discussed later in this chapter. The differential equations that these methods are used to model are first, Poisson equation, which has a wide range of use in almost every field of Physics, and second, Stokes equation, which governs the motion of non-convective fluids. Both equations are modelled over different set of domains and different type of boundary conditions. Further, this work includes time dependent problems in addition to steady ones.

Throughout Chapter 4, we approximated the unknowns of our problems using different sub-types of both RBFCM and FEM[18] . Therefore, overall observations on each of these sub-types and experiences as a user of both methods are:

Finite Element Method with First Order Elements (FEM-P1): This is the most basic version of FEM. Even though approximations made by it have bigger least square error(LSE), root mean square error(RMSE) and maximum relative error(MRE), in general, it reaches the expected convergence rate, and thus, making it at least a robust and reliable method. It is observed that the FEM-P1 models have low runtime. Especially

[18]FEM-P1, FEM-P2, RBFCM-MQ, RBFCM-TPS are referred as sub-types here.

for the smallest δh, *i.e.* for the finest mesh, value that we use, no other method, including RBFCM-TPS, run faster. Moreover, well-conditioned system matrices generated by FEM-P1 also increase its robustness. On the other hand, the implementation of the method was relatively difficult. One reason of this difficulty was because we used our own algorithm to generate mesh and its connectivity, rather than using a software for creating mesh. However, putting that aside, FEM approximations include an integration process which immediately makes the method in need of a numerical integrator. The numerical integrator that we used was Gaussian quadratures, and even though, it estimates the integral almost to a certain degree, the additional burden of this process make the implementation of the method harder. Final remarks on FEM-P1 can be summarized as such: It is a reliable method. However, if accuracy is the significant criteria, other alternatives or higher degrees should be looked for.

Finite Element Method with Second Order Elements (FEM-P2): This is the second order version of FEM-P1. The results achieved by it is noteworthy since it is the most accurate one among the methods we use. Additionally, it has a very good convergence rate, which makes it very robust like its first order counterpart. Nevertheless, these advantages comes with a cost of increase in degree of freedom. Despite using the same number of elements for the same δh values, FEM-P2 utilizes many more nodes inside the domain compared to FEM-P1. An immediate downside of this is the runtime. Even in some cases, the runtime of our FEM-P2 algorithm appears to be bigger than RBFCM-MQ in which a time consuming shape factor optimization algorithm has been implemented. Moreover, it must be borne in mind that these additional nodes mean a finer mesh in a sense. Thus making it less fair to RBFCM since the total number of nodes are bigger in FEM-P2 for the same δh value. Other than that, the condition numbers of system matrices generated by this method is relatively small, which increases its reliability also. Considering the ease of implement, same arguments made for FEM-P1 can also be made for FEM-P2. In addition to those arguments, it can be said that the additional algorithms to deal with the connectivity of extra nodes and extra work brought by the increase in number of local shape factors made the overall algorithm of this method even harder to implement. Therefore, final

remarks on FEM-P2 can be summarized as such: It is an accurate and reliable method with the downsides of runtime and difficulty in implementation.

Radial Basis Function Collocation Method with Multiquadrics (RBFCM-MQ): This method is a meshless method utilizing multiquadrics as RBFs. Because of its meshless nature, any problem with, even non-rectangular, convex domain can easily be modelled with it. The accuracy of this method is noteworthy and it can be said that it approximated the analytical solutions of most examples best for bigger δh values, *i.e.* less collocation nodes. However, the multiquadrics needs an arbitrary constant, the shape factor, and because of this arbitrary constant, the robustness of the method is doubtful. Additionally, in our RBFCM-MQ algorithms, to get the most accurate result available, we implemented a shape factor optimization algorithm which increased the runtime considerably. Using one of the proposed shape factors in the literature is also a time saving option, but we did not choose this option since our aim was to get the most accurate results. Additionally, it is observed that using more nodes did not necessarily increased its convergence. Moreover, the condition number of the system matrices generated by this methods is extremely large, making its reliability shady. On the other hand, the implementation of this method is relatively effortless especially considering all the additional processes that should be done for FEM. Thus, it can be concluded as such: It is a highly accurate method for bigger δh values with very easy model implementation. However, the robustness of the method is doubtful considering the lack of h-convergence.

Radial Basis Function Collocation Method with Thin Plate Splines (RBFCM-TPS): The discussion made for RBFCM-MQ about the advantages of being meshless method is completely applicable also for this method. Even though the method is worse accurate for the biggest δh value we use, because of its excellent convergence rate observed in many examples, it achieves a fair amount of accuracy for smaller δh values, even outrunning FEM-P2 in some cases. It does not include a shape factor like RBFCM-MQ. That is why an optimization algorithm is not needed and runtime of this method, overall, is the best. Additionally, the implementation of the method is

even easier than that of RBFCM-MQ because of the same fact. The condition number of the system matrices is not large as much as that of RBFCM-MQ; however, it is bigger than that of FEM-P1 and P2. Since the condition numbers are not extremely large and ill-conditioned, this method can be regarded as a robust one. An interesting observation about the error results of this method was that the domain with non-uniform, randomized nodes achieved more accurate results. We were not able to reason this happening. To conclude, final remarks on RBFCM-TPS can be summarized as such: It is mostly accurate, reliable, robust, easy to implement and time-efficient only with a downside of bad accuracy for relatively small number of nodes, or big δh values.

In addition to these specific observations for each sub-type, it is observed that the sensitivity of both RBFCM and FEM to boundary conditions is the same, meaning that the difference in the accuracy did not change considerably by switching the boundary conditions. However, it is observed that FEM achieved a slightly better compatibility with the time integrator we use. Therefore, its deterioration in unsteady problems was less than that of RBFCM compared to steady problems.

Based on these conclusions, a further research on implementing RBFCM-TPS on Stokes equation can be considered to see its performance on this equation. Moreover, the same analysis made for Poisson and Stokes equation here can be conducted on Navier-Stokes equation to test both methods performance on modeling the real fluid motion. An additional recommendation would be to carry out the same analysis with higher order finite elements versus improved versions of RBFCM.

REFERENCES

1. McHenry, D., "A Lattice Analogy for the Solution of Stress Problems.", *Journal of the Institution of Civil Engineers*, Paper No. 5350., 1943.

2. Zienkewicz, O.C., and Taylor, *The Finite Element Method*, 5th ed., Butterworth-Heinemann, Oxford, 2000.

3. Turner, M.J., and Clough, and Martin, and Topp, "Stiffness and Deflection Analysis of Complex Structures", *Journal of the Aeronautical Sciences*, Vol. 23(9), pp. 805-823, 1956.

4. Clough, R.W., "The Finite Element Method in Plane Stress Analysis", *Proceedings of 2nd ASCE Conference on Electronic Computation*, Pittsburgh Pa., Sept. 8 and 9, 1960.

5. Temam, R., *Navier-Stokes Equations*, ISBN 0-7204-2840-8, North-Holland Publishing Company, New York, 1977.

6. Thomasset, F., *Implementation of Finite Element Methods for Navier-Stokes Equations*, ISBN 978-3-642-87049-1, Springer-Verlag New York Inc., New York, 1981.

7. Glowinski, R., and Periaux, and Pironneau, "Use of Optimal Control Theory for the numerical simulation of transonic flows by the method of finite elements", *In Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, Lecture Notes in Physics, Vol. 59, Springer, Berlin, 1976.

8. Nickell, R.E., and Tanner, and Caswell, "The solution of viscous incompressible jet and free-surface flows using finite-element methods", *Journal of Fluid Mechanics*, Vol. 65(1), pp. 189-206, 1974.

9. Chavent, G., "Finite Element Method for the Water Flooding Problem", *Colloque International sur les Methodes de Calcul Scientifique et Technique*, Versailles, 1979.

10. Benim, A.C., Zinser, "Investigation into the finite element analysis of confined turbulent flows using a κ-ϵ model of turbulence", *Computer Methods in Applied Mechanics and Engineering*, Vol. 51(1-3), pp. 507-523, 1985.

11. Volker, J., and Kindl, "Numerical studies of finite element variational multiscale methods for turbulent flow simulations", *Computer Methods in Applied Mechanics and Engineering*, Vol. 199(13-16), pp. 841-852, 2010.

12. Tezduyar T.E., Takizawa, Moorman, Wright, Christopher, "Space–time finite element computation of complex fluid–structure interactions", *International Journal for Numerical Methods in Fluids*, Vol. 64(10-12), 2010.

13. Oshima, M., Torii, Kobayashi, Taniguchi, Takagi, "Finite element simulation of blood flow in the cerebral artery", *Computer Methods in Applied Mechanics and Engineering*, Vol. 191(6-7), pp.661-671, 2001.

14. Jin, J., *The Finite Element Method in Electromagnetics*, 3rd ed., ISBN 978-1-118-57136-1, Wiley, 2014.

15. Marburg, S., and Nolte, *Computational Acoustics of Noise Propagation in Fluids - Finite and Boundary Element Methods*, ISBN 978-3-540-77447-1, Springer, 2008.

16. Liu, W.K., and Liu, and Farrell, and Zhang, "Immersed finite element method and its applicationsto biological systems", *Computational Methods in Applied Mechanics and Engineering*, Vol. 195, pp. 722–1749, 2006.

17. Naghibi Beidokhti, H., and Khoshgoftar, and Sprengers, and Van den Boogaard, "A comparison between dynamic implicit and explicit finite element simulations of the native knee joint", *Medical Engineering and Physics*, Vol. 38(10), pp. 1123-1130, 2016.

18. Arregui-Mena, J.D., and Margetts, and Mummery, "Practical Application of the Stochastic Finite Element Method", *Archives of Computational Methods in Engineering*, Vol. 23(1), pp. 171–190, 2016.

19. Hardy, R.L., "Multiquadric Equations of Topography and Other Irregular Surfaces", *Journal of Geophysical Research*, Vol. 76(8), 1971.

20. Hardy, R.L., "Research results in the application of multiquadric equations to surveying and mapping problems", *Surveying and Mapping*, Vol. 35, pp. 321-332, 1975.

21. Franke, R., "Scattered Data Interpolation: Test of Some Methods", *Mathematical Computation*, Vol. 38, pp. 181–200, 1982.

22. Harder R.L., and Desmarais,"Interpolation using surface splines," *Journal of Aircraft*, Vol. 9, pp. 189-191, 1972.

23. Meinguet, J., "Multivariate Interpolation at Arbitrary Points Made Simple", *Journal of Applied Mathematics and Physics*, Vol. 30, 1979.

24. Kansa, E.J., "Multiquadrics - A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics I-Surface Approximations and Partial Derivative Estimates", *Computers and Mathematics with Application*, Vol. 19(8), pp. 127-145, 1990.

25. Kansa, E.J., "Multiquadrics - A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics II-Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations", *Computers and Mathematics with Application*, Vol. 19(9), pp. 147-161, 1990.

26. Kansa, E.J., and Carlson, "Improved Accuracy of Multiquadric Interpolation Using Variable Shape Parameters", *Computers and Mathematics with Application*, Vol. 24(12), pp. 99-120, 1992.

27. Kansa, E.J., and Hon, "Circumventing the Ill-Conditioning Problem with Multiquadric Radial Basis Functions: Applications to Elliptic Partial Differential Equations", *Computers and Mathematics with Application*, Vol. 39, pp. 123-137, 2000.

28. Sarler B., "Towards a mesh-free computation of transport phenomena", *Engineering Analysis with Boundary Elements*, Vol. 26, pp. 731-738, 2002.

29. Sarler B., and Perko, and Chen, "Radial basis function collocation method solution of natural convection in porous media", *International Journal of Numerical Methods for Heat and Fluid Flow*, Vol. 14(2), pp. 187-212, 2004.

30. Lee, C.K., and Liu, and Fan, "Local multiquadric approximation for solving boundary value problems", *Computational Mechanics*, Vol. 30, pp. 396-409, 2003.

31. Sarler B., and Vertnik, "Meshfree Explicit Local Radial Basis Function Collocation Method for Diffusion Problems", *Computers and Mathematics with Applications*, Vol. 51, pp. 1269-1282, 2006.

32. Kosec, C., and Sarler, "Local RBF Collocation Method for Darcy Flow", *Computer Modeling in Engineering and Sciences*, Vol. 25(3), pp. 197-207, 2008.

33. Islam, S., and Vertnik, and Sarler, "Local radial basis function collocation method along with explicit time stepping for hyperbolic partial differential equations", *Applied Numerical Mathematics*, Vol. 67, pp. 136-151, 2013.

34. Kassab A.J., and Pepper, and Divo, *An Introduction to Finite Element, Boundary Element, and Mesless Methods with Applications to Heat Transfer and Fluid Flow*, ISBN 978-0-7918-6033-5, ASME Press, New York, 2014.

35. Hardy, R.L., "Theory and Applications of the Multiquadric-Biharmonic Method", *Computers and Mathematics with Application*, Vol. 19, pp. 163-208, 1990.

36. Tarwater, A.E., "A parameter study of Hardy's multiquadric method for scattered data interpolation", UCRL-54670, 1985.

37. Li, J., and Cheng, and Chen, "A comparison of efficiency and error convergence of multiquadric collocation method and finite element method", *Engineering Analysis with Boundary Elements*, Vol. 27, pp. 251-257, 2003.

38. Gu, Y.T., and Liu, "Meshless techniques for convection dominated problems", *Computational Mechanics*, Vol. 38, pp. 171-182, 2006.

39. Golbabai, A., and Rabiei, "A meshfree method based on radial basis functions for the eigenvalues of transient Stokes equations", *Engineering Analysis with Boundary Elements*, Vol. 36, pp. 1555-1559, 2012.

40. Tanbay, T., and Ozgener, "A comparison of the meshless RBF collocation method with finite element and boundary element methods in neutron diffusion calculations", *Engineering Analysis with Boundary Elements*, Vol. 46, pp. 30-40, 2014.

41. Elman, H., and Silvester and Wathen, *Finite Elements and Fast Iterative Solvers With Applications in Incompressible Fluid Dynamics*, ISBN: 9780199678808, Oxford University Press, New York, 2014.

42. Hon, Y.C., and Cheung, and Mao, and Kansa, "Multiquadric solutions for shallow water equations", *Journal of Hydraulic Engineering*, Vol. 125(5), 1999.

43. Wu, Z., and Hon, "Convergence error estimate in solving free boundary diffusion problem by radial basis functions method", *Engineering Analysis with Boundary Elements*, Vol. 27, pp. 73-79, 2003.

44. Brezzi, F., Fortin, M., *Mixed and Hybrid Finite Element Methods*, ISBN: 0-387-97582-9, Springer-Verlag, Michigan, 1991.

45. Guermond, J.L., and Minev, and Shen, "An overview of projection methods for incompressible flows", *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, pp. 6011-6045, 2005.

46. Su, L., "A radial basis function (RBF)-finite difference (FD) method for the backward heat conduction problem", *Applied Mathematics and Computation*, Vol. 354(1), pp. 232-247, 2019.

47. Wu, Z.M., and Schaback, "Local error estimates for radial basis function interpolation of scattered data", *Journal of Numerical Analysis*, Vol. 13, pp. 13-27, 1993.

48. Schaback, R., "Error estimates and condition numbers for radial basis function interpolation", *Advances in Compitational Mathematics*, Vol. 1995, pp. 251-264, 1995.

49. Franke, C., and Schaback, "Solving partial differential equations by collocation using radial basis functions", *Applied Mathematics and Computation*, Vol. 93, pp. 73-82, 1998.

50. Wu, Z.M., and Schaback, "Shape Preserving Properties and Convergence of Univariate Multiquadric Quasi-Interpolation", *Acta Mathematicae Applicatae Sinica*, Vol. 10(4), 1994.

51. Becker, E.B., and Carey, and Oden, *Finite Elements An Introduction*, ISBN 0-13-317057-8, Prentice-Hall, N.J., 1981.

52. Narcowich F.J., and Ward, "Norm estimates for the inverses of a general class of scattered- data radial-function interpolation matrices", *Journal of Approximation Theory*, Vol. 69, pp. 84-109, 1992.

53. Sun, X., "Norm estimates for inverses of Euclidean distance matrices", *Journal of Approximation Theory*, Vol. 70, pp. 339-347, 1992.

54. Michelli, C.A., "Interpolation of scattered data: distance matrices and con- ditionally positive definite functions", *Constructive Approximation*, Vol. 2, pp. 11-22, 1986.

55. Madych, W.R., and Nelson, "Multivariate Interpolation and conditionally positive definite functions", *Mathematics of Computation*, Vol. 4, pp. 77-89, 1988.

56. Madych, W.R., and Nelson, "Multivariate Interpolation and conditionally positive definite functions II", *Mathematics of Computation*, Vol. 54(189), pp. 211-230, 1990.

57. Bernardi, C., and Rebollo, and Yakoubi, "Finite element discretization of the Stokes and Navier-Stokes equations with boundary conditions on the pressure", *Jurnal on Numerical Analysis*, Vol. 53(3), pp. 1256-1279, 2015.

58. Bernardi, C., and Chorfi, "Spectral discretization of the vorticity, velocity and pressure formulation of the Stokes problem", *Journal of Numerical Analysis*, Vol. 44, pp. 826-850, 2006.

59. Boland, J., and Nicolaides, "Stability of finite elements under divergence constraints", *Journal of Numerical Analysis*, Vol. 20, pp. 722-731, 1983.

60. Conca, C., and Murat, and Pironneau, "The Stokes and Navier–Stokes equations with boundary conditions involving the pressure", *Japanese Journal of Mathematics*, Vol. 20(2), pp. 279-318, 1994.

61. Goldberg, M.A., and Chen, and Karur, "Improved multiquadric approximation for partial differential equations", *Engineering Analysis with Boundary Elements*, Vol. 18, pp. 9-17, 1996.

62. Powell M.J.D., "The uniform convergence of thin plate spline interpolation in two dimensions", *Numerische Mathematik*, Vol. 68(1), pp. 107-128, 1994.

63. Carlson R.E., and Foley, "The parameter R^2 in multiquadric interpolation", *Computers and Mathematics with Applications*, Vol. 21, pp.29-42, 1991.

64. Atkinson, K.E., "The numerical evaluation of particular solutions for Poisson's equation", *Journal of Numerical Analysis*, Vol. 5, pp. 319-338, 1985.

65. Donea, J., and Huerta, *Finite Element Methods for Flow Problems*, ISBN 0-471-49666-9, Wiley, West Sussex, 2003.

66. Gallinari, P., Maday, Sangnier, Schwander, Taddei, "Reduced Basis' Acquisition by a Learning Process for Rapid On-line Approximation of Solution to PDE's: Laminar Flow Past a Backstep", *Archives of Computational Methods in Engineering*, Vol. 25, pp. 131-141, 2018.

67. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed., Addison Vealey, MA, 1989.

68. Otten, R.H.J.M., and van Ginneken, *The Annealing Algorithm*, ISBN 978-1-4613-1627-5, Kluwer Academic Publishers, Norwell, 1989.

69. Zerroukat, M., and Djidjeli, and Charafi, "Explicit and implicit meshless methods for linear advection-diffusion-type partial differential equations", *International Journal for Numerical Methods in Engineering*, Vol. 48, pp. 19-35, 2000.

70. Zerroukat, M., and Power, and Chen, "A numerical method for heat transfer problems using collocation and radial basis functions", *International Journal for Numerical Methods in Engineering*, Vol. 42, pp. 1263-1278, 1998.

APPENDIX A: WEAK FORMULATIONS

A.1. Steady Poisson Equation

Strong form of steady Poisson equation is as follows:

$$\nabla \cdot (-k\nabla u) = f \tag{A.1}$$

where f is the source term, k is a constant and u is the unknown. Multiplying Equation A.1 with a test function, v, and taking integral of both sides:

$$\int_\Omega v\nabla \cdot (-k\nabla u)\ d\Omega = \int_\Omega fv\ d\Omega \tag{A.2}$$

Applying integration by parts and Divergence Theorem on the right hand side of the Equation A.2:

$$\int_\Omega v\nabla \cdot (-k\nabla u)\ d\Omega = \int_\Omega k\nabla u \cdot \nabla v\ d\Omega - \int_\Omega \nabla \cdot (k\nabla u \cdot v)\ d\Omega \tag{A.3}$$

$$= \int_\Omega k\nabla u \cdot \nabla v\ d\Omega - \int_{\partial\Omega} (k\nabla u \cdot \vec{n}\,)v\ dS \tag{A.4}$$

Let $(k\nabla u \cdot \vec{n}\,)$ equal to a function $g(x, y)$. Combining Equation A.2 with Equation A.3, the weak form is:

$$\int_\Omega k\nabla u \cdot \nabla v\ d\Omega = \int_\Omega fv\ d\Omega + \int_{\partial\Omega} gv\ dS \tag{A.5}$$

A.2. Unsteady Poisson Equation

Strong form of unsteady Poisson equation is as follows:

$$\frac{\partial u}{\partial t} + \nabla \cdot (-k\nabla u) = f \tag{A.6}$$

where f is the source term, k is a constant and u is the unknown. Multiplying Equation A.6 with a test function, v, and taking integral of both sideswith respect to both space and time:

$$\int_T \int_\Omega \frac{\partial u}{\partial t} v \; d\Omega dt + \int_T \int_\Omega v \nabla \cdot (-k\nabla u) \; d\Omega dt = \int_T \int_\Omega f v \; d\Omega dt \tag{A.7}$$

Applying the same procedure done to Equation A.2 to Equation A.7, it becomes:

$$\int_T \int_\Omega \frac{\partial u}{\partial t} v \; d\Omega dt + \int_T \int_\Omega k\nabla u \cdot \nabla v \; d\Omega dt = \int_T \int_\Omega f v \; d\Omega dt + \int_T \int_{\partial\Omega} (k\nabla u \cdot \vec{n}) v \; dS dt \tag{A.8}$$

A.3. Steady Stokes Equation

Strong form of the steady Stokes equations:

$$-\nabla^2 \mathbf{u} + \nabla p = \mathbf{0} \tag{A.9}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{A.10}$$

where $\mathbf{u}$ is the velocity vector and p is the pressure.

Multiplying equation A.9 with the second order test function $\mathbf{v}$ and equation A.10 with the first order test function q and taking integral of both sides[19] :

$$-\int_{\Omega} \nabla^2 \mathbf{u} \cdot \mathbf{v} \; d\Omega + \int_{\Omega} (\nabla p)\mathbf{v} \; d\Omega = \mathbf{0} \tag{A.11}$$

$$\int_{\Omega} (\nabla \cdot \mathbf{u}) q \; d\Omega = 0 \tag{A.12}$$

Arranging the terms in the equation A.11 using integration by parts:

$$-\int_{\Omega} \nabla^2 \mathbf{u} \cdot \mathbf{v} \; d\Omega = \int_{\Omega} \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_{\Omega} \nabla \cdot (\nabla \mathbf{u} \cdot \mathbf{v}) \; d\Omega \tag{A.13}$$

$$\int_{\Omega} \mathbf{v} \cdot \nabla p \; d\Omega = -\int_{\Omega} p(\nabla \cdot \mathbf{v}) \; d\Omega + \int_{\Omega} \nabla \cdot (p\mathbf{v}) \; d\Omega \tag{A.14}$$

Using Divergence Theorem for last terms in equations A.13 and A.14, those will become:

$$-\int_{\Omega} \nabla^2 \mathbf{u} \cdot \mathbf{v} \; d\Omega = \int_{\Omega} \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_{\partial\Omega} (\vec{n} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} dS \tag{A.15}$$

$$\int_{\Omega} \mathbf{v} \cdot \nabla p \; d\Omega = -\int_{\Omega} p(\nabla \cdot \mathbf{v}) \; d\Omega + \int_{\partial\Omega} p\vec{n} \cdot \mathbf{v} dS \tag{A.16}$$

where $\vec{n}$ is the unit vector pointing outwards from boundary. Putting equations A.15 and A.16 in A.11:

$$\int_{\Omega} \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega - \int_{\Omega} p(\nabla \cdot \mathbf{v}) \; d\Omega = \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} dS \tag{A.17}$$

[19] P1-P2 elements are used in this work. Using finite element discretization to solve Stokes equation simultaneously for both velocity and pressure unknowns, P1-P2 TH element pair is preferred, so we avoided stability issues.

A.4. Unsteady Stokes Equation

Strong form of the unsteady Stokes equations with a source term and continuity equation are :

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \tag{A.18}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{A.19}$$

where $\mathbf{u}$ is the velocity vector and p is the pressure.

Multiplying equation A.18 with the second order test function $\mathbf{v}$ and taking integral of both sides with respect to time and space:

$$\int_T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \; d\Omega dt - \int_T \int_\Omega \nabla^2 \mathbf{u} \cdot \mathbf{v} \; d\Omega dt + \int_T \int_\Omega (\nabla p) \mathbf{v} \; d\Omega dt = \int_T \int_\Omega \mathbf{f} \cdot \mathbf{v} \; d\Omega dt \tag{A.20}$$

The weak form of the continuity equation is exactly the same as Equation A.12. Also, applying the same procedure done in Equations A.13-A.16 to Equation A.20:

$$\begin{aligned} \int_T \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \; d\Omega dt + \int_T \int_\Omega \nabla \mathbf{u} \cdot \nabla \mathbf{v} \; d\Omega dt - \int_T \int_\Omega p(\nabla \cdot \mathbf{v}) \; d\Omega dt \\ = \int_T \int_\Omega \mathbf{f} \cdot \mathbf{v} \; d\Omega dt + \int_T \int_{\partial\Omega} (\frac{\partial \mathbf{u}}{\partial n} - p\vec{n}\,) \cdot \mathbf{v} dS dt \end{aligned} \tag{A.21}$$

Lastly, the weak form of the Equation A.19 is exactly the same as Equation A.12.

APPENDIX B: Gaussian Quadrature Point Weights and Locations

Table B.1. Gaussian quadrature 12 points weights and locations.

w_i	x_i	y_i
0.0254224531851035	0.873821971016996	0.0630890144915020
0.0254224531851035	0.0630890144915020	0.0630890144915020
0.0254224531851035	0.0630890144915020	0.873821971016996
0.0583931378631895	0.501426509658179	0.249286745170910
0.0583931378631895	0.249286745170910	0.249286745170910
0.0583931378631895	0.249286745170910	0.501426509658179
0.0414255378091870	0.636502499121399	0.310352451033785
0.0414255378091870	0.310352451033785	0.636502499121399
0.0414255378091870	0.636502499121399	0.0531450498448160
0.0414255378091870	0.0531450498448160	0.636502499121399
0.0414255378091870	0.310352451033785	0.0531450498448160
0.0414255378091870	0.0531450498448160	0.310352451033785

Table B.2. Gaussian quadrature 25 points weights and locations.

w_i	x_i	y_i
0.0417616999025982	0.333333333333333	0.333333333333333
0.00361492529602837	0.00426913409105029	0.497865432954475
0.00361492529602837	0.497865432954475	0.497865432954475
0.00361492529602837	0.497865432954475	0.00426913409105023
0.0372460889604903	0.143975100541888	0.428012449729056
0.0372460889604903	0.428012449729056	0.428012449729056
0.0372460889604903	0.428012449729056	0.143975100541888
0.0393232367015543	0.630487174513551	0.184756412743225
0.0393232367015543	0.184756412743225	0.184756412743224
0.0393232367015543	0.184756412743224	0.630487174513551
0.00346416154355375	0.959037562856645	0.0204812185716776
0.00346416154355375	0.0204812185716777	0.0204812185716773
0.00346416154355375	0.0204812185716773	0.959037562856645
0.0147591601673897	0.0350029898972720	0.136573576256033
0.0147591601673897	0.136573576256033	0.828423433846695
0.0147591601673897	0.828423433846695	0.0350029898972721
0.0147591601673897	0.828423433846695	0.136573576256034
0.0147591601673897	0.0350029898972720	0.828423433846695
0.0147591601673897	0.136573576256034	0.0350029898972717
0.0197896835980306	0.0375490702584427	0.332743600588639
0.0197896835980306	0.332743600588639	0.629707329152919
0.0197896835980306	0.629707329152919	0.0375490702584427
0.0197896835980306	0.629707329152919	0.332743600588639
0.0197896835980306	0.0375490702584427	0.629707329152919
0.0197896835980306	0.332743600588639	0.0375490702584425

APPENDIX C: PARTIAL DERIVATIVES OF MULTIQUADRICS AND THIN PLATE SPLINES

C.1. Partial Derivatives of Multiquadrics

The multiquadrics radial basis function is:

$$\varphi(r) = (r^2 + c^2)^{\beta/2} \tag{C.1}$$

where $r = ((x - x_j)^2 + (y - y_j)^2)^{1/2}$.

Assuming $\beta = 1$, the first order derivative with respect to x can be written as:

$$\frac{\partial \varphi}{\partial x} = \frac{\partial \varphi}{\partial r}\frac{\partial r}{\partial x} = \frac{r}{(r^2 + c^2)^{1/2}}\frac{(x - x_j)}{r} \tag{C.2}$$

$$\frac{\partial \varphi}{\partial x} = \frac{(x - x_j)}{\varphi} \tag{C.3}$$

Likewise, the first order derivative with respect to y is:

$$\frac{\partial \varphi}{\partial y} = \frac{(y - y_j)}{\varphi} \tag{C.4}$$

The second order derivative with respect to x is:

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{1}{\varphi}\frac{\partial (x - x_j)}{\partial x} + (x - x_j)\frac{\partial}{\partial r}\left[\frac{1}{\varphi}\right]\frac{\partial r}{\partial x} = \frac{1}{\varphi} + \frac{-r}{\varphi^3}\frac{(x - x_j)^2}{r} \tag{C.5}$$

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{1}{\varphi} - \frac{(x - x_j)^2}{\varphi^3} \tag{C.6}$$

Likewise, the second order derivative with respect to y is:

$$\frac{\partial^2 \varphi}{\partial y^2} = \frac{1}{\varphi} - \frac{(y - y_j)^2}{\varphi^3} \tag{C.7}$$

C.2. Partial Derivatives of Thin Plate Splines

The thin plate splines radial basis function is:

$$\varphi(r) = r^\beta \ln r, \ \beta > 0, \beta \epsilon 2N \tag{C.8}$$

where $r = ((x - x_j)^2 + (y - y_j)^2)^{1/2}$.

The first order derivative with respect to x can be written as:

$$\frac{\partial \varphi}{\partial x} = \frac{\partial \varphi}{\partial r}\frac{\partial r}{\partial x} = (\beta r^{\beta-1} \ln r + r^{\beta-1})\frac{(x - x_j)}{r} \tag{C.9}$$

$$\frac{\partial \varphi}{\partial x} = (x - x_j) r^{\beta-2}(\beta \ln r + 1) \tag{C.10}$$

Likewise, the first order derivative with respect to y is:

$$\frac{\partial \varphi}{\partial y} = (y - y_j) r^{\beta-2}(\beta \ln r + 1) \tag{C.11}$$

The second order derivative with respect to x is:

$$\frac{\partial^2 \varphi}{\partial x^2} = r^{\beta-2}(\beta \ln r + 1)\frac{\partial (x - x_j)}{\partial x} + (x - x_j)\frac{\partial}{\partial r}\left[r^{\beta-2}(\beta \ln r + 1)\right]\frac{\partial r}{\partial x} \tag{C.12}$$

$$\frac{\partial^2 \varphi}{\partial x^2} = r^{\beta-2}(\beta \ln r + 1) + (x - x_j)^2 r^{\beta-4}(2 * (\beta - 1) + \beta(\beta - 2) \ln r) \tag{C.13}$$

Likewise, the second order derivative with respect to y is:

$$\frac{\partial^2 \varphi}{\partial y^2} = r^{\beta-2}(\beta \ln r + 1) + (y - y_j)^2 r^{\beta-4}(2 * (\beta - 1) + \beta(\beta - 2) \ln r) \tag{C.14}$$