A COMPARISON OF MESHLESS AND FINITE DIFFERENCE METHODS FOR THE BRUSSELATOR MODEL

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Mathematics

by
Leyla AKBALIK

July 2022
İZMİR

ACKNOWLEDGMENTS

First, I would like to express my respect and gratitude to my supervisor Prof. Dr. Gamze Tanoğlu. It has been an honor to be her graduate student. Also, I would like to thank to the Neslişah İmamoğlu Karabaş for her patient and support.

I would like to thank my dear family for their financial and moral support during the preparation of this thesis.

Also, I am grateful to the person who is special to me for his endless support and motivation. Last but not least, I would like to thank my dear friends Shodijon, Nisa, Zeynep, and Ecem, who encouraged me during the writing process of this thesis.

ABSTRACT

A COMPARISON OF MESHLESS AND FINITE DIFFERENCE METHODS FOR THE BRUSSELATOR MODEL

The purpose of this thesis is the Brusselator model, which is used to model the reaction-diffusion occurring in processes of a chemical such as the formation of turing patterns in the skin of an animal, enzymatic reaction, ozone formation through triple collision with atomic oxygen; using methods such as Meshless Method and Finite Difference Method in space discretization and Runge Kutta Method, and the Adaptive Runge Kutta Method in time discretization, to find the method that gives more accurate. It is also to estimate the degree of the Meshless Method using the Finite Difference Method.

ÖZET

BRUSSELATOR MODELİ İÇİN AĞSIZ VE SONLU FARK YÖNTEMLERİNİN KARŞILAŞTIRILMASI

Bu tezin amacı, enzimatik reaksiyon, hayvan derisinde turing paternlerinin oluşumu, atomik oksijen ile üçlü çarpışma yoluyla ozon oluşumu gibi kimyasal süreçlerde meydana gelen reaksiyon-difüzyonu modellemek için kullanılan Brusselator modelini; uzay ayrıklaştırmasında Ağsız Yöntemi ve Sonlu Fark Yöntemi, zaman ayrıklaştırmasında Runge Kutta Yöntemi ve Uyarlanabilir Runge Kutta Yöntemi kullanılarak en iyi sonucu veren yöntemin bulunmasıdır. Ayrıca Ağsız Yönteminin derecesini, Sonlu Fark Yöntemini kullanarak tahmin etmektir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

FDM	Finite Difference Method
RK4	Runge Kutta Method
ARK	Adaptive Runge Kutta Method
RBFs	Radial Basis Functions
MQ	Multiquadric
MQ RBF	Multiquadric Radial Basis Function
ETD	Exponential Time Differencing
CN	Crank-Nicolson

CHAPTER 1

INTRODUCTION

In the modern world, nonlinear equations are significant. Finding new approaches to identify new exact or approximate solutions remains a challenging task, despite the importance of finding exact solutions to nonlinear partial differential equations in physics and applied mathematics.

Many authors have focused their attention in recent years on studying nonlinear partial differential equation solutions using diverse methodologies (Hilal and Elzaki, 2014).

In order to solve nonlinear 1D and 2D problems governed by the Brusselator model, in this thesis, compares the accuracy and computing efficiency of Meshless Method and Finite Difference Method. We will give information about the Meshless Method and Finite Difference Method (FDM) for space discretization and Runge Kutta Method (RK4), and the Adaptive Runge Kutta Method (ARK) as a time discretization.Different numerical problems are presented.

The Brusselator Model is one of the most important models in chemical reactions such as enzymatic reactions, the creation of turing patterns on animal skin, and the ozone synthesis from atomic oxygen via a triple collision with oscillations. A nonlinear reaction-diffusion equation is largely described by this model. Ilya Prigogine and his colleagues proposed the Brusselator Model (Lefever and Nicolis (1971)), (Prigogine and Nicolis, 1985), (Prigogine and Lefever, 1968), (Tyson J, 1973). Its name comes from the combination of the words Brussels and oscillator.

The Brusselator Model is based on mathematical modeling of an autocatalytic chemical reaction that is governed by

$$\tilde{\alpha} \mapsto U$$
$$\tilde{\beta} + U \mapsto V + \tilde{D}$$

$$2U + V \mapsto 3U$$

$$U \mapsto \tilde{E}$$

where $\tilde{\alpha}$ and $\tilde{\beta}$ are the input chemical, $\tilde{D}$ and $\tilde{E}$ are the output chemical, U and V are the intermediate species. Therefore, the Brusselator Model chemical reaction is described by the partial differential equations as follows

$$\frac{\partial u}{\partial t} = \tilde{\alpha} - (\tilde{\beta} + 1)u + u^2 v \tag{1.1}$$

$$\frac{\partial v}{\partial t} = \tilde{\beta} u - u^2 v \tag{1.2}$$

The Brusselator Model has a fixed point at $u = \tilde{\alpha}, v = \frac{\tilde{\beta}}{\tilde{\alpha}}$, as can be seen.
When $\tilde{\beta} > 1 + \tilde{\alpha}^2$, the fixed point of the model becomes unstable, causing the system to oscillate.
When the diffusion is added, we get the generic reaction-diffusion Brusselator Model of the form

$$\frac{\partial u}{\partial t} = \tilde{\alpha} - (\tilde{\beta} + 1)u + u^2 v + k\nabla^2 u(x,t) \tag{1.3}$$

$$\frac{\partial v}{\partial t} = \tilde{\beta} u - u^2 v + k\nabla^2 v(x,t) \tag{1.4}$$

with initial conditions and Neumann boundary conditions

$$u(x,0) = f(x), \quad v(x,0) = g(x), \quad x \in \tilde{\Omega} \tag{1.5}$$

$$\frac{\partial u(x,t)}{\partial n} = \frac{\partial v(x,t)}{\partial n} = 0, \quad (x,t) \in \partial\tilde{\Omega} \times [0,T]. \tag{1.6}$$

where $\tilde{\alpha}, \tilde{\beta}$, and k are the constants, the Laplace operator is denoted by ∇^2 and $\tilde{\Omega}$ is multi-dimensional domain (Mittal et al., 2019).

The numerical solution of the Brusselator Model is essential since the analytic solution, with or without diffusion, is unknown. Various methods may be used to estimate the solutions of 1D and 2D Brusselator Model, however, few researchers have put forward these ideas. Adomian (Adomian, 1995) used a Decomposition Method, which Wazwaz (Wazwaz, 2000) later improved, to approximate the solution of the reaction-diffusion Brusselator Model. Among the well-known mesh-based differencing algorithms developed for the Brusselator Model, the numerical approach proposed by Twizell (Twizell et al., 1999) is worth mentioning. As a further reliable and efficient computing strategy for solving the future PDEs from a coupled Brusselator Model, the Gauss-Seidel-like implicit Finite Difference Method was introduced in (Yu and Gumel, 2001). This method is unconditionally convergent and offers accuracy comparable to that of the classical Runge Kutta Method. In (Kleefeld et al., 2012), the conventional and Exponential Time Differencing (ETD) and Crank-Nicolson (CN) Methods were used to resolve the following initial-value problem from the 1-D Brusselator Model. In (Ang, 2003), a dual-reciprocity Boundary Element Method was employed to look for a 2D Brusselator system approximative solution (Bhatt and Chowdhury, 2019).

The outline of this thesis is organized as follows;
The methods we use for space discretization are presented in Chapter 2 which are the Multiquadric Radial Basis Function (MQ RBF) technique and the Finite Difference Method. Ideas of these methods are simply explained. For the MQ RBF technique, the interpolation matrix's existence and uniqueness are intuitively proved. It is shown how to approximate derivatives using MQ RBF. It is shown how to take the first and second derivatives for the Finite Difference Method.

In Chapter 3, we introduced time discretization. For this discretization, we used the Runge-Kutta method and the Adaptive Runge-Kutta method. We showed how to compute these methods.

Chapter 4 focuses on Brusselator Model. This model was solved by the presented methods. We solved this model by the combinations of the FDM with RK4 and ARK and the Meshless Method with RK4 and ARK. Numerical results are presented to show the accuracy of the method.

Finally, in Chapter 5 we present the summary and give the brief conclusion.

CHAPTER 2

SPACE DISCRETIZATION

2.1. Multiquadric Radial Basis Function

The introduction of Radial Basis Functions (RBFs) a truly meshless method for approximating the solution of partial differential equations has piqued the interest of many scientists and engineers in the recent decade. The RBFs method is an interpolation method for multidimensional scattered data (Micchelli, 1984), (Larsson and Fornberg, 2003). Ronald Hardy (Hardy, 1971), (Hardy, 1990) proposed the multiquadric (MQ) radial basis function approach in 1971. Until 1979, Hardy's MQ interpolation method went unobserved. Hardy (Hardy, 1990) proved that multiquadric RBFs are associated with a consistent solution to the biharmonic potential issue, implying that they have a physical basis. Edward Kansa was the first to apply the MQ approach to solve differential equations (Kansa I, 1990), (Kansa II, 1990) in 1990. The popularity of the MQ approach grew rapidly after it was first employed to solve PDEs, (Fasshauer, 1996), (Larsson and Fornberg, 2003) and a significant variety of applications arose (Alipanah, 2016).

2.1.1. Definiton of the Multiquadric Radial Basis Function

Hardy used the quadric surfaces

$$\tilde{\Phi}(r;c) = \sqrt{c^2 + r^2} \tag{2.1}$$

as the basis functions where $r = \|x\|_2 = \sqrt{x_1^2 + ... + x_d^2}, x \in \mathcal{R}^d$ and c represent a surface shape parameter that impacts the surface's shape. The basis function (2.1) is a variation of the Multiquadric Radial Basis Function (MQ RBF). The shape of the surface on the unit circle is shown in Figure (2.1) (Karabaş, 2020).

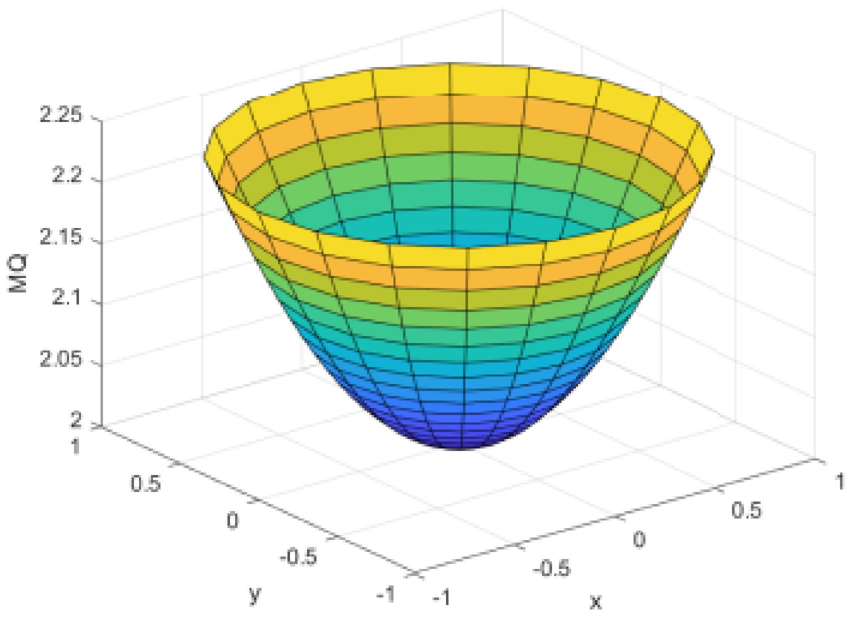

Figure 2.1. Shape of MQ RBF (2.1) with parameter c=2

2.1.2. Alternate Definition of the MQ

The MQ $\tilde{\Phi}(r;c) = \sqrt{c^2 + r^2}$ is commonly redefined by first letting $c = \frac{1}{\tilde{\varepsilon}}$, which yields

$$\tilde{\Phi}(r;\tilde{\varepsilon}) = \tilde{\varepsilon}\sqrt{1 + \tilde{\varepsilon}^2 r^2} \tag{2.2}$$

MQ can be rewritten as

$$\tilde{\Phi}(r;\tilde{\varepsilon}) = \sqrt{1 + \tilde{\varepsilon}^2 r^2} \tag{2.3}$$

if the scaling factor is disregarded (Karabaş, 2020).

The updated definition makes the MQ, like other indefinitely differentiable RBFs with a shape parameter, behave as a function of the shape parameter, $\tilde{\varepsilon}$,. When using the

MQ defined as $\tilde{\Phi}(r;c) = \sqrt{c^2 + r^2}$, we use c to represent the shape parameter, and when using the MQ defined as (2.2), we use $\tilde{\varepsilon}$. Figure 2.2 shows the MQ with different shape parameter values, demonstrating how the function increases flat as it approaches zero.

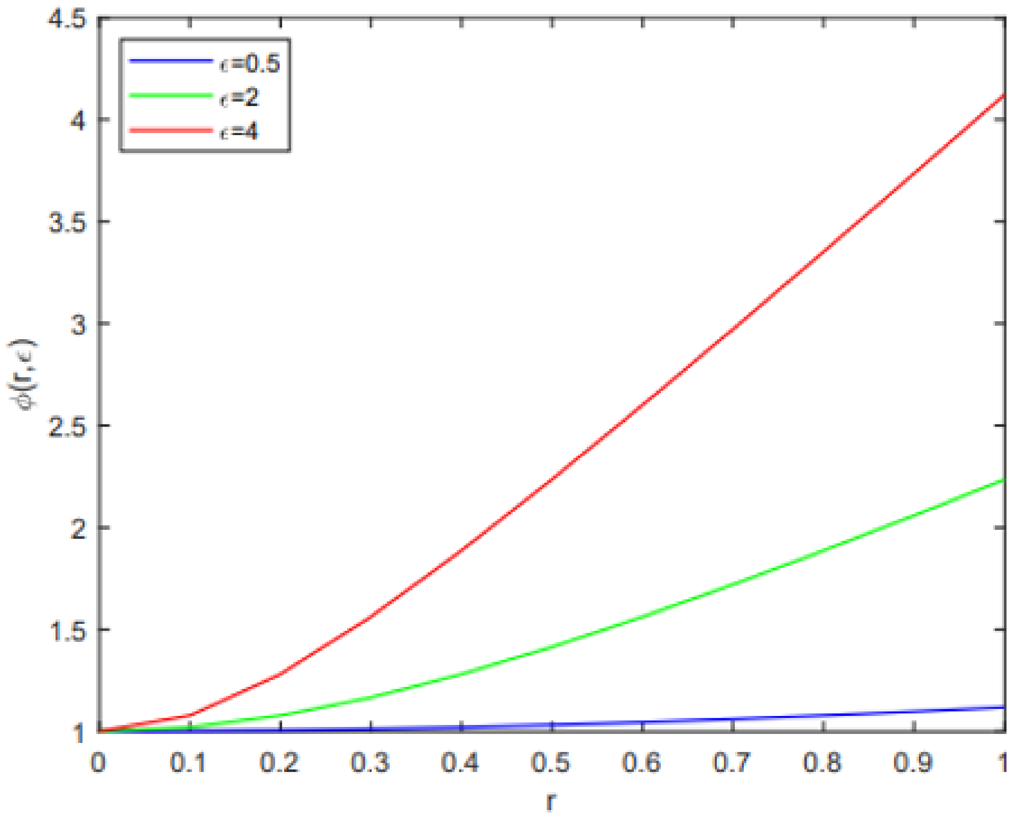

Figure 2.2. Multiquadric RBF with different parameters $\tilde{\varepsilon}$

2.2. RBF Interpolation

Linear combinations of translations of one function $\tilde{\Phi}(r)$ of a single real variable are used in RBF interpolation and given in the following form:

$$\tilde{\Psi}(x) = \sum_{k=1}^{N} \tilde{\Phi}(r,c)\lambda_k, \quad x \in \mathcal{R}^d, \tag{2.4}$$

where $\lambda_k, k = 1, 2, ..., N$ are the RBF coefficients and

$$r = \|x - x_k\|_2 = \sqrt{(x - x_1)^2 + ... + (x - x_N)^2}.$$

The implementation of the interpolation condition

$$\tilde{\Psi}(x_i) = \tilde{f}(x_i), \tag{2.5}$$

where $x_i, i = 1, ..., N$ are the set of centers yields RBF coefficients, λ_k's (Karabaş, 2020).

When the interpolation criteria are enforced at N centers, a $N \times N$ linear system

$$B\lambda = \tilde{f} \tag{2.6}$$

is created, which must be solved for the MQ expansion coefficients λ.
Entries of the matrix B are (Karabaş, 2020)

$$b_{jk} = \tilde{\Phi}(\|x_j - x_k\|_2, c) \quad j, k = 1, ..., N. \tag{2.7}$$

2.2.1. Invertibility of Interpolation Matrix

The solution to the MQ interpolation problem will exist and be unique if and only if B is invertible, as shown by equation (2.6). Micchelli's theorem (Micchelli, 1984) states that the MQ interpolation matrix is invertible in the following way.

Theorem 2.1 *Let $\tilde{\phi}(r) = \tilde{\Phi}(\sqrt{r}) \in C[0, \infty)$ and $\tilde{\phi}(r) > 0$ for $r > 0$. Let $\tilde{\phi}'(r)$ be completely monotone and nonconstant on $(0, \infty)$. Then for any set of distinct centers $\{x_j\}_{j=1}^N$, the $N \times N$ matrix B with entries $b_{jk} = \tilde{\Phi}(\|x_j - x_k\|_2)$ is invertible.*

The definition of a completely monotone function is required to show the invertibility of the MQ interpolation matrix.

Definition 2.1 *A function $\tilde{\phi}$ is completely monotone on* $[0, \infty)$ *if*
(i) $\tilde{\phi} \in C[0, \infty)$
(ii) $\tilde{\phi} \in C^{\infty}(0, \infty)$
(iii) $(-1)^{\ell}\tilde{\phi}^{\ell}(r)$ *where* $r > 0$ *and* $\ell = 0, 1, ...$

Proof: For the MQ we have

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

and

$$\tilde{\phi}'(r) = \frac{\tilde{\varepsilon}^2}{2\sqrt{1 + \tilde{\varepsilon}^2 r}},$$

$$\tilde{\phi}''(r) = \frac{-\tilde{\varepsilon}^4}{4\left(1 + \tilde{\varepsilon}^2 r\right)^{3/2}},$$

$$\tilde{\phi}^{(3)}(r) = \frac{3\tilde{\varepsilon}^6}{8\left(1 + \tilde{\varepsilon}^2 r\right)^{5/2}},$$

$$\tilde{\phi}^{(4)}(r) = \frac{-15\tilde{\varepsilon}^8}{16\left(1 + \tilde{\varepsilon}^2 r\right)^{7/2}}, \cdots$$

Hence,

$$(-1)^{\ell}\tilde{\phi}^{\ell}(r) \geq 0$$

And $\tilde{\phi}'(r)$ is completely monotone and MQ interpolation matrix B is invertible (Karabaş, 2020).

2.2.2. Derivative Approximations

By using the RBF expansion

$$u(x) = \sum_{i=1}^{N} \tilde{\Phi}(\|x - x_i\|_2; \tilde{\varepsilon})\lambda_j \tag{2.8}$$

the derivative approximations of the function u(x) can be written as

$$\frac{\partial}{\partial x_i} u(x) = \sum_{j=1}^{N} \lambda_j \frac{\partial}{\partial x_i} \tilde{\Phi}(\|x - x_j^c\|; \tilde{\varepsilon}). \tag{2.9}$$

In a similar way, higher order derivatives can be assessed.
If we write (2.9) in vector-matrix notation and evaluate it at the centers $\{x_j\}_{j=1}^{N}$, we get

$$\frac{\partial}{\partial x_i} u(x) = \frac{\partial}{\partial x_i} H\lambda. \tag{2.10}$$

where the evaluation matrix is the $N \times N$ matrix $\frac{\partial}{\partial x_i} H$ with entries

$$h_{ij} = \frac{\partial}{\partial x_i} \tilde{\Phi}(\|x_i - x_j\|_2) \quad i, j = 1, ..., N$$

The matrix of differentiation can be defined as

$$D = \frac{\partial}{\partial x_i} H B^{-1}. \tag{2.11}$$

by replacing $\lambda = B^{-1}u$ into (2.10).
The differentiation matrix is well-defined because system matrix B is invertible. The derivatives of MQ RBF derived using the chain rule are as follows:

$$\frac{\partial \tilde{\Phi}}{\partial x_i} = \frac{d\tilde{\Phi}}{dr}\frac{\partial r}{\partial x_i} \tag{2.12}$$

and

$$\frac{\partial^2 \tilde{\phi}}{\partial x_i^2} = \frac{d\tilde{\Phi}}{dr}\frac{\partial^2 r}{\partial x_i^2} + \frac{d^2\tilde{\Phi}}{dr^2}\left(\frac{\partial r}{\partial x_i}\right)^2 \tag{2.13}$$

where

$$\frac{\partial r}{\partial x_i} = \frac{x_i}{r}, \frac{d\tilde{\Phi}}{dr} = \frac{\tilde{\varepsilon}^2 r}{\sqrt{1+\tilde{\varepsilon}^2 r^2}}, \frac{d^2\tilde{\Phi}}{dr^2} = \frac{\tilde{\varepsilon}^2}{\left(1+\tilde{\varepsilon}^2 r^2\right)^{3/2}}. \tag{2.14}$$

2.3. Finite Difference Method

The Finite Difference Method (FDM) was initially proposed in the 1920s by A.Thom (Thom and Apelt, 1961) to solve nonlinear hydrodynamic equations under the name "the method of squares." The approach has since been used to solve a variety of issues in several fields. Approximations are used in finite difference techniques to allow differential equations to be replaced by finite difference equations. These algebraic finite difference approximations connect the value of the dependent variable point in the solution zone to the values at some nearby locations.

Since we need to approximate the second order derivative, we derive a scheme using the Taylor's series expansion. Expanding $\tilde{f}\left(x \pm \Delta x\right)$ around $x = x_0$

$$\tilde{f}\left(x_o + \Delta x\right) = \tilde{f}\left(x_o\right) + \Delta x \tilde{f}'\left(x_o\right) + \frac{1}{2!}(\Delta x)^2 \tilde{f}''\left(x_o\right) + \frac{1}{3!}(\Delta x)^3 \tilde{f}'''\left(x_o\right) + \cdots \tag{2.15}$$

and

$$\tilde{f}\left(x_o - \Delta x\right) = \tilde{f}\left(x_o\right) - \Delta x \tilde{f}'\left(x_o\right) + \frac{1}{2!}(\Delta x)^2 \tilde{f}''\left(x_o\right) - \frac{1}{3!}(\Delta x)^3 \tilde{f}'''\left(x_o\right) + \cdots \tag{2.16}$$

Upon adding these expansions,

$$\tilde{f}\,(x_o + \Delta x) + \tilde{f}\,(x_o - \Delta x) = 2\tilde{f}\,(x_o) + (\Delta x)^2 \tilde{f}''\,(x_o) + O(\Delta x)^4 \tag{2.17}$$

where $O(\Delta x)^4$ is the error introduced by truncating the series. This is referred to as an error of order $O(\Delta x)^4$ or just $O(\Delta x)^4$. As a result, $O(\Delta x)^4$ stands for phrases that are not greater than $O(\Delta x)^4$. Assuming these terms are negligible,

$$\tilde{f}''\,(x_o) \simeq \frac{\tilde{f}\,(x_o + \Delta x) - 2\tilde{f}\,(x_o) + \tilde{f}\,(x_o - \Delta x)}{(\Delta x)^2} \tag{2.18}$$

CHAPTER 3

TIME DISCRETIZATION

3.1. Runge Kutta Methods

Carl Runge and Wilhelm Kutta, two German mathematicians, created Runge Kutta methods in 1900. Each Runge Kutta method is developed from a Taylor method of appropriate order. Several function evaluations are performed at each step in exchange for not having to compute the higher derivatives. These methods can all be built for any Nth order. The Runge Kutta method, for order $N = 4$, is the most often used. It's a great option for most applications since it's accurate, reliable, and simple to program. The majority of authorities argue that using a higher-order approach is unnecessary since the greater accuracy is negated by the additional processing work (Mathews and Fink, 2004). The method works by computing y_{k+1} as follows:

$$y_{k+1} = y_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4), \tag{3.1}$$

where

$$f_1 = f(t_k, y_k), \tag{3.2}$$

$$f_2 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2}f_1), \tag{3.3}$$

$$f_3 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2}f_2), \tag{3.4}$$

$$f_4 = f(t_k + h, y_k + hf_3). \tag{3.5}$$

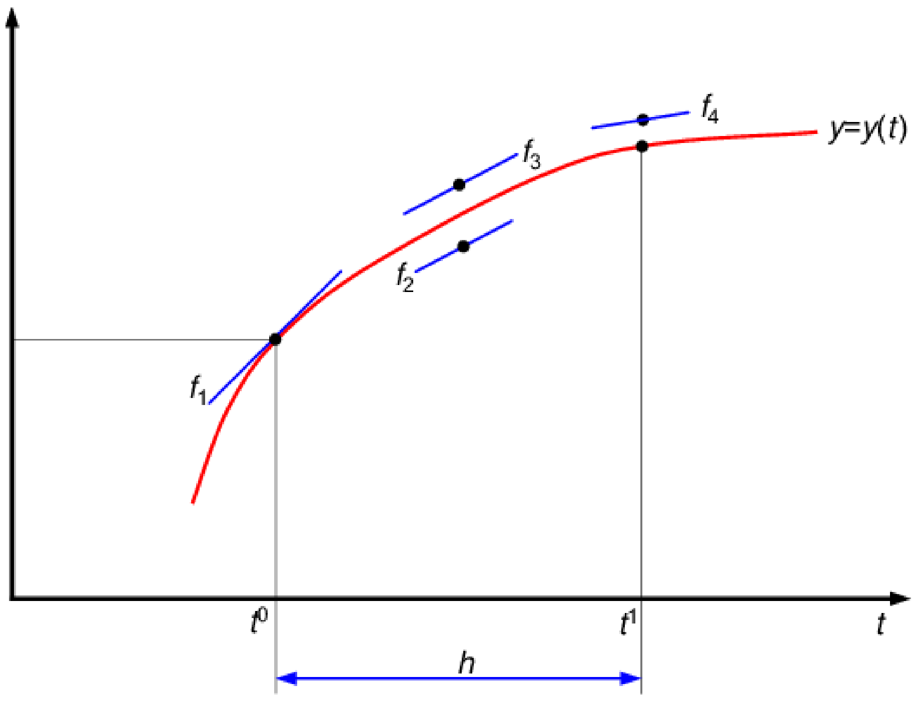

Figure 3.1. Geometric interpretation of the fourth order Runge Kutta Method

3.2. Adaptive Runge Kutta Method

A class of adaptive method for solving differential equations is the so-called embedded Runge-Kutta methods. An embedded Runge-Kutta method uses two ordinary Runge-Kutta methods for comparing the numerical solutions and selecting the step size. Moreover, the two methods in an embedded method typically share the evaluation of f (we are solving $\dot{y} = f(t, y)$). Therefore, the required computation effort is minimized.

The method works by computing y^*_{n+1}as follows:

$$\begin{aligned}
&k_1 = hf\left(x_n, y_n\right),\\
&k_2 = hf\left(x_n + a_2 h, y_n + b_{21}k_1\right),\\
&\dots\\
&k_6 = hf\left(x_n + a_6 h, y_n + b_{61}k_1 + \cdots + b_{65}k_5\right),\\
&y_{n+1} = y_n + c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4\\
&+ c_5k_5 + c_6k_6 + O\left(h^6\right).
\end{aligned} \tag{3.6}$$

The embedded fourth-order formula is

$$\begin{aligned}
y^*_{n+1} =& y_n + c^*_1k_1 + c^*_2k_2 + c^*_3k_3 + c^*_4k_4\\
&+ c^*_5k_5 + c^*_6k_6 + O\left(h^5\right)
\end{aligned} \tag{3.7}$$

Although we are not going to use y^*_{n+1} as the numerical solution at t_{n+1}, we can still use y^*_{n+1} to compare with the 5th order solution y_{n+1}. If their difference is too large, we reject the solution and use a smaller stepsize h to repeat the calculation. If their difference is small enough, we will accept y_{n+1}. But we also use this information to suggest a step size for the next step. A user must specify a small number ε (called the error tolerance) to control the error for selecting the step size. The difference between y_{n+1} and y^*_{n+1} is

$$e \equiv y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} \left(c_i - c^*_i\right) k_i \tag{3.8}$$

Based on the user specified error tolerance ε, the following relation update the step-size, h.

$$h := 0.9h\left(\frac{\varepsilon}{e}\right)^{1/5} \tag{3.9}$$

Table 3.1. Coefficients for RK5 (4)

a_i	b_{ij}						c_i	c_i^*
0							$\frac{35}{384}$	$\frac{5179}{57600}$
$\frac{1}{5}$	$\frac{1}{5}$						0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					$\frac{500}{1113}$	$\frac{7571}{16695}$
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				$\frac{125}{192}$	$\frac{393}{640}$
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			$-\frac{2187}{6784}$	$-\frac{92097}{339200}$
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		$\frac{11}{84}$	$\frac{187}{2100}$
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0	$\frac{1}{40}$

CHAPTER 4

NUMERICAL SOLUTIONS OF THE BRUSSELATOR MODEL

In this chapter, we solve the 1D and 2D Brusselator Model with the combinations of the FDM with RK4 and ARK and the Meshless Method with RK4 and ARK. This chapter includes mainly two parts which are tables and graphs. Last, all simulations will be produced by executing the Matlab programming language issues.

Test Problem 4.1 Consider the 1D Brusselator Model

$$\frac{\partial u}{\partial t} = \tilde{\alpha} - (\tilde{\beta} + 1)u + u^2 v + k\left(\frac{\partial^2 u}{\partial x^2}\right)$$

$$\frac{\partial v}{\partial t} = \tilde{\beta} u - u^2 v + k\left(\frac{\partial^2 v}{\partial x^2}\right)$$

with the following initial data (Zegeling and Kok, 2004), (Alqahtani, 2018), (Mittal and Rohila, 2016)

$$u(x, 0) = 0.5$$

$$v(x, 0) = 1 + 5x$$

and Neumann boundary conditions

$$u_x(0,t) = u_x(1,t) = 0,$$

$$v_x(0,t) = v_x(1,t) = 0.$$

By considering different parameters, the model's varied types of patterns are illustrated in Figures 1, 2, 3, and 4. Figures 1, 2 and 3 show the concentration patterns $u(0.3,t)$, $v(0.3,t)$ that converge to the fixed point,$(\tilde{\alpha},\ \frac{\tilde{\alpha}}{\tilde{\beta}}) = (1, 0.5)$ while Figure 4 presents oscillatory behavior of $u(0.3)$, $v(0.3)$ for parameters $\tilde{\alpha} = 1, \tilde{\beta} = 3.4$, and $k = 0.0001$.

Method of Lines: The semidiscrete form the Brusselator Model can be written as

$$\dot{\mathbf{U}} = A\mathbf{U} + F(\mathbf{U})$$

where $\mathbf{U} = \begin{bmatrix} U \\ V \end{bmatrix}$, $A = \begin{bmatrix} k\bar{D} & 0 \\ 0 & k\bar{D} \end{bmatrix}$ and $F(\mathbf{U}) = \begin{bmatrix} \tilde{\alpha} - (\tilde{\beta}+1)U + U^2V \\ \tilde{\beta}U - U^2V \end{bmatrix}$ where $U = [U_1 \quad U_2 \quad ... \quad U_N]^T$ and $V = [V_1 \quad V_2 \quad ... \quad V_N]^T$.

When we using the RBF expansion (2.8) the second derivative approximations of the function u(x) can be written as

$$\frac{\partial^2}{\partial x_i^2}u(x) = \sum_{j=1}^{N} \lambda_j \frac{\partial^2}{\partial x_i^2}\tilde{\Phi}(\|x - x_j^c\|; \tilde{\varepsilon}). \tag{4.1}$$

If we write (4.1) in vector-matrix notation and evaluate it at the centers $\{x_j\}_{j=1}^N$, we get

$$\frac{\partial^2}{\partial x_i^2}u(x) = \frac{\partial^2}{\partial x_i^2}\bar{H}\lambda. \tag{4.2}$$

By replacing $\lambda = B^{-1}u$ into (4.2) the matrix of differentiation for second derivative can be defined as

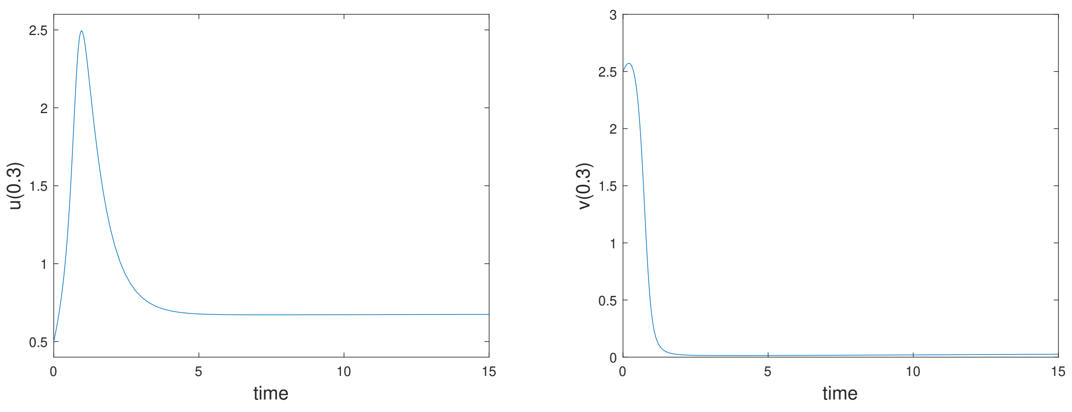

Figure 4.1. The solution profile using the FDM & ARK of $u(0.3)$, and $v(0.3)$ $0 \leq t \leq 15$, $\tilde{\alpha} = 1$, $\tilde{\beta} = 0.5$, $k = 0.0001$ and $\varepsilon = 10^{-6}$

$$\bar{D} = \frac{\partial^2}{\partial x_i^2} \bar{H} B^{-1}. \tag{4.3}$$

The matrix $\frac{\partial^2}{\partial x_i^2} \bar{H}$ has the entries

$$\bar{h_{ij}} = \frac{\partial^2}{\partial x_i^2} \tilde{\Phi}(\|x_i - x_j\|_2) \quad i, j = 1, ..., N$$

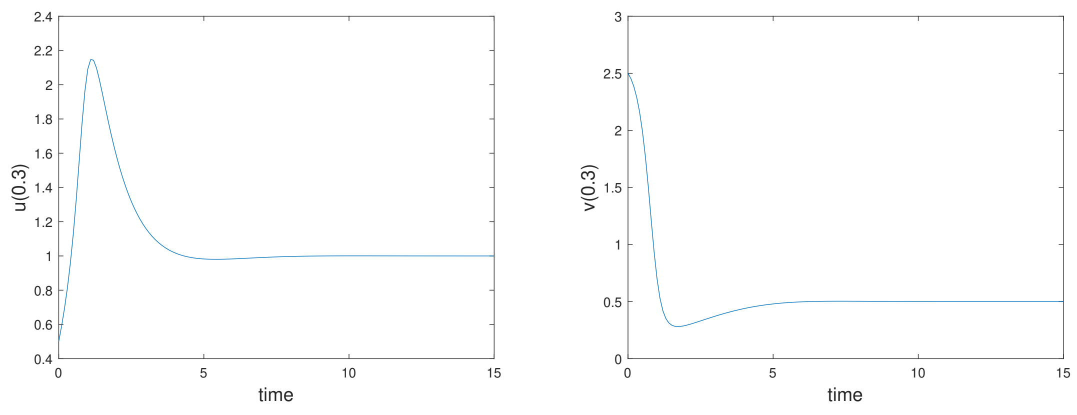

Figure 4.2. The solution profile using the FDM & RK of $u(0.3)$, and $v(0.3)$ $0 \leq t \leq 15$, $\tilde{\alpha} = 1$, $\tilde{\beta} = 0.5$, $k = 0.0001$, $dx = 0.1$ and $\Delta t = 0.03$

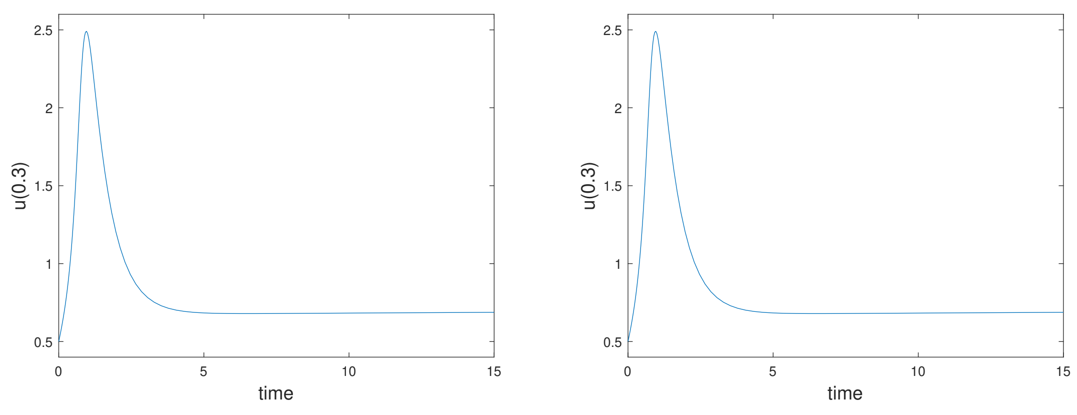

Figure 4.3. The solution profile using the Meshless & ARK of $u(0.3)$, and $v(0.3)$ $0 \leq t \leq 15, \tilde{\alpha} = 1, \tilde{\beta} = 0.5, k = 0.0001$ and $\varepsilon = 10^{-6}$

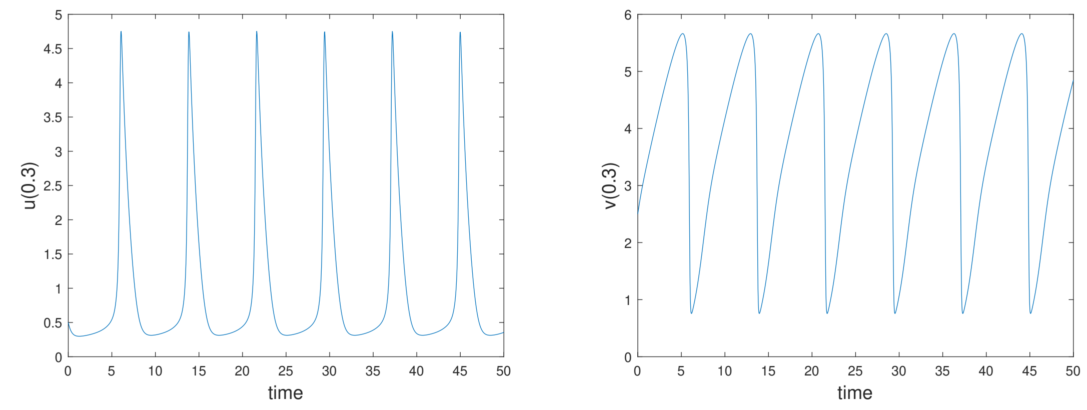

Figure 4.4. The solution profile using the FDM & RK of $u(0.3)$, and $v(0.3)$ $0 \leq t \leq 50$, $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$ and $k = 0.0001$.

Test Problem 4.2 We consider the 2D Brusselator Model with the initial and Neumann boundary conditions given as follows (Mittal and Jiwari, 2011), (Jiwari and Yuan, 2014), (Alqahtani, 2018), (Arshed et al., 2010), (Ang, 2003)

$$u(x, y, 0) = 0.5 + y, \quad (x, y) \in [0, 1] \times [0, 1]$$

$$v(x, y, 0) = 1 + 5x, \quad (x, y) \in [0, 1] \times [0, 1]$$

$$\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0,$$

With parameters $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ and $\Delta t = 0.01$ at various times $t = 0.5, 1.5, 3, 5, 7, 8.5, 10$, it is discovered that these patterns are remarkably similar to those presented in (Mittal and Jiwari, 2011), (Jiwari and Yuan, 2014), (Alqahtani, 2018), (Arshed et al., 2010), (Ang, 2003). A distinct concentration pattern v(x, y, t) in 3D and contour form of the issue is displayed in Figures $4.5 - 4.11$.

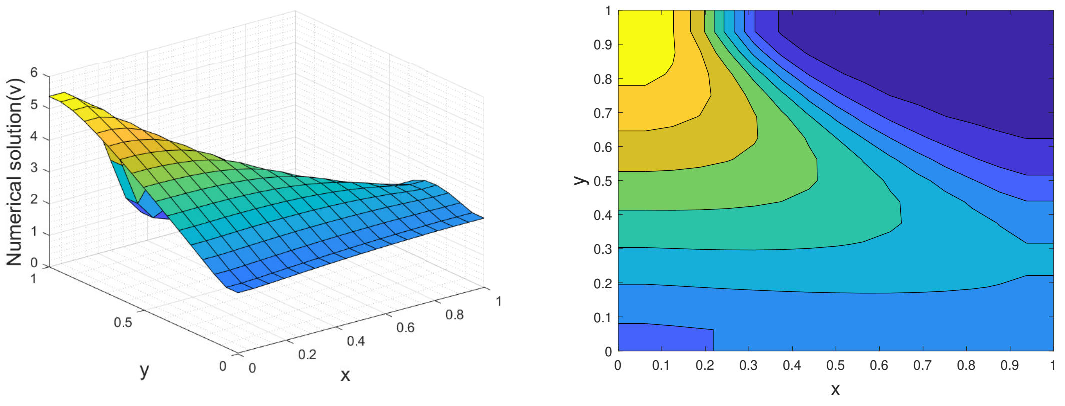

Figure 4.5. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$, $\Delta t = 0.01$, 15×15 and t = 0.5

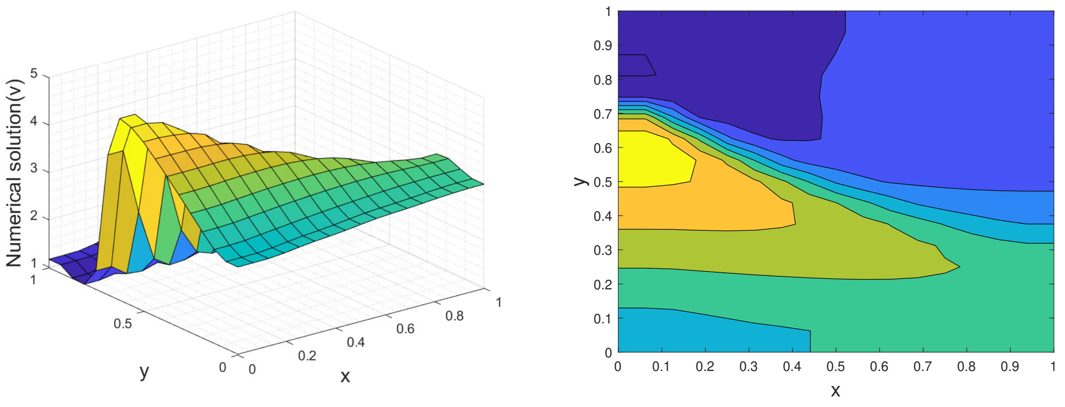

Figure 4.6. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =1.5

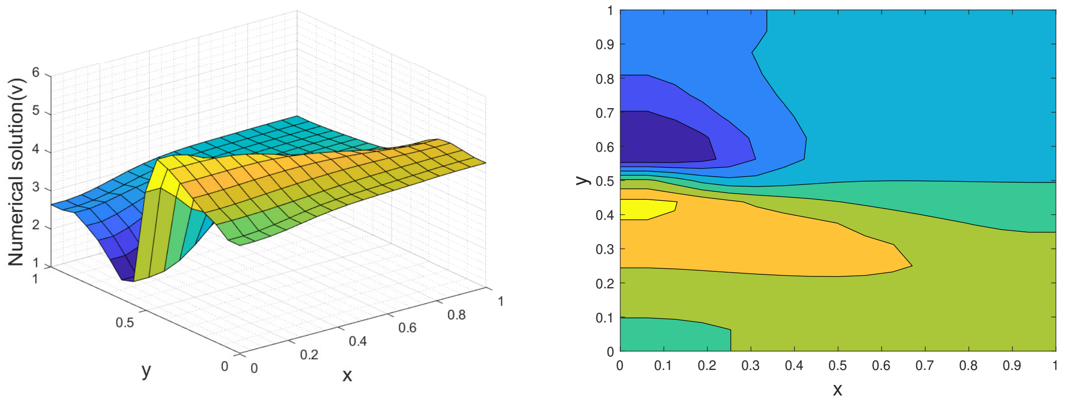

Figure 4.7. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =3

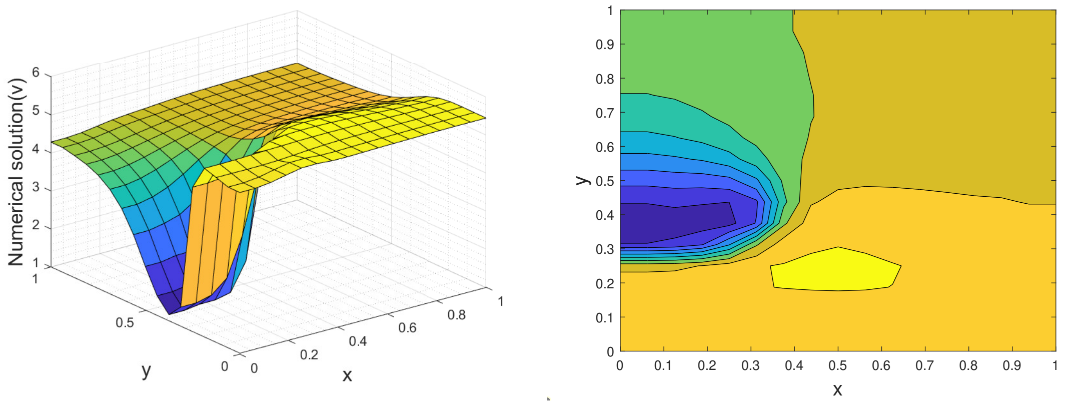

Figure 4.8. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =5

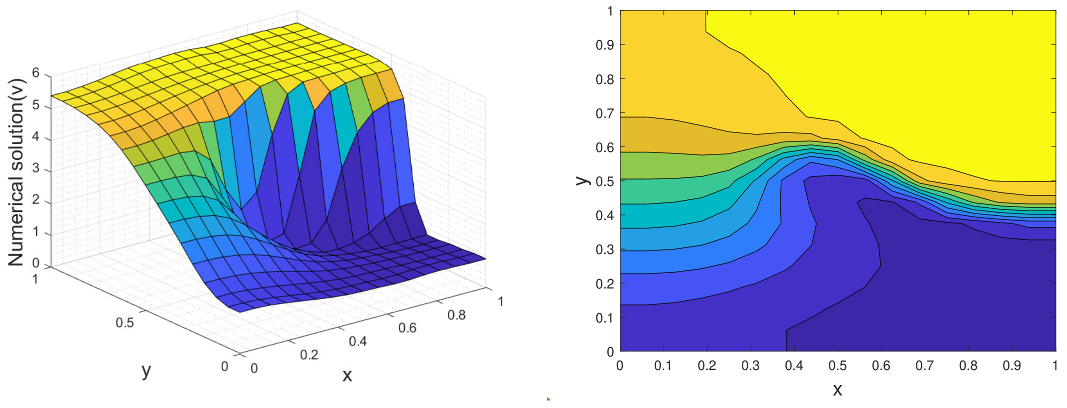

Figure 4.9. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =7

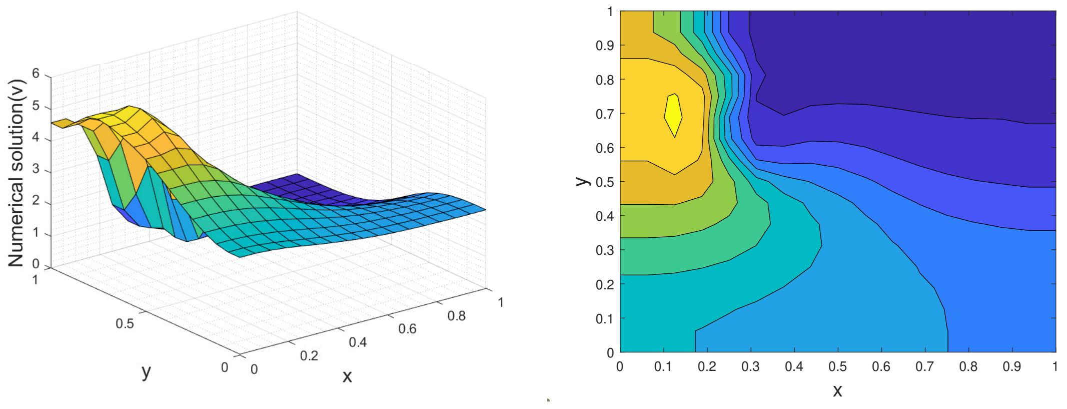

Figure 4.10. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =8.5

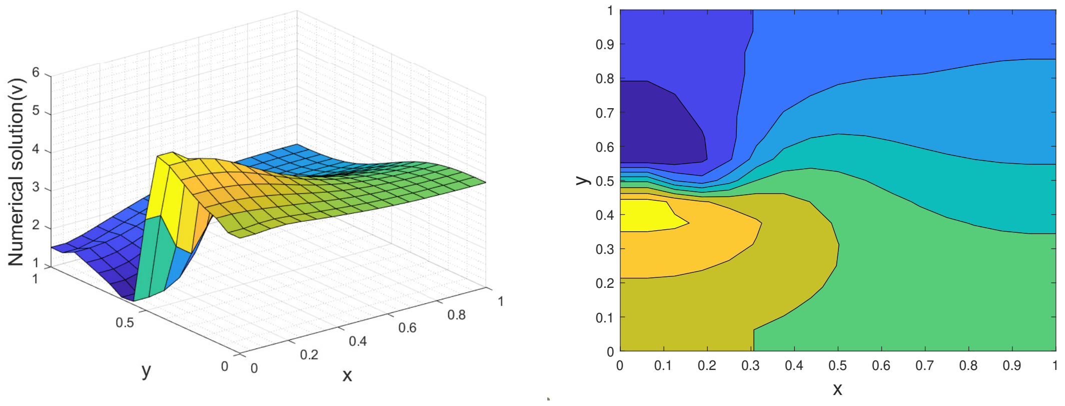

Figure 4.11. Physical phenomena of v concentration in 3D and contour plot at $\tilde{\alpha} = 1$, $\tilde{\beta} = 3.4$, $k = 0.002$ $\Delta t = 0.01$, 15×15 and t =10

Test Problem 4.3 In this problem, we consider the 2D Brusselator Model over the domain $\Omega = \{(x, y) : 0 \leq x \leq 1 \, , 0 \leq y \leq 1\}$ with the exact solution given as follows (Arshed et al., 2010), (Ang, 2003)

$$u(x, y, t) = exp(-x - y - 0.5t)$$

$$v(x, y, t) = exp(x + y + 0.5t)$$

The exact solution contributes to the derivation of the initial and boundary conditions. The numerical solution for this problem is calculated by combinations of FDM with RK4 and ARK and Meshless with RK4 and ARK. Tables 4.1 and 4.2 show the problem's maximum absolute error for the parameters $\tilde{\alpha} = 0$, $\tilde{\beta} = 1$, and $k = 0.25$ at $t = 2$ and time step $\Delta t = 0.002$. Finally, it is determined that FDM combinations perform similar as the Meshless method's combinations.

Table 4.1. L_∞ norms at $\Delta t = 0.002$

N	FDM & RK4		Meshless & RK4	
	u	v	u	v
10×10	1.2000×10^{-03}	6.0400×10^{-02}	2.3000×10^{-03}	1.2820×10^{-01}
15×15	1.8000×10^{-03}	9.2300×10^{-02}	3.5000×10^{-03}	1.9170×10^{-01}
19×19	2.3000×10^{-03}	1.1770×10^{-01}	4.4000×10^{-03}	2.4250×10^{-01}

Table 4.2. L_∞ norms at $\varepsilon = 10^{-6}$

N	FDM & ARK		Meshless & ARK	
	u	v	u	v
10×10	1.7114×10^{-04}	6.3000×10^{-03}	3.51924×10^{-04}	1.35361×10^{-02}
15×15	1.1179×10^{-04}	4.1000×10^{-03}	3.83133×10^{-04}	1.33072×10^{-03}
19×19	8.5874×10^{-05}	3.1000×10^{-03}	4.95716×10^{-04}	1.72319×10^{-03}

CHAPTER 5

SUMMARY AND CONCLUSION

In both science and engineering, nonlinear problems play a major role. Hence, the solutions to nonlinear partial differential equations are significant.

The procedure of the proposed method consists of the FDM and Multiquadric Radial Basis Function (MQ RBF) Method. The Multiquadric Radial Basis Function (MQ RBF) Method and the Finite Difference Method are the methods we employ for space discretization, and they are both discussed in Chapter 2. These methods concepts are explained in brief. The existence and uniqueness of the interpolation matrix are intuitively shown for the MQ RBF method. The use of MQ RBF to approximate derivatives is explained. For the Finite Difference Method, it is demonstrated how to take the derivatives of the first and second derivatives. Time discretization was first presented in Chapter 3. We utilized the Runge-Kutta Method and the Adaptive Runge-Kutta Method for this discretization. They observed while we computed these techniques for them.

In this thesis, we presented a comparative study of the FDM and Meshless Method numerical results for solving the 1D and 2D Brusselator Model. From test problem 2, when t's are getting bigger patterns are consistent. For both problems under consideration, the numerical results exhibited that similar results. We obtain from the last problem, the order of the Meshless Method by comparing the second order FDM in space.

REFERENCES

Adomian, G., 1995: The diffusion-Brusselator equation, *Computers & mathematics with applications*, 29, 1-3.

Alipanah, A., 2016: Multiquadric Radial Basis Function Method for the Solution of Brachistochrone Problem, *Rom. J. Math. Comput. Sci*, 6, 126-133.

Alqahtani, A., 2018: Numerical simulation to study the pattern formation of reaction–diffusion Brusselator model arising in triple collision and enzymatic, *Journal of Mathematical Chemistry*, 56, 1543-1566.

Ang, W. 2003: The two-dimensional reaction–diffusion Brusselator system: a dual-reciprocity boundary element solution, *Engineering Analysis with Boundary Elements*, 27, 897-903.

Arshed et al., 2010: A computational modeling of the behavior of the two-dimensional reaction–diffusion Brusselator system, *Applied Mathematical Modelling*, 34, 3896-3909.

Bhatt, H. and Chowdhury, A., 2019: Comparative analysis of numerical methods for the multidimensional Brusselator system, *Open J. Math. Sci*, 3, 262-272.

Dehghan, M. and Abbaszadeh, M., 2016: Variational multiscale element free Galerkin (VMEFG) and local discontinuous Galerkin (LDG) methods for solving two-dimensional Brusselator reaction–diffusion system with and without cross-diffusion, *Computer Methods in Applied Mechanics and Engineering*, 300, 770-797.

Fasshauer, G.E., 1996: Solving partial differential equations by collocation with radial basis functions, *Proceedings of chamonix*, 1997, 1-8.

Hardy, R.L., 1971:Multiquadric equations of topography and other irregular surfaces, *Journal of geophysical research*, 76, 1905-1915.

Hardy, R.L., 1990: Theory and applications of the multiquadric-biharmonic method, *Computers & Mathematics with Applications*, 19, 163-208

Hilal, E. and Elzaki, T.M., 2014: Solution of nonlinear partial differential equations by new Laplace variational iteration method, *Journal of Function Spaces*, 2014.

Jiwari, R. and Yuan, J., 2014: A computational modeling of two dimensional reaction–diffusion Brusselator system arising in chemical processes, *Journal of mathematical Chemistry*, 52, 1535-1551.

Kansa, E. J. 1990: Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates, *Computers and Mathematics with Applications*, 19, 127-145.

Kansa, E. J. 1990: 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 Applications*, 19, 147-161.

Karabaş, N., 2020: Analysis and Application of a Linearization Tecnique for Nonlinear Problems, Ph. D. thesis, *İzmir Institute of Technology*.

Kleefeld et al., 2012: An ETD Crank-Nicolson method for reaction-diffusion systems, *Numerical Methods for Partial Differential Equations*, 28, 1309-1335.

Larsson, E. and Fornberg, B., 2003: A numerical study of some radial basis function based solution methods for elliptic PDEs, *Computers & Mathematics with Applications*, 46, 981-902.

Lefever, R., and Nicolis, G., 1971: Chemical instabilities and sustained oscillations, *Journal of theoretical Biology* , 30, 267-284

Mathews, J.H. and Fink, K.D., 2004: Numerical methods using MATLAB, *Pearson prentice hall Upper Saddle River, NJ*4, 489-491.

Micchelli, C.A., 1984: Interpolation of scattered data: distance matrices and conditionally positive definite functions, *Approximation theory and spline functions*, 143-145.

Mittal, R and Jiwari, R, 2011: Numerical solution of two-dimensional reaction–diffusion Brusselator system, *Applied mathematics and computation*, 2017, 5404-5415.

Mittal, R.C. and Rohila, R., 2016: Numerical simulation of reaction-diffusion systems by modified cubic B-spline differential quadrature method, *Chaos, Solitons & Fractals*,92, 9-19.

Mittal, R.C. et al., 2019: Numerical simulation for computational modelling of reaction–diffusion Brusselator model arising in chemical processes, *Journal of Mathematical Chemistry*, 57, 149-179.

Press, W.H. and Teukolsky, S.A., 1992: Adaptive stepsize Runge-Kutta integration, *Computers in Physics*, 6, 188-191.

Prigogine, I. and Nicolis, G, 1985: Self-organisation in nonequilibrium systems: towards a dynamics of complexity, *Springer*, 3-12

Prigogine, I. and Lefever, R., 1968: Symmetry breaking instabilities in dissipative systems. II, *The Journal of Chemical Physics*, 48, 1695-1700.

Thom A. and Apelt C.J., 1961: Field Computations in Engineering and Physics

Twizell et al., 1999: A second-order scheme for the âBrusselatorâ reaction-diffusion system, *Journal of Mathematical Chemistry*,266, 297-316.

Tyson J, 1973: Some further studies of nonlinear oscillations in chemical systems, *The Journal of Chemical Physics*, 58, 3919–3930.

Verwer et al., 1990: Convergence properties of the Runge-Kutta-Chebyshev method, *Numerische Mathematik*, 57, 157-178.

Wazwaz, A., 2000: The decomposition method applied to systems of partial differential equations and to the reaction–diffusion Brusselator model, *Applied mathematics and computation*, 110, 251-264.

Yu, P and Gumel, A.B., 2001: Bifurcation and stability analyses for a coupled Brusselator model, *Journal of Sound and vibration*,244, 795-820.

Zegeling, P.A. and Kok, H.P., 2004: Adaptive moving mesh computations for reaction–diffusion systems, *Journal of Computational and Applied Mathematics*, 168, 519-528.

APPENDIX A

MATLAB CODES FOR NUMERICAL EXPERIMENTS

A.1. Codes for Finite Difference Method & Runge Kutta in 1D for Test Problem 4.1

```
clear
clc
tic
%Time interval, space interval
global nx nt dx dt
xmin = 0; xmax = 1;
nx = 10;  dx = (xmax-xmin)/nx;
nt = 150*nx;
dt = 50/nt;

U = zeros(nx+1,nt+1);
V = zeros(nx+1,nt+1);
x = (xmin:dx:xmax);
f = @(x) 0.5+0*x;
g = @(x) 1+5*x;
U(:,1) = f(x);
V(:,1) = g(x);
for j=1:nt

%% boundary conditions
  U(1,j) = U(2,j); U(end,j) = U(end-1,j);
  V(1,j) = V(2,j); V(end,j) = V(end-1,j);
  [F1,G1] = Fun(dx,nx+1,U(:,j),V(:,j));
  [F2,G2] = Fun(dx,nx+1,U(:,j) +...
  (dt/2)*F1,V(:,j) + (dt/2)*G1);
  [F3,G3] = Fun(dx,nx+1,U(:,j) +...
  (dt/2)*F2,V(:,j) + (dt/2)*G2);
  [F4,G4] = Fun(dx,nx+1,U(:,j) +...
dt*F3,V(:,j) + dt*G3);
  U(:,j+1) = U(:,j) + (dt/6)*(F1+2*(F2+F3)+F4);
  V(:,j+1) = V(:,j) + (dt/6)*(G1+2*(G2+G3)+G4);
end
toc

T=0:dt:50;
% subplot(1,2,1)
figure(1)
plot(T,U(4,:))

     xlabel('time', 'Fontsize', 14)
ylabel('u(0.3)', 'Fontsize', 14)
% zlabel('U')
axis([0 50 0 5])
% subplot(1,2,2)
figure(2)
plot(T,V(4,:))
%  surf(X,T,V)
xlabel('time', 'Fontsize', 14)
ylabel('v(0.3)', 'Fontsize', 14)
% zlabel('v')
axis([0 50 0 6])

function [F,G] = Fun(dx,n,u,v)
k = 0.0001;
alpha = 1;
beta = 3.4;
F = k*FDM2D2(dx,n,u) + alpha-(beta+1)*u+u.^2.*v;
```

```
G = k*FDM2D2(dx,n,v) + beta*u-u.^2.*v;
end
```

A.2. Codes for Finite Difference Method & Adaptive Runge Kutta in 1D for Test Problem 4.1

```
clear
clc
%Time interval, space interval
t0 = 0; t1 = 15;
xmin = 0; xmax = 1; dx = 0.1;
x = (xmin:dx:xmax)';
n = length(x);
%% initial conditions
f = @(x) 0.5+0*x;  g = @(x) 1+5*x;
U(:,1) = f(x);     V(:,1) = g(x);

y0 = [U V];
tol=1e-6;

t=t0; tout = 0;
y=y0;
h=tol^(1/5)/4;
step=0;
nrej=0;
fcall=1;
a4=[44/45 -56/15 32/9]';
a5=[19372/6561 -25360/2187 64448/6561 -212/729]';
a6=[9017/3168 -355/33 46732/5247 49/176 -5103/18656]';
a7=[35/384 0 500/1113 125/192 -2187/6784 11/84]';
e=[71/57600 -1/40 -71/16695 71/1920
-17253/339200 22/525]';
k1=func(dx, n, y);
```

```
while t < t1
if t+h > t1; h=t1-t; end
k2=func(dx, n, y+h*k1/5);
k3=func(dx, n, y+h*(3*k1+9*k2)/40);
k4=func(dx, n, y+h*(a4(1)*k1+a4(2)*k2+a4(3)*k3));
k5=func(dx, n, y+h*(a5(1)*k1+a5(2)*k2+a5(3)*...
k3+a5(4)*k4));
k6=func(dx, n, y+h*(a6(1)*k1+a6(2)*k2+a6(3),...
*k3+a6(4)*k4+a6(5)*k5));
yt=y+h*(a7(1)*k1+a7(3)*k3+a7(4)*k4+a7(5)*k5+a7(6)*k6);
U(:,step+2) = yt(:,1);
V(:,step+2) = yt(:,2);
k2=func(dx, n, yt);
est=norm(h*(e(1)*k1+e(2)*k2+e(3)*k3+e(4)*...
k4+e(5)*k5+e(6)*k6),inf);
fcall=fcall+6;
% [t h est]

if est < tol
t=t+h;
tout = [tout t];
k1=k2;
step=step+1;
y=yt;
else
nrej=nrej+1;
end
h=.9*min((tol/(est+eps))^(1/5),10)*h;
end
T = tout';
figure(1)
```

```
plot(T,U(4,:))
axis([0 15 0.4 2.6])
xlabel('time', 'Fontsize', 14)
ylabel('u(0.3)', 'Fontsize', 14)
figure(2)
plot(T,V(4,:))
axis([0 15 0 3])
xlabel('time', 'Fontsize', 14)
ylabel('v(0.3)', 'Fontsize', 14)
% figure(3)
% plot(T,U(3,:))
% axis([0 15 0.4 2.4])
% figure(4)
% plot(T,U(4,:))
% axis([0 15 0.4 2.4])
% figure(5)
```

A.3. Codes for Meshless & Adaptive Runge Kutta in 1D for Test Problem 4.1

```
clear
clc
%Time interval, space interval
t0 = 0; t1 = 15;
xmin = 0; xmax = 1; dx = 0.1;
x = (xmin:dx:xmax)'; n = length(x);
%% Matrix for derivatives
A = zeros(n);
c=0.4;
%beta1=3;
for j=1:n
  for i=1:n
      A(j,i)=sqrt((x(j)-x(i))^2+c^2);
  end
```

```
end
for j=1:n
  for i=1:n
      B(j,i)=(x(j)-x(i))/A(j,i);
      C(j,i)=c^2/A(j,i)^(3/2);
  end
end
I=eye(n);
invA=A\I;
% a=B*invA;
b1=C*invA; %u_xx

% U0 = zeros(n,86);  V0 = zeros(n,86);

%% initial conditions
f = @(x) 0.5+0*x;  g = @(x) 1+5*x;
U(:,1) = f(x);     V(:,1) = g(x);

y0 = [U V];
tol=1e-6;

t=t0; tout = 0;
y=y0;
h=tol^(1/5)/4;
step=0;
nrej=0;
fcall=1;
a4=[44/45 -56/15 32/9]';
a5=[19372/6561 -25360/2187 64448/6561 -212/729]';
a6=[9017/3168 -355/33 46732/5247 49/176 -5103/18656]';
a7=[35/384 0 500/1113 125/192 -2187/6784 11/84]';
e=[71/57600 -1/40 -71/16695 71/1920
-17253/339200 22/525]’;
k1=func(dx, b1,y);

while t < t1
if t+h > t1; h=t1-t; end
k2=func(dx, b1, y+h*k1/5);
k3=func(dx, b1, y+h*(3*k1+9*k2)/40);
k4=func(dx, b1, y+h*(a4(1)*k1+a4(2)*k2+a4(3)*k3));
k5=func(dx, b1, y+h*(a5(1)*k1+a5(2)*...
k2+a5(3)*k3+a5(4)*k4));
k6=func(dx, b1, y+h*(a6(1)*k1+a6(2)*k2+a6(3)*...
k3+a6(4)*k4+a6(5)*k5));
yt=y+h*(a7(1)*k1+a7(3)*k3+a7(4)*k4+a7(5)*k5+a7(6)*k6);
U(:,step+2) = yt(:,1);

V(:,step+2) = yt(:,2);

k2=func(dx, b1, yt);
est=norm(h*(e(1)*k1+e(2)*k2+e(3)*k3+e(4)*k4+e(5)*k5+...
e(6)*k6),inf);
fcall=fcall+6;
% [t h est]

if est < tol
t=t+h;
tout = [tout t];
k1=k2;
step=step+1;
y=yt;
else
nrej=nrej+1;
end
```

```
h=.9*min((tol/(est+eps))^(1/5),10)*h;

end
T = tout';
figure(1)
plot(T,U(4,:))
axis([0 15 0.4 2.6])
xlabel('time', 'Fontsize', 14)
ylabel('u(0.3)', 'Fontsize', 14)
figure(2)
plot(T,V(4,:))
axis([0 15 0 3])
xlabel('time', 'Fontsize', 14)
ylabel('v(0.3)', 'Fontsize', 14)
% figure(3)
% plot(T,U(3,:))
% axis([0 15 0.4 2.4])
% figure(4)
% plot(T,U(4,:))
% axis([0 15 0.4 2.4])
% figure(5)
% plot(T,U(5,:))
% axis([0 15 0.4 2.4])
```

A.4. Codes for Meshless & Adaptive Runge Kutta in 2D for Test Problem 4.2

```
clc;
clear;
xs=0; xf=1; Nx=16; nx=Nx+1;
ys=0; yf=1; Ny=16; ny=Ny+1;
t0=0; tf=10;
x=linspace(xs,xf,nx)';
y=linspace(ys,yf,ny)';
```

```
[xm, ym]=meshgrid(x,y);
xx=xm(:); yy=ym(:);
alpha = 1;
beta = 3.4;
k = 0.002;
advx  = 0;
advy  = 0;

%% initial cond. implementation
f1 = @(x,y) 0.5 + y + 0*x;
f2 = @(x,y) 1 + 5*x + 0*y;

uu0 = f1(xm,ym); vv0 = f2(xm,ym);

% figure(1)
% subplot(1,2,1)
% surf(xm,ym,uu0)
% grid on
% grid minor
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{mesh}')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,uu0)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar

% figure(3)
% pcolor(xm,ym,uu0)
```

```
U0=reshape(uu0',nx*ny,1);
V0=reshape(vv0',nx*ny,1);

% c the value for multiquadric meshfree method
% tol: tolerance for adaptive rk
c=0.42;
[D2]= meshless_2D(xx,yy,nx,ny,c,1,1);%,1,1);
M2=[k*D2, zeros(size(D2)); zeros(size(D2)), k*D2];
%% numerical process
tol=1e-6;
ff=@(X) [alpha-(beta+1).*X(1:length(D2),1)+....
X(1:length(D2),1).^2.*(X(length(D2)+1:end))
;beta.*X(1:length(D2),1)-X(1:length(D2),1).^2.*;
(X(length(D2)+1:end))]
func=@(t,X) M2*X+ff(X);
Y=@(U,V) (vertcat(U,V));
t=t0;
h=tol^(1/5)/4;
step=0;
nrej=0;
fcall=1;
a4=[44/45 -56/15 32/9]';
a5=[19372/6561 -25360/2187 64448/6561 -212/729]';
a6=[9017/3168 -355/33 46732/5247 49/176 -5103/18656]';
a7=[35/384 0 500/1113 125/192 -2187/6784 11/84]';
e=[71/57600 -1/40 -71/16695 71/1920
-17253/339200 22/525]';
while t < tf
  u=Y(U0,V0);
  k1=func(t,u);
if t+h > tf; h=tf-t; end
u_prev=u;
```

```
k2=func(t+h/5,u+h*k1/5);
k3=func(t+3*h/10,u+h*(3*k1+9*k2)/40);
k4=func(t+4*h/5,u+h*(a4(1)*k1+a4(2)*k2+a4(3)*k3));
k5=func(t+8*h/9,u+h*(a5(1)*k1+a5(2)*k2+...
a5(3)*k3+a5(4)*k4));
k6=func(t+h,u+h*(a6(1)*k1+a6(2)*k2+a6(3)*k3+...
a6(4)*k4+a6(5)*k5));
u=u+h*(a7(1)*k1+a7(3)*k3+a7(4)*k4+a7(5)*k5+a7(6)*k6);
k2=func(t+h,u);
est=max(max((h*(e(1)*k1+e(2)*k2+e(3)*k3+e(4)*...
k4+e(5)*k5+e(6)*k6))));
fcall=fcall+6;
% rr=norm(u-u_prev,inf)
% [t h est]

if est < tol
t=t+h;
% k1=k2;
step=step+1;
ut=reshape(u(1:length(U0)),nx,ny)';
vt=reshape(u(length(U0)+1:end),nx,ny)';
ut(1,:) = ut(2,:);
ut(end,:) = ut(end-1,:);
ut(:,1) = ut(:,2);
ut(:,end) = ut(:,end-1);
vt(1,:) = vt(2,:);
vt(end,:) = vt(end-1,:);
vt(:,1) = vt(:,2);
vt(:,end) = vt(:,end-1);
U0 = reshape(ut',nx*ny,1);
V0 = reshape(vt',nx*ny,1);
surf(xm,ym,vt)
```

```
else
nrej=nrej+1;
end
h=.9*min((tol/(est+eps))^(1/5),10)*h;
% surf(xm,ym,vt)

end

% figure(1)
% subplot(1,2,1)
% surf(xm,ym,ut)
% hold on
% plot3(xm, ym, exact_u(xm,ym,tf),'*')
% grid on
% grid minor
% xlabel('x')
% ylabel('y')
% zlabel('u')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,ut)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar
% figure
% levels=[0.02:0.02:0.16];
% contour(xm,ym,c_analytical,levels,':');
% hold on
% [cont,h]=contour(xm,ym,ut,levels,'ShowText','On');
% clabel(cont,h,'FontSize',8)
% legend('boxoff')
% grid on
% grid minor
% xlabel('x (m)')
% ylabel('y (m)')
% yticks(0:0.2:2)
% xticks(0:0.2:2)
% colorbar
figure(1)
% subplot(1,2,1)
surf(ym,xm,vt)
hold on
% plot3(xm, ym, exact_v(xm,ym,tf),'*')
grid on
grid minor
xlabel('x','FontSize',15)
ylabel('y','FontSize',15)
zlabel('Numerical solution(v)','FontSize',15)
% colorbar
% subplot(1,2,2)
figure(2)
contourf(ym,xm,vt)
xlabel('x','FontSize',15)
ylabel('y','FontSize',15)
zlabel('c_{ana}')
% colorbar
```

A.5. Codes for Finite Difference Method & Runge Kutta in 2D for Test Problem 4.3

```
clear
clc
tic
%% Time interval, space interval
```

```
xmin = 0; xmax = 1; ymin = 0; ymax = 1; tmin = 0;
tmax = 2; global nx ny nt dx dy dt
nx=19; ny = 19; dt = 0.001; nt = (tmax-tmin)/dt;
dx=(xmax-xmin)/(nx-1); dy=(ymax-ymin)/(ny-1);
x = (xmin:dx:xmax); y = (ymin:dx:ymax);
t = (tmin:dt:tmax); [X, Y] = meshgrid(x,y);

%% initial condition
u = @(x,y,t) exp(-x-y-0.5*t);
v = @(x,y,t) exp(x+y+0.5*t);

U0 = u(X,Y,tmin);  V0 = v(X,Y,tmin);
U = U0;            V = V0;

t = tmin;
for n = 1:nt+1

%% boundary condition
  U(1,:) = u(xmin,y,t);     V(1,:) = v(xmin,y,t);
  U(:,1) = u(x,ymin,t);     V(:,1) = v(x,ymin,t);
  U(end,:) = u(xmax,y,t);   V(end,:) = v(xmax,y,t);
  U(:,end) = u(x,ymax,t);   V(:,end) = v(x,xmax,t);

  [F1,G1] = Fun(dx, nx, U, V);
  [F2,G2] = Fun(dx, nx, U+(dt/2)*F1, V+(dt/2)*G1);
  [F3,G3] = Fun(dx, nx, U+(dt/2)*F2, V+(dt/2)*G2);
  [F4,G4] = Fun(dx, nx, U+dt*F3, V+dt*G3);

  U = U + (dt/6)*(F1+2*F2+2*F3+F4);
  V = V + (dt/6)*(G1+2*G2+2*G3+G4);
  t = t + dt;
%figure(2)
```

```
%surf(X, Y, V); axis([xmin xmax ymin ymax 0 22])
%pause(0.001)
end
toc
error_u=norm(U-u(X,Y,tmax), inf)
error_v=norm(V-v(X,Y,tmax), inf)
% figure(1)
% surf(X, Y, U); axis([xmin xmax ymin ymax 0 0.4])
% figure(2)
% contour(X,Y,U)
% figure(3)
% surf(X, Y, V); axis([0 xmax 0 ymax 0 22])
% figure(4)
% contour(X,Y,V)

function [F, G] = Fun(dx, nx, U, V)
%% constant
k = 0.25; alpha = 0; beta = 1;
F1 = zeros(nx); F2 = zeros(nx);
G1 = zeros(nx); G2 = zeros(nx);
for n = 1:nx
  F1(n,:) = FDM2D2(dx,nx,U(n,:))';
  G1(n,:) = FDM2D2(dx,nx,V(n,:))';
end
for m = 1:nx
  F2(:,m) = FDM2D2(dx,nx,U(:,m));
  G2(:,m) = FDM2D2(dx,nx,V(:,m));
end
F = k*(F1 + F2) + alpha - (beta+1)*U + (U.^2).*V;
G = k*(G1 + G2) + beta*U - (U.^2).*V;
end
```

A.6. Codes for Finite Difference Method & Adaptive Runge Kutta in 2D for Test Problem 4.3

```
clear
tic
%% Time interval, space interval
xmin = 0; xmax = 1; ymin = 0; ymax = 1; t0 = 0;
tf = 2; global nx ny dx dy h
nx=19; ny = 19; dx=(xmax-xmin)/(nx-1);
dy=(ymax-ymin)/(ny-1);
x = (xmin:dx:xmax); y = (ymin:dx:ymax);
[Xn, Yn] = meshgrid(x,y);
%% exact solution
u = @(x,y,t) exp(-x-y-0.5*t);
v = @(x,y,t) exp(x+y+0.5*t);

U0 = u(Xn,Yn,t0);  V0 = v(Xn,Yn,t0);
U = U0;            V = V0;

tol=1e-6;
t=t0;
h=tol^(1/5)/4;
step=0;
nrej=0;
fcall=1;
a4=[44/45 -56/15 32/9]';
a5=[19372/6561 -25360/2187 64448/6561 -212/729]';
a6=[9017/3168 -355/33 46732/5247 49/176
-5103/18656]';
a7=[35/384 0 500/1113 125/192 -2187/6784 11/84]';
e=[71/57600 -1/40 -71/16695 71/1920
-17253/339200 22/525]';
while t < tf
```

```
[k1, g1]=func(dx, nx, U, V);
if t+h > tf; h=tf-t; end
  U(1,:) = u(xmin,y,t);     V(1,:) = v(xmin,y,t);
  U(:,1) = u(x,ymin,t);     V(:,1) = v(x,ymin,t);
  U(end,:) = u(xmax,y,t);   V(end,:) = v(xmax,y,t);
  U(:,end) = u(x,ymax,t);   V(:,end) = v(x,xmax,t);

[k2, g2]=func(dx, nx, U+h*k1/5, V+h*g1/5);
[k3, g3]=func(dx, nx, U+h*(3*k1+9*k2)/40,
V+h*(3*g1+9*g2)/40);
[k4, g4]=func(dx, nx,
U+h*(a4(1)*k1+a4(2)*k2+a4(3)*k3),...
...V+h*(a4(1)*g1+a4(2)*g2+a4(3)*g3));
[k5, g5]=func(dx, nx,U+h*(a5(1)*k1+...
a5(2)*k2+a5(3)*k3+a5(4)*k4),...
V+h*(a5(1)*g1+a5(2)*g2+a5(3)*g3+a5(4)*g4));
[k6, g6]=func(dx, nx, U+h*(a6(1)*k1+a6(2)*...
k2+a6(3)*k3+a6(4)*k4+a6(5)*k5)+V+h*(a6(1)*g1+...
+a6(2)*g2+a6(3)*g3+a6(4)*g4+a6(5)*g5));
Ut=U+h*(a7(1)*k1+a7(3)*k3+a7(4)*k4+a7(5)*k5+a7(6)*k6);
Vt=V+h*(a7(1)*g1+a7(3)*g3+a7(4)*g4+a7(5)*g5+a7(6)*g6);
[k2, g2]=func(dx, nx, U, V);
est=max(norm(h*(e(1)*k1+e(2)*k2+e(3)*k3+e(4)*...
k4+e(5)*k5+e(6)*k6),inf),...
norm(h*(e(1)*g1+e(2)*g2+e(3)*g3+e(4)*...
g4+e(5)*g5+e(6)*g6),inf));
fcall=fcall+6;
% rr=norm(u-u_prev,inf)
% [t h est]

if est < tol
t=t+h;
```

```
% k1=k2;
step=step+1;
U = Ut;
V = Vt;
else
nrej=nrej+1;
end
h=.9*min((tol/(est+eps))^(1/5),10)*h;
end
toc

error_u = norm(U - u(Xn,Yn,tf),inf)
error_v = norm(V - v(Xn,Yn,tf),inf)

% figure(1)
% surf(Xn, Yn, V); axis([xmin xmax ymin ymax 0 22])
% xlabel('x')
% ylabel('y')
% zlabel('v')
% figure(2)
% contour(Xn,Yn,V)
% plot3(Xn, Yn, v(Xn,Yn,tf),'*')
% figure(3)
% surf(Xn, Yn, U); axis([xmin xmax ymin ymax 0 0.4])
% xlabel('x')
% ylabel('y')
% zlabel('u')
% figure(4)
% contour(Xn,Yn,U)
% plot3(Xn,Yn,u(Xn,Yn,tf),'*')

function [F, G] = func(dx, nx, U, V)
```

```
%% constant
k = 0.25; alpha = 0; beta = 1;
F1 = zeros(nx); F2 = zeros(nx);
G1 = zeros(nx); G2 = zeros(nx);
for n = 1:nx
  F1(n,:) = FDM2D2(dx,nx,U(n,:))';
  G1(n,:) = FDM2D2(dx,nx,V(n,:))';
end
for m = 1:nx
  F2(:,m) = FDM2D2(dx,nx,U(:,m));
  G2(:,m) = FDM2D2(dx,nx,V(:,m));
end
F = k*(F1 + F2) + alpha - (beta+1)*U + (U.^2).*V;
G = k*(G1 + G2) + beta*U - (U.^2).*V;
end
```

A.7. Codes for Meshless & Runge Kutta in 2D for Test Problem 4.3

```
clc
clear
% format long
xs=0; xf=1; Nx=18; nx=Nx+1;
ys=0; yf=1; Ny=18; ny=Ny+1;
t0=0; tf=2; Nt=1000; dt=(tf-t0)/Nt;
x=linspace(xs,xf,nx)';
y=linspace(ys,yf,ny)';
[xm, ym]=meshgrid(x,y);
xx=xm(:); yy=ym(:);
alpha=0;
beta=1;
k = 0.25;
advx  = 0;
advy  = 0;
```

```
%% exact solution
exact_u=@(x,y,t) exp(-x-y-0.5*t);
exact_v=@(x,y,t) exp(x+y+0.5*t);

%% initial cond. implementation
uu0=zeros(nx,ny); vv0=zeros(nx,ny);

uu0(1,:)=exact_u(x(1),y,t0);
uu0(end,:)=exact_u(x(end),y,t0);
uu0(:,1)=exact_u(x,y(1),t0);
uu0(:,end)=exact_u(x,y(end),t0);
vv0(1,:)=exact_v(x(1),y,t0);
vv0(end,:)=exact_v(x(end),y,t0);
vv0(:,1)=exact_v(x,y(1),t0);
vv0(:,end)=exact_v(x,y(end),t0);

U0=reshape(uu0',nx*ny,1);
V0=reshape(vv0',nx*ny,1);

% c the value for multiquadric meshfree method
% tol: tolerance for adaptive rk
c=0.42;
[D2]= meshless_2D(xx,yy,nx,ny,c,1,1);%,1,1);
M2=[k*D2, zeros(size(D2)); zeros(size(D2)), k*D2];

%% numerical process
ff=@(X) [alpha-(beta+1).*X(1:length(D2),1)+...
X(1:length(D2),1).^2...
*(X(length(D2)+1:end));beta.*X(1:length(D2),1)-
X(1:length(D2),1).^2.*(X(length(D2)+1:end))];
func=@(t,X) M2*X+ff(X);
```

```
Y=@(U,V) (vertcat(U,V));
t=t0;
h=dt;

for i = 1:Nt+1
u=Y(U0,V0);
k1=func(t,u);
k2=func(t+h/2, u + (h/2)*k1);
k3=func(t+h/2, u + (h/2)*k2);
k4=func(t+h, u + h*k3);
u = u + (h/6)*(k1 + 2*k2 + 2*k3 + k4);

t = t + h;
ut=reshape(u(1:length(U0)),nx,ny)';
vt=reshape(u(length(U0)+1:end),nx,ny)';
ut(1,:)=exact_u(x(1),y,t);
ut(end,:)=exact_u(x(end),y,t);
ut(:,1)=exact_u(x,y(1),t);
ut(:,end)=exact_u(x,y(end),t);
vt(1,:)=exact_v(x(1),y,t);
vt(end,:)=exact_v(x(end),y,t);
vt(:,1)=exact_v(x,y(1),t);
vt(:,end)=exact_v(x,y(end),t);
U0=reshape(ut',nx*ny,1);
V0=reshape(vt',nx*ny,1);
% surf(xm,ym,vt)
end

error_u=norm(ut-exact_u(xm,ym,tf), inf)
error_v=norm(vt-exact_v(xm,ym,tf), inf)

% figure(1)
% % subplot(1,2,1)
% surf(xm,ym,ut)
% hold on
% % plot3(xm, ym, exact_u(xm,ym,tf),'*')
% grid on
% grid minor
% xlabel('x')
% ylabel('y')
% zlabel('u')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,vt)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar
% figure
% levels=[0.02:0.02:0.16];
% contour(xm,ym,c_analytical,levels,':');
% hold on
% [cont,h]=contour(xm,ym,ut,levels,'ShowText','On');
% clabel(cont,h,'FontSize',8)
% legend('boxoff')
% grid on
 grid minor
% xlabel('x (m)')
% ylabel('y (m)')
% yticks(0:0.2:2)
% xticks(0:0.2:2)
% colorbar
% figure(3)
% subplot(1,2,1)
% surf(xm,ym,vt)
% hold on
% plot3(xm, ym, exact_v(xm,ym,tf),'*')
% grid on
% grid minor
% xlabel('x')
% ylabel('y')
% zlabel('v')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,vt)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar
```

A.8. Codes for Meshless & Adaptive Runge Kutta in 2D for Test Problem 4.3

```
clc;
clear;
% close all;
% format long
xs=0; xf=1; Nx=18; nx=Nx+1;
ys=0; yf=1; Ny=18; ny=Ny+1;
t0=0; tf=2;
x=linspace(xs,xf,nx)';
y=linspace(ys,yf,ny)';
[xm, ym]=meshgrid(x,y);
xx=xm(:); yy=ym(:);
alpha=0;
beta=1;
k = 0.25;
advx  = 0;
```

```
advy  = 0;

%% exact solution
exact_u=@(x,y,t) exp(-x-y-0.5*t);
exact_v=@(x,y,t) exp(x+y+0.5*t);

%% initial cond. implementation
uu0=zeros(nx,ny); vv0=zeros(nx,ny);
% for j=1:ny
%   for i=1:nx
%       if (x(i)>=0 && x(i)<=1.5) && (y(j)>=0
            && y(j)<=1.5)
%           ww0(i,j)=inital_w(x(i),y(j));
%           vv0(i,j)=inital_v(x(i),y(j));
%       else
%           ww0(i,j)=0;
%           vv0(i,j)= 1-2*inital_w(x(i),y(j));
%       end
%   end
% end

uu0(1,:)=exact_u(x(1),y,t0);
uu0(end,:)=exact_u(x(end),y,t0);
uu0(:,1)=exact_u(x,y(1),t0);
uu0(:,end)=exact_u(x,y(end),t0);
vv0(1,:)=exact_v(x(1),y,t0);
vv0(end,:)=exact_v(x(end),y,t0);
vv0(:,1)=exact_v(x,y(1),t0);
vv0(:,end)=exact_v(x,y(end),t0);
% figure(1)
% subplot(1,2,1)
% surf(xm,ym,uu0)
% grid on
% grid minor
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{mesh}')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,uu0)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar

% figure(3)
% pcolor(xm,ym,uu0)
U0=reshape(uu0',nx*ny,1);
V0=reshape(vv0',nx*ny,1);

% c the value for multiquadric meshfree method
% tol: tolerance for adaptive rk
c=0.42;
[D2]= meshless_2D(xx,yy,nx,ny,c,1,1);%,1,1);
M2=[k*D2, zeros(size(D2)); zeros(size(D2)), k*D2];

%% numerical process
tol=1e-6;
ff=@(X) [alpha-(beta+1).*X(1:length(D2),1)+...
X(1:length(D2),1).^2...
*(X(length(D2)+1:end));beta.*X(1:length(D2),1)-
X(1:length(D2),1).^2.*(X(length(D2)+1:end))];
func=@(t,X) M2*X+ff(X);
Y=@(U,V) (vertcat(U,V));
```

```
t=t0;
h=tol^(1/5)/4;
step=0;
nrej=0;
fcall=1;
a4=[44/45 -56/15 32/9]';
a5=[19372/6561 -25360/2187 64448/6561 -212/729]';
a6=[9017/3168 -355/33 46732/5247 49/176
-5103/18656]';
a7=[35/384 0 500/1113 125/192 -2187/6784 11/84]';
e=[71/57600 -1/40 -71/16695 71/1920
-17253/339200 22/525]';
while t < tf
  u=Y(U0,V0);
  k1=func(t,u);
if t+h > tf; h=tf-t; end
u_prev=u;
k2=func(t+h/5,u+h*k1/5);
k3=func(t+3*h/10,u+h*(3*k1+9*k2)/40);
k4=func(t+4*h/5,u+h*(a4(1)*k1+a4(2)*k2+a4(3)*k3));
k5=func(t+8*h/9,u+h*(a5(1)*k1+a5(2)*...
k2+a5(3)*k3+a5(4)*k4));
k6=func(t+h,u+h*(a6(1)*k1+a6(2)*k2+a6(3)*k3+...
a6(4)*k4+a6(5)*k5));
u=u+h*(a7(1)*k1+a7(3)*k3+a7(4)*k4+a7(5)*k5+a7(6)*k6);
k2=func(t+h,u);
est=max(max((h*(e(1)*k1+e(2)*k2+e(3)*k3+e(4)*...
k4+e(5)*k5+e(6)*k6))));
fcall=fcall+6;
% rr=norm(u-u_prev,inf)
% [t h est]
```

```
if est < tol
t = t + h;
ut=reshape(u(1:length(U0)),nx,ny)';
vt=reshape(u(length(U0)+1:end),nx,ny)';
ut(1,:)=exact_u(x(1),y,t);
ut(end,:)=exact_u(x(end),y,t);
ut(:,1)=exact_u(x,y(1),t);
ut(:,end)=exact_u(x,y(end),t);
vt(1,:)=exact_v(x(1),y,t);
vt(end,:)=exact_v(x(end),y,t);
vt(:,1)=exact_v(x,y(1),t);
vt(:,end)=exact_v(x,y(end),t);
U0=reshape(ut',nx*ny,1);
V0=reshape(vt',nx*ny,1);
surf(xm,ym,vt)
else
nrej=nrej+1;
end
h=.9*min((tol/(est+eps))^(1/5),10)*h;
end
error_u=norm(ut-exact_u(xm,ym,tf), inf)
error_v=norm(vt-exact_v(xm,ym,tf), inf)
% [ut,vt]=DOPRI5_2D_system(U0,V0,t0,tf,F,tol,...
nx,ny,x,y,exact_u,exact_v);
% A: output for the numerical result
% D2: second der matrix
% D: first der matrix
%-----------------------------------------
% c_analytical=zeros(nx,ny);
% for j=1:ny
%     for i=1:nx
%         c_analytical(i,j)=exact_u(x(i),y(j),tf);
%     end
% end
% err = abs(c_analytical-ut);
% max_err = max(max(err))
% ave_err = sum(sum(err))/(ny*nx)

     % figure(2)
% % subplot(1,2,1)
% surf(xm,ym,ut)
% hold on
% % plot3(xm, ym, exact_u(xm,ym,tf),'*')
% grid on
% grid minor
% xlabel('x')
% ylabel('y')
% zlabel('u')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,vt)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar
% figure
% levels=[0.02:0.02:0.16];
% contour(xm,ym,c_analytical,levels,':');
% hold on
% [cont,h]=contour(xm,ym,ut,levels,'ShowText','On');
% clabel(cont,h,'FontSize',8)
% legend('boxoff')
% grid on
 grid minor
```

```
% xlabel('x (m)')
% ylabel('y (m)')
% yticks(0:0.2:2)
% xticks(0:0.2:2)
% colorbar
% figure(3)
% subplot(1,2,1)
% surf(xm,ym,vt)
% hold on
% plot3(xm, ym, exact_v(xm,ym,tf),'*')
% grid on
% grid minor
% xlabel('x')
% ylabel('y')
% zlabel('v')
% colorbar
% subplot(1,2,2)
% contourf(xm,ym,vt)
% xlabel('x (m)')
% ylabel('y (m)')
% zlabel('c_{ana}')
% colorbar
```