An-Najah National University

Faculty of Graduate Studies

COMPARISON OF NUMERICAL METHODS FOR SOLVING SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

By

Abdul Fattah Boshnaq

Supervisor

Prof. Naji Qatanani

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master's in Computerized Mathematics, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

2024

COMPARISON OF NUMERICAL METHODS FOR SOLVING SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

By

Abdul Fattah Boshnaq

This thesis was discussed on 31/10/2024, and it was approved.

Prof. Naji Qatanani	
Supervisor	Signature
Dr. Saed Mallak	
External Examiner	Signature
Dr. Hadi Hamad	
Internal Examiner	Signature

Dedication

إلى من كان لي سنداً وعوناً طوال عمري، إلى الرجل الأبرز في حياتي

أبي العزيز

إلى القلب المعطاء والصدر الحاني

أمي الحبيبة

إلي رفيقة دربي وشريكة حياتي

زوجتي الغالية

إلى من شد الله بهم عضدي فكانوا خير معين

إخواني وأخواتي

إلى كل من ساعدني ولو بحرف في حياتي الدراسية...

إلى هؤلاء جميعاً: أهديكم هذا العمل

Acknowledgment

أتوجه بالشكر إلى الأساتذة الكرام الذين لم يبخلوا علينا بعلمهم، ولم يألوا جهداً في سبيل المعرفة والعلم.

والشكر موصول إلى إدارة الجامعة التي ذللت الصعوبات من أجل إنجاز هذه الرسالة بجودة وكفاءة.

كما أوجه الشكر والتقدير إلى لجنة المناقشة الفاضلة المتمثلة في مشرفي أ. د. ناجي قطناني والممتحن الداخلي د. هادي حمد والممتحن الخارجي أ. د. سائد ملاك لكم مني جزيل الشكر والعرفان

Decaration

I, the undersigned, declare that I have submitted the thesis entitled:

COMPARISON OF NUMERICAL METHODS FOR SOLVING SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

I further declare that the work presented in this thesis, unless otherwise referenced, is the researcher's own work and has not been submitted elsewhere for any other degree or qualification.

Student's Name: **Abdul Fattah Boshnaq**

Signature: *Abdul Boshnaq*

Date: 31/10/2024

Table of Contents

List of Tables

List of Figures

List of Appendices

COMPARISON OF NUMERICAL METHODS FOR SOLVING SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS

By
Abdul Fattah Boshnaq
Supervisor
Prof. Naji Qatanani

Abstract

In this work, we shed the light on the numerical handling of systems of ordinary differential equations. These systems have wide range of applications in mathematical physics, chemistry, biology, stereology, heat conducing and engineering models.

After introducing some important aspects of systems of differential equations including the solvability of homogeneous and non-homogeneous systems, we focus on the numerical techniques for solving systems of differential equations. Namely; one step and multistep methods. The one step methods include Euler and Runge-Kutta methods. The multistep methods involve Adams-Bashforth method, Adams-Moulton method and the Predictor-Corrector method.

The mathematical framework of these numerical methods together with their convergence properties and their error bound associated with these methods will be presented.

The proposed numerical methods will be illustrated by solving some numerical examples. Numerical results show clearly that the multistep methods are more efficient and give faster convergence than other methods.

Keywords: Comparison, Numerical Methods, Solving Systems.

Introduction

Many physical and technological problems in science and engineering are modeled mathematically by differential equations and systems of differential equations. For example, the mathematical systems involving several springs.

Moreover, systems of ordinary differential equations are encountered in electro-chemistry, ecology, stereology, biology and various engineering applications [22, 29]. A standard class of problem for which considerable and software exist is the initial value problem for first order systems of ordinary differential equations [3, 4, 14].

There are many contributions from well-known mathematicians involved in the numerical approximations of the initial value problems in ordinary differential equations. One of these is the Euler's method which is regarded as the most elementary approximation technique for solving initial value problems[17, 26]. The Runge-Kutta methods which are devised by the German mathematician Carl David Runge (1856-1827) and Martin Wilhelm Kutta (1867-1944). Also, the Adomian decomposition method developed between 1970 and 1990 [2]. This method is known as a semi-analytical method which is used by George Adomian to solve ordinary differential equations and nonlinear partial differential equations [12].

The multistep method known as the fourth order Adam-Bashforth method is devised by the English astronomer and mathematician John Couch Adams (1819 -1892) and the English mathematician Francis Bashforth (1819-1912) [5], then the forth order Adams-Moulton method is devised by the American astronomer Forest Moulton (1872-1952) [20]. The combination of an explicit method to predict and an implicit to improve the prediction is called predictor- corrector method [19].

This thesis is organized as follows:

In chapter one, we introduce some basic definitions and properties including the solvability of homogenous and non-homogenous systems of ordinary differential equations.

In chapter two, we present the numerical methods for solving systems of linear ordinary differential equations. These methods are the one step and multistep methods.

In chapter three, we solve three numerical examples involving initial value problems for first order systems of ordinary differential equations of known exact solution.

In chapter four, involves some comparisons between the proposed numerical methods.

Chapter One
Mathematical Preliminaries

In this chapter we introduce some important aspects for the systems of ordinary differential equations [1, 16, 21].

1.1 Systems of Linear Algebraic Equations

A set of n simultaneous linear algebraic equations in n variables can be written as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{1.1}$$

System (1.1) can be expressed in matrix form as

$$AX = b \tag{1.2}$$

where the $n \times n$ matrix A and the n-dimensional vector b are given, and the components of the n-dimensional vector X are to be determined [31].

If the matrix A is nonsingular that lead to know that, the$\det A$ is not zero, the A^{-1} exists and there is a unique solution founded by multiplying each side of equation (1.2) on the left by A^{-1}.

$$X = A^{-1}b \tag{1.3}$$

Otherwise, the solution of equation (1.2) infinitely many solutions and A^{-1} not exists [13].

1.2 Systems of Linear Differential Equations

This system of linear differential equations in n unknowns has the general form:

$$\frac{dx_1}{dt} = a_{11}(t)x_1 + a_{12}(t)x_2 + \ldots + a_{1n}(t)x_n + f_1(t)$$

$$\frac{dx_2}{dt} = a_{21}(t)x_1 + a_{22}(t)x_2 + \ldots + a_{2n}(t)x_n + f_2(t) \qquad (1.4)$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$\frac{dx_n}{dt} = a_{n1}(t)x_1 + a_{n2}(t)x_2 + \ldots + a_{nn}(t)x_n + f_n(t)$$

We assume that the coefficients $a_{ij}(t)$and $f_i(t)$ are continuous on a common interval I.

When$f_i(t) = 0$, $i = 1,2, \ldots , n$, the linear system (1.4) is said to be homogeneous, otherwise, it is nonhomogeneous[8].

1.3 Initial Value Problem (IVP)

Let t_0 denotes a number in an interval I and

$$X(t_0) = \begin{pmatrix} x_1(t_0) \\ x_2(t_0) \\ \vdots \\ x_n(t_0) \end{pmatrix} \text{ and } X_0 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_n \end{pmatrix}$$

where the$\gamma_i(t)$, $i = 1, 2, \ldots, n$ are given constants,then the problem

$$X'(t) = A(t)X(t) + F(t) \qquad (1.5)$$

subject to

$$X(t_0) = X_0$$

is called an initial value problem.

System (1.4) can be written into the matrix form as:

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}(t) & a_{12}(t) & \ldots & a_{1n}(t) \\ a_{21}(t) & a_{22}(t) & \ldots & a_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(t) & a_{n2}(t) & \ldots & a_{nn}(t) \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_n(t) \end{pmatrix}$$

where

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad A(t) = \begin{pmatrix} a_{11}(t) & a_{12}(t) & \ldots & a_{1n}(t) \\ a_{21}(t) & a_{22}(t) & \ldots & a_{2n}(t) \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(t) & a_{n2}(t) & \ldots & a_{nn}(t) \end{pmatrix}, F = \begin{pmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_n(t) \end{pmatrix} \qquad (1.6)$$

Moreover, in vector form,

$$X'(t) = AX(t) + F \tag{1.7}$$

If the system is homogeneous, then system (1.7) reduces to

$$X'(t) = AX(t) \tag{1.8}$$

A solution vector of system (1.7) is equivalent to n scalar equations $x_1 = \Phi_1(t)$, $x_2 = \Phi_2(t), \dots , x_n = \Phi_n(t)$[23].

1.4 Existence and Uniqueness

To discuss the existence and the uniqueness of a solution for the initial value problem (1.5), we consider the following definitions and theorems.

Definition 1.1[9]:

A function $f(t , x)$is said to satisfy a Lipschitz condition in the variable x on a set $V \subset \mathbb{R}^2$ if a constant $L > 0$ exists with

$$|f(t , x_1) - f(t , x_2)| \leq L\, |x_1 - x_2|$$

whenever(t , x_1) and (t , x_2) are in V. The constant L is called a Lipschitz constant for f.

Definition 1.2 [9]:

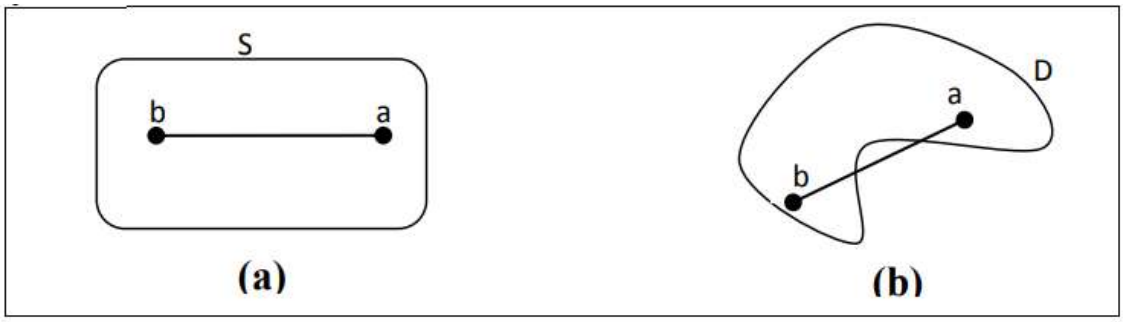
A set $D \subset \mathbb{R}^2$ is said to be convex set if, for any two pointx_1, $x_2$in$D$, the line segment joining $x_1(t)$ and $x_2(t)$ lies entirely with in D. Formally, D is convex if for all x_1, $x_2 \in D$ and for all θ in the interval $[0,1]$ the point

$$\theta x_1 + (1 - \theta)x_2 \in D \qquad\qquad \forall\, \theta \in [0,1]$$

The set S in Figure (1.1) (a) is convex because the line segment joining any two point in S lies entirely within S

The set D in Figure (1.1) (b) is non-convex because there exist point a and b in D such that line segment joining a and b dose not lie entirely within D

Figure (1.1)
is convex set S and non-convex set D



Theorem 1.1[9]:

Suppose $f(t,x)$ is defined on a convex set $V \subset \mathbb{R}^2$. If a constant $L > 0$ exists with

$$\left|\frac{\partial f}{\partial x}(t,x)\right| \leq L$$

for all $(t,x) \in V$

then f satisfies a lipschitz condition on V in the variable y with a Lipschitz constant L.

Proof: Holding t constant and applying the Mean Value Theoremto the function $f(t,x)$, when $x_1 < x_2$, a number α in (x_1, x_2) exists with

$$\lim_{x_2 \to x_1} \frac{f(t\,, x_2) - f(t\,, x_1)}{x_2 - x_1} = \frac{\partial f}{\partial x}(t, x\,)$$

and

$$|f(t\,, x_2) - f(t\,, x_1)| = |x_2 - x_1| \left|\frac{\partial f}{\partial x}(t,x)\right| \leq |x_2 - x_1| L.$$

Thus, f satisfies a Lipschitz condition on V in the variable x with a Lipschitz constant L[28].

Definition 1.3[9]:

The initial value problem

$$\frac{dx}{dt} = f(t, x(t)), \quad a \leq t \leq b, \quad x(a) = \alpha$$

is said to be a well-posed problem if:

A unique solution, $x(t)$to the problem exists and there exist constants $\epsilon_0 > 0$ and $k > 0$, such that for any ϵ, in $(0, \epsilon_0)$ wherever δ(t) is continuous with $|\delta(t)| < \epsilon$ for all t in $[a, b]$, and when $|\delta_0| < \epsilon$, the initial value problem

$$\frac{dy}{dt} = f(t, y) + \delta(t), \qquad a \le t \le b, \qquad y(a) = \alpha + \delta_0$$

has a unique solution $y(t)$ that satisfies

$$|y(t) - x(t)| < k\epsilon \quad for\ all\ t\ in\ [a, b]$$

The second part of the definition says that small perturbations of the original problem and small perturbations of the initial condition have only small error effects on the approximated solution.

Theorem 1.2[9]:

Suppose that $V = \{(t, x): a \le t \le b\ and - \infty < x < \infty\}$. If $f(t, x)$ is continuous and satisfies a lipschitz condition in the variable x on the set V then the initial value problem

$$y'(x) = f(x, y), \qquad a \le x \le b, \quad y(a) = a$$

is well posed.

We note, that this theorem ensures the uniqueness of the solution $x(t)$for $a \le t \le b$.

Theorem 1.3[10]:

Suppose that f and $\frac{\partial f}{\partial x}$ its first partial derivative with respect to x, are continuous for t in $[a, b]$ and for all x. Then the initial value problem

$$\frac{\partial x}{\partial t} = f\big(t, x(t)\big), \qquad a \le t \le b,$$

with initial condition

$$x(a) = \alpha$$

has a unique solution $x(t)$ for $a \le t \le b$, and the problem is well posed.

Proof**:** the set $V = \{(t,x): a \leq t \leq b \; and - \infty < x < \infty\}$ is convex. Since $\frac{\partial f}{\partial x}$ is continuous on $[a,b]$, then $\frac{\partial f}{\partial x}$ is bounded. Therefore, there exists a real number $L > 0$ such that $\left|\frac{\partial f}{\partial x}\right| \leq L$. Thus, f satisfies a Lipschitz condition on V in the variable x. Also f is continuous. It follows from theorem (1.2), that the IVP is well-posed [25].

1.5 Linear Dependence and Linear Independence

The set X is linearly dependent on the interval I if there exist real or complex constants$c_1, c_2, \dots, c_k$at least one of which is nonzero, such that

$$c_1X_1 + c_2X_2 + \dots + c_kX_k = 0 \tag{1.9}$$

On the other hand, if $c_1 = c_2 = \cdots = c_k = 0$, then $X_1, X_2, \dots, X_k$ are said to be linearly independent[8].

To consider the solution ofthe system of ordinary differential equations linearly dependence or independence we take the Wronskian.

Theorem 1.4[31]:

Criterion for linearly independent solution

$$\text{Let} \quad X_1 = \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{pmatrix}, \qquad X_2 = \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{n2} \end{pmatrix}, \dots, X_n = \begin{pmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{nn} \end{pmatrix}$$

be$n$ solution vectors of the homogenous system (1.8) on an interval I. Then the set of solution vectors is linearly independent on interval$I$ if and only if the Wronskian

$$W(X_1, X_2, \dots, X_n) = \begin{vmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{vmatrix} \neq 0 \tag{1.10}$$

for every t in the interval$I$.

If $X_1, X_2, \dots, X_n$ are solution vectors of (1.8) and $W(X_1, X_2, \dots, X_n) \neq 0$ for some t_0 in, then the solutions are linearly independent on the interval$I$.

1.6 Homogeneous Vector Differential Equations

Homogeneous vector differential equations are a class of differential equations where the function and its derivatives are vector-valued, and the equations are linear and homogeneous. They often appear in the context of systems of linear differential equations with vector solutions.

The meaning of homogenous differential equations is the right side is zero, in the other words, it is in the form:

$$X'(t) = AX(t)$$

To solve the system, find the eigenvalues and eigenvectors of the matrix A, the eigenvalues λ's are found by solving the characteristic equation.

$$det(A - \lambda I) = 0 \tag{1.11}$$

where$I$ is the identity matrix.

For each eigenvalue λ, find the corresponding eigenvector V by solving

$$(A - \lambda I) = \vec{0} \tag{1.12}$$

the general solution to the system can be expressed in term of the eigenvalues and eigenvectors. If λ_i is an eigenvalue with corresponding eigenvector V_i, then the solution associated with eigenvalueλ_i is

$$X_i(t) = e^{\lambda_i t} V_i \tag{1.13}$$

the general solution is a linear combination of these solutions

$$X(t) = C_1 e^{\lambda_1 t} V_1 + C_2 e^{\lambda_2 t} V_2 + \cdots + C_n e^{\lambda_n t} V_n$$

where $C_1, C_2, \dots, C_n$are constants determined by initial conditions.

1.7 Eigenvalues and Eigenvectors

Suppose $Ke^{\lambda t}$ be a solution vector of the homogenous system (1.8), then after differentiation and rearranging we obtain

$$(A - \lambda I)K = 0$$

The matrix equation (1.12) is equivalent to the simultaneous algebraic equations

$$\begin{array}{cccc} (a_{11}-\lambda)k_1 + & a_{12}k_2 + & \dots & + a_{1n}k_n = 0 \\ a_{21}k_1 + & (a_{22}-\lambda)k_2 + & \dots & + a_{2n}k_n = 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}k_1 + & a_{n2}k_2 + & \dots & +(a_{nn}-\lambda)k_n = 0 \end{array}$$

the nontrivial solution need to found it

$$det(A - \lambda I) = 0$$

1.8 Kind of Eigenvalues

1.8.1 Distinct real eigenvalues

When an $n \times n$ matrix A has n distinct real eigenvalues$\lambda_1, \lambda_2, \dots, \lambda_n$, it implies, the matrix A has n linearly independent eigenvectors $K_1, K_2, \dots, K_n$ corresponding to the eigenvalues, the solutions of the system of differential equations given by $X' = AX$ can be expressed in the form

$$X_1(t) = K_1 e^{\lambda_1 t}\ , X_2(t) = K_2 e^{\lambda_2 t}\ , \dots\ , X_n(t) = K_n e^{\lambda_n t}$$

these solutions represent a fundamental set of solutions on the interval $(-\infty, \infty)$ [31]. And since $K_1, K_2, \dots, K_n$ are linearly independent, the entire set

$\{X_1, X_2, \dots, X_n\}$ forms a basis for the solution space of the system.

1.8.2 Repeated eigenvalues

The eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ of an $n \times n$ matrix A my be not distinct, some of them my be repeated, for example $\lambda_1 = \lambda_2$ the second solution can be expressed as the form

$$X_2(t) = Kte^{\lambda_1 t} + Pe^{\lambda_1 t} \qquad (1.14)$$

where

$$K = \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \text{ and } P = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}.$$

we substitute (1.14) into the system $X' = AX$ and simplify:

$$(AK - \lambda_1 K)te^{\lambda_1 t} + (AP - \lambda_1 P - K)e^{\lambda_1 t} = 0$$

since this last equation is to hold for all values of t, we must have

$$(A - \lambda_1 I)K = 0 \tag{1.15}$$

$$(A - \lambda_1 I)P = K \tag{1.16}$$

the first solution X_1 is solving system (1.15) for the vector K, which the first solution $X_1 = Ke^{\lambda_1 t}$, the second solution solving system(1.16) for the vector P .

1.8.3 Complex Eigenvalues

If $\lambda_1 = \alpha + i\beta$and $\lambda_2 = \alpha - i\beta, \beta > 0$,are complex eigenvalues of the coefficient matrix A, the eigenvalue λ_2 is the complex conjugate of λ_1. If K_1 is the eigenvector corresponding to λ_1, then K_2 corresponding to λ_2, will often be the complex conjugate of K_1. Thus, both eigenvectors will have complex components.

we defined

$$\mathbf{B_1} = \frac{1}{2}\left(K_1 + \overline{K_1}\right) \text{ and } \mathbf{B_2} = \frac{i}{2}\left(-K_1 + \overline{K_1}\right) \tag{1.17}$$

The linearly independent solutions on (1.8) is

$$X_1(t) = [\mathbf{B_1} \cos\beta t - \mathbf{B_2} \sin\beta t]e^{\alpha t} \tag{1.18}$$

$$X_2(t) = [\mathbf{B_2} \cos\beta t + \mathbf{B_1} \sin\beta t]e^{\alpha t}$$

1.9 Nonhomogeneous Vector Differential Equations

Nonhomogeneous vector differential equations are a generalization of homogeneous vector differential equations that include a non-zero forcing term or external input.

To solve system (1.7) we typically find the general solution by combining the solution to the corresponding homogeneous system and a particular solution to the nonhomogeneous system.

To find a particular solution $X_P(t)$ to the nonhomogeneous system, there are various methods to find a particular solution, including: Method of Undetermined Coefficients and Variation of Parameters.

Theorem 1.5 [31]:

In the nonhomogeneous vector differential equations (1.5) if $\{X_1, X_2, \dots, X_n\}$ is a fundamental solution set of the corresponding homogenous equations (1.8), then every solution $X(t)$ to the nonhomogeneous equations can be expressed in the form

$$X(t) = C_1X_1(t) + C_2X_2(t) + \cdots + C_nX_n(t) + X_P(t)$$

Where $X_P(t)$ is a particular solution.

Proof:

Consider the homogenous equation $X'(t) = A(t)X(t)$, the fundamental solution set $\{X_1, X_2, \dots, X_n\}$ consists of solutions that form a basis for the solution space of the homogenous equation, any solution $X_C(t)$ of the homogenous equation can be expressed as

$$X_C(t) = C_1X_1(t) + C_2X_2(t) + \cdots + C_nX_n(t)$$

where, $C_1, C_2, \dots, C_n$ are constants determined by initial conditions.

Now, let $X_P(t)$ be a particular solution to the nonhomogeneous equation

$$X_P{}'(t) = A(t)X_P(t) + b(t)$$

consider any solution $X(t)$to the nonhomogeneous equation,we can express it as the sum of the complementary solution and a particular solution

$$X(t) = X_C(t) + X_P(t)$$

Then we can write

$$X(t) = C_1X_1(t) + C_2X_2(t) + \cdots + C_nX_n(t) + X_P(t)$$

1.10 The Method of Undetermined Coefficients

This method is technique used to find a particular solution x_p to non-homogenous equation (1.7)

To choose a form for a particular solution $X_p(t)$ based on the form of (t) , the guess should be similar to the nonhomogeneous term $F(t)$, but with undetermined coefficients, the common forms;

- Polynomial: if $F(t)$ is a polynomial of degree n, guess a polynomial of the same degree, if $F(t) = 4t^2 + 2t + 1$ the suggested particular solution will be $x_p(t) = At^2 + Bt + C$.
- Exponential: if $(t) = e^{rt}$, the suggested particular solution will be $x_p(t) = Ae^{rt}$.
- Sine or Cosine: if $F(t) = \sin(rt)$or $F(t) = \cos(rt)$, the suggested particular solution will be $x_p(t) = A\sin(rt) + B\cos(rt)$.
- Combination: if $F(t)$ is a combination of function, combine the respective supposing, if $F(t) = e^{2t}\sin(t)$the suggested particular solution will be $x_p(t) = e^{2t}(A\sin(rt) + B\cos(rt))$

1.11 Variation of Parameters

After finding the homogenous solution of a system, find a fundamental matrix solution $\Phi(t)$, which consists of the solutions to the homogenous system if $X_1, X_2, \dots, X_n$ are the linearly independent solutions, then the fundamental matrix is

$$\Phi(t) = [X_1, X_2, \dots, X_n]$$

assume a particular solution of the form

$$X_P(t) = \Phi(t)U(t)$$

where$U(t)$ is a vector of functions to be determined, differentiate $X_P(t)$

$$X_P'(t) = \Phi(t)U'(t) + \Phi'(t)U(t)$$

subsisting$X_P(t)$ into the original equation

$$\Phi(t)U'(t) + \Phi'(t)U(t) = A(t)\Phi(t)U(t) + F(t)$$

from the homogenous system

$$\Phi'(t) = A(t)\Phi(t)$$

thee equations simplifies to

$$\Phi(t)U'(t) = F(t)$$

multiply both side by $\Phi^{-1}(t)$

$$U'(t) = \Phi^{-1}(t)F(t)$$

integrate $U'(t)$

$$U(t) = \int \Phi^{-1}(t)\, F(t)dt$$

substitute $U(t)$ back into the expression for $X_P(t)$

$$X_P(t) = \Phi(t) \int \Phi^{-1}(t)\, F(t)dt$$

the general solution to the non-homogenous system is

$$X(t) = X_C(t) + X_P(t)$$

1.12 Higher Order Initial Value Problems

Higher order initial value problem involve differential equations of order greater than one.

The higher order initial value problem typically takes the form

$$y^{(n)}(t) = f\left(t\,, y(t)\,, y'(t), \dots, y^{(n-1)}(t)\right) \qquad a \le t \le b$$

with initial conditions

$$y(t_0) = y_0$$

$$y'(t_0) = y_1$$

$$\vdots$$

$$y^{(n-1)}(t_0) = y_{n-1}$$

Methods of solution

1. Reduction of order: higher order equations can often be reduced to a system of first order equations.
2. Numerical methods: when analytical solutions are difficult to find, numerical methods such as Runge-Kutta methods, Multistep method are used.
3. Series solutions: in some cases, power series or Taylor series expansions can be used to approximate the solution.
4. Laplace transforms: this technique transforms the differential equation into an algebraic equation, which can then be solved for the Laplace transform of the function, followed by inverse transformation to find the solution.

Chapter Two

Numerical Methods for Solving System of Differential Equation

In this chapter, we will some of the numerical methods used for solving systems of ordinary differential equations, namely; one step and multistep methods.

2.1 The Adomian Decomposition Method

The Adomian decomposition method(ADM), also known as the inverse operator method, is a mathematical effective approach for solving broad classes of linear and nonlinear mathematical physics equations with important applications in different fields of applied mathematics, engineering, physics and biology, it was proposed by George Adomian (1986, 1988, 1994)[1, 3,4].

The Adomian decomposition method involves the decomposing the unknown equation $F(t) = g(t)$ into the sum of the components of different degree solution, and also to find the solution of each order, for these sum's we want to approximate the true solution to a desired accuracy[30].

First, the whole equation is decomposed into several parts, which is linear and nonlinear, the linear part is divided into invertible and residual of the linear operator and other part is nonlinear [6,7].

The operator F is decomposed as

$$F = L + R + N \tag{2.1}$$

where$L$ is linear part, R is remainder term and N is nonlinear term.

therefore

$$Fu(t) = g(t) = Lu + Ru + Nu \tag{2.2}$$

where$L$ is invertible we got

$$Lu = g(t) - Ru - N$$

we take suitable operator L such that L^{-1} exist and take L^{-1} to both sidesgives

$$u = L^{-1}g - L^{-1}Ru - L^{-1}Nu \tag{2.3}$$

we decompose the true solution n into the sum of infinitely component

$$u = \sum_{n=0}^{\infty} u_n$$

the nonlinear term evaluated to $\sum_{n=0}^{\infty} A_n$ where A_n are special polynomial to be discussed, so the equation

$$\sum_{n=0}^{\infty} u_n = L^{-1}g - L^{-1}R\sum_{n=0}^{\infty} u_n - L^{-1}\sum_{n=0}^{\infty} A_n \tag{2.4}$$

we can write as

$$u_0 = L^{-1}g$$

$$u_1 = -L^{-1}Ru_0 - L^{-1}A_0$$

$$u_2 = -L^{-1}Ru_1 - L^{-1}A_1$$

$$u_{m+1} = -L^{-1}Ru_m - L^{-1}A_m$$

Now for solving system of linear and nonlinear ordinary differential equations

consider the following system of ordinary differential equations [15]:

$$y_i'(x) = \sum_{j=1}^{n} b_{ij}(x)y_i + N_i(x, y_1, y_2, \dots, y_n) + g_i(x) \tag{2.5}$$

withinitial conditions

$$y_i(0) = c_i$$

$$i = 1,2, \dots, n$$

where $b_{ij}(x)$, $g_i(x) \in [0, T]$ and N_i's are nonlinear continuous functions of its argument. Integrating both side of equation (2.1) from 0 to x and the using initial conditions, we got

$y_i(x) =$

$$c_i + \int_0^x g_i(x)\,dx + \int_0^x \sum_{j=1}^{n} b_{ij}(x) y_i\,dx + \int_0^x N_i(x, y_1, y_2, \dots, y_n)\,dx \qquad (2.6)$$

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

The standard ADM yields the solution $y_i(x)$ by series

$$y_i(x) = \sum_{m=0}^{\infty} y_{im}(x) \qquad (2.7)$$

and the nonlinear terms by an infinite series of Adomian polynomials

$$N_i(x, y_1, y_2, \dots, y_n) = \sum_{m=0}^{\infty} A_{im} \qquad (2.8)$$

where $A_{im}(y_{10}, y_{11}, \dots, y_{1m}, y_{21}, \dots, y_{2m}, \dots, y_{n0}, y_{n1}, \dots, y_{nm})$.

These polynomials can be constructed using the general formula [2]

$$A_{im} = \frac{1}{m!}\frac{d^m}{d\lambda^m}\left[N_i\left(x, \sum_{m=0}^{\infty} y_{1m}\,\lambda^m, \dots, \sum_{m=0}^{\infty} y_{nm}\,\lambda^m\right)\right]_{\lambda=0}$$

$$= \left[\frac{1}{m!}\frac{d^m}{d\lambda^m} N_i\left(x, \sum_{m=0}^{\infty} y_{1m}\,\lambda^m, \dots, \sum_{m=0}^{\infty} y_{nm}\,\lambda^m\right)\right]_{\lambda=0}$$

and the components y_{im} , $m \geq 0$, can be determined in a recursive manner. In view of (2.6) - (2.8), the ADM defines the component y_{im} , $m \geq 0$,by the following recursion relation :

$$y_{i0}(x) = c_i + \int_0^x g_i(x)\,dx\,, i = 1,2, \dots, n\,. \qquad (2.9)$$

$$y_{i,m+1}(x) = \int_0^x \sum_{j=1}^{n} b_{ij}(x) y_{im}\,dx + \int_0^x A_{im}(x)\,dx\ , \qquad m = 0\,, 1\,, \dots$$

we approximate the solution $y_i(x)$ by the truncated series

$$f_{ik}(x) = \sum_{m=0}^{k-1} y_{im}(x) \tag{2.10}$$

and

$$\lim_{k\to\infty} f_{ik}(x) = y_i(x), \quad i = 1,2,\dots,n$$

The advantages of Adomian decomposition method:

- Simplicity: The method is straightforward to implement, requiring minimal computational effort for obtaining terms in the series.
- No Linearization: It handles nonlinear problems directly without needing to linearize the equations, preserving the original system's characteristics.
- Convergence: The series solution often converges quickly, providing accurate approximations even with a few terms.
- Flexibility: Applicable to various types of differential equations, including initial and boundary value problems.

The disadvantages of Adomian decomposition method:

- Convergence Issues: While it often converges, some problems may exhibit slow or even non-convergence, depending on the nature of the nonlinearity.
- Complexity for Higher Orders: As the order of the series increases, the calculations for higher-order Adomian polynomials can become complex and cumbersome.
- Initial Condition Sensitivity: The method may be sensitive to initial conditions, which can affect solution accuracy.
- Limited Applicability: It may not be suitable for all types of nonlinear equations, particularly those with discontinuities or singularities.
- Error Estimation: Estimating the error of the approximation can be challenging, making it difficult to assess solution accuracy.

2.2 Runge-KuttaMethod

The Runge-Kutta methods are a family of iterative techniques used for approximating the solutions of ordinary differential equations.

2.2.1 Euler's method

The Euler method is one of the simplest and most straightforward techniques for numerically solving ordinary differential equations (ODE's). It's often used as an introductory method in numerical analysis due to its simplicity.

The Euler method is designed to approximate the solution of an ODE with initial condition of the form:

$$\frac{dy}{dt} = f(x, y) \qquad y(x_0) = y_0 \tag{2.11}$$

on some interval $[a, b]$.

We want to approximate $y(x)$ at the mesh point $x_i = a + ih$ with step size $h = (b - a)/N$, where $i = 0\,, 1\,, \ldots N - 1$.

For each x_i ,the Taylor expansion of $y(x)$ around x_i is given by

$$y(x_{i+1}) = y(x_i) + h\frac{dy}{dx}\Big|_{x_i} + \frac{h^2}{2!}\frac{d^2y}{dx^2}\Big|_{x_i} + \cdots$$

assuming that $y(x)$ is twice continuously differentiable on $[a, b]$,

we get

$$y(x_{i+1}) = y(x_i + h)$$

$$= y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(\epsilon) \tag{2.12}$$

for some ϵ between x_i and $x_i + h$

we neglect the error term in equation (2.12) and using equation (2.11), we obtain the formula

$$y(x_i + h) \approx y(x_i) + hf(x_i\,, y_i)$$

if we denote $y_i \approx y(x_i)$ then

$$y_{i+1} = y_i + hf(x_i, y_i) \tag{2.13}$$

equation (2.13) is known as Euler's method.

TheTaylor's method of order k given by the formula

$$y_{i+1} = y_i + h\varphi_k(x_i, y_i) \tag{2.14}$$

where

$$\varphi_k(x_i, y_i) = f(x_i, y_i) + \frac{h}{2!}f'(x_i, y_i) + \cdots + \frac{h^{k-1}}{k!}f^{(k-1)}(x_i, y_i)$$

2.2.2 Runge-Kutta Methods (RKM)

Generalizing the Euler method involves enhancing its accuracy by allowing multiple evaluations of the derivative within a single step. This approach leads to the development of more sophisticated numerical methods for solving ordinary differential equations.

The generally attributed to Runge(1895) [24], which him forming the basis of what we now call Runge-Kutta methds, Heun(1900)[18], he introduced what is now know as Heun's method. A simple second-order Runge-Kutta method, Kutta (1901) [24] fully characterized fourth-order Runge-Kutta methods and proposed the first explicit methods of this order [11], Nyström (1925) Contributed methods for second-order differential equations and advanced the development of methods for first-order equations, Hutta (1956,1957)[27] Introduced sixth-order Runge-Kutta methods, extending the range of accuracy for numerical solutions.

The Runge-Kutta methods provide a practical approach for numerically solving ordinary differential equations without requiring the differentiation of the function$f(x, y)$. The simplest of these methods, the Runge-Kutta method of order 2 (RK2), offers a balance between simplicity and accuracy.

The formula for RK2 can be expressed as

$$y_{i+1} = y_i + \alpha k_1 + \beta k_2 \tag{2.15}$$

where

$$k_1 = hf(x_i\,, y_i) \tag{2.16}$$

$$k_2 = hf(x_i + \alpha h\,, y_i + \beta k_1) \tag{2.17}$$

the common choice for the coefficients in the RK2 method is

$$\alpha = \beta = \frac{1}{2}$$

thus, we can rewrite the k_2 term as

$$k_2 = hf\left(x_i + \frac{h}{2}\,, y_i + \frac{1}{2}k_1\right) \tag{2.18}$$

this leads to the Runge-Kutta method of order 2, sometimes known as the modified Euler method

$$y_{i+1} = y_i + \tfrac{h}{2}\left[f(x_i\,, y_i) + f\left(x_i + h\,, y_i + hf(x_i\,, y_i)\right)\right] \tag{2.19}$$

or

$$y_{i+1} = y_i + \tfrac{h}{2}(k_1 + k_2) \tag{2.20}$$

The fourth-order Runge-Kutta method (RK4) is indeed one of the most accurate and widely used numerical techniques for solving ordinary differential equations.

To derive the RK4 formula, we need to use multiple evaluations of the function f to capture more information about the functions behavior

calculate k_1:

$$k_1 = h\, f(x_0\,, y_0)$$

calculate k_2:

$$k_2 = h\, f\left(x_0 + \frac{h}{2}\,, y_0 + \frac{1}{2}\, k_1\right)$$

calculate k_3:

$$k_3 = h\, f\left(x_0 + \frac{h}{2}\, , y_0 + \frac{1}{2}\, k_2\right)$$

calculate k_4:

$$k_4 = h\, f(x_0 + h\, , y_0 + k_3)$$

the RK4 formula is given by

$$y_1 = y_0 + \frac{h}{6}(k_1 + \ 2k_2 + 2k_3 + \ k_4)$$

The advantages of Runge-Kutta methods

- Simplicity: It's easy to implement and understand, making it accessible for various applications.
- Higher Accuracy: Higher-order Runge-Kutta methods provide improved accuracy compared to simpler methods like Euler's method, particularly for stiff equations.
- Stability: It offers better stability properties for a wide range of problems, reducing numerical errors in solutions.
- Flexibility: The method can be adapted for both ordinary and partial differential equations, as well as systems of equations.
- No Need for Complex Formulations: Unlike some methods, it doesn’t require the derivation of complex formulas for each problem, streamlining the process.

The disadvantages of Runge-Kutta methods

- Computational Cost: Higher-order Runge-Kutta methods require more function evaluations per step, which can increase computational time and resource usage.
- Stiff Equations: It can struggle with stiff differential equations, where alternative methods like implicit methods for example Backward Differentiation Formula may perform better.
- Global Error: The method may accumulate significant global errors, especially when used with large step sizes, which can affect long-term accuracy.
- Step Size Selection: Choosing an appropriate step size is crucial; too large can lead to instability and inaccuracies, while too small can lead to excessive computation.

2.2.3 System of first order differential equations

Consider an initial value problem of the first order differential equation:

$$\frac{dx}{dt} = f(t, x, y)$$

$$\frac{dy}{dt} = g(t, x, y)$$

with $x(t_0) = x_0$, $y(t_0) = y_0$ and $t_0 \leq t \leq t_n$, this system consists a single of ordinary differential equations, the solution domain is discredited such that t_0 , $t_1 = t_0 + h$, $t_2 = t_0 + 2h$, $t_n = t_0 + nh$, where h is the step size of t.

The solution that is obtained by the RK4 method is given by

$$x_{n+1}(t) = x_n(t) + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4)$$

$$y_{n+1}(t) = y_n(t) + \frac{h}{6}(g_1 + 2g_2 + 2g_3 + g_4)$$

where

$$f_1 = h\, f(t_n, x_n, y_n)$$

$$g_1 = h\, g(t_n, x_n, y_n)$$

$$f_2 = h\, f(t_n + \frac{h}{2}, x_n + \frac{f_1}{2}, y_n + \frac{g_1}{2})$$

$$g_2 = h\, g(t_n + \frac{h}{2}, x_n + \frac{f_1}{2}, y_n + \frac{g_1}{2})$$

$$f_3 = h\, f(t_n + \frac{h}{2}, x_n + \frac{f_2}{2}, y_n + \frac{g_2}{2})$$

$$g_3 = h\, g(t_n + \tfrac{h}{2}, x_n + \tfrac{f_2}{2}, y_n + \tfrac{g_2}{2})$$

$$f_4 = h\, f(t_n + h, x_n + f_3\,, y_n + g_3)$$

$$g_4 = h\, g(t_n + h, x_n + f_3\,, y_n + g_3)$$

2.3 Adams-Bashforth Method (ABM)

The Adams-Bashforth method is a family of explicit multistep methods used to solve ordinary differential equations. These methods use information from previous time steps to estimate the value at the next time step.

If we integrating the solution of the initial value problem (2.13) between x_i and x_{i+1} ,

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x\,, y(x))\ dx$$

thus

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x\,, y(x)\,)\, dx \tag{2.21}$$

to carry out the integration in (2.21) we can use a finite-difference method to approximate $f(x\,, y)$ at some of the data points $x_0, x_1, \ldots, x_i$ .This will lead to the formula

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} p(x)\, dx \tag{2.22}$$

where

$$y(x_i) \approx y(x_{i+1}) \text{ and } p(x) \approx f(x\,, y).$$

The Newton backward-difference polynomial $p_{m-1}(x)$ based on m point $\big(x_i\,, y(x_i)\big)$, $\big(x_{i-1}\,, y(x_{i-1})\big), \ldots, \big(x_{i-m+1}\,, y(x_{i-m+1})\big)$ can be expressed as

$$p_{m-1}(x) = y_i + (x - x_i)\ \nabla\, y_i + \frac{(x - x_i)(x - x_{i-1})}{2!}\nabla^2 y_i + \cdots$$
$$+ \frac{(x - x_i)(x - x_{i-1}) \ldots\ (x - x_{i-m+1})}{m!}\nabla^m y_i$$

where$\nabla^k y_i$ is the k^{th} backward difference of y.

Substituting $p_{m-1}(x)$ into (2.11), we evaluate x_{i+1}

$$p_{m-1}(x_{i+1}) = y_{i+1}$$

this gives

$$y_{i+1} = y_i + h(f(x_i, y_i) + \frac{h}{2!}\nabla f(x_i) + \frac{h^2}{3!}\nabla^2 f(x_i) + \cdots + \frac{h^{m-1}}{m!}\nabla^{m-1} f(x_i))$$

from this, we can derive the ABM for m points

$$y_{i+1} = y_i + \frac{h}{m!}\sum_{j=0}^{m-1} b_j\, f(x_{i-j}, y_{i-j})$$

whereb_j are the coefficients determined from the polynomial.

The coefficients b_j depend on the order m and can be derived from the Newton polynomial, for $m = 1$

$$y_{i+1} = y_i + hf(x_i, y_i)$$

for $m = 2$

$$y_{i+1} = y_i + \frac{h}{2}(3f(x_i, y_i) - f(x_{i-1}, y_{i-1}))$$

for $m = 3$

$$y_{i+1} = y_i + \frac{h}{12}(23f(x_i, y_i) - 16\, f(x_{i-1}, y_{i-1}) + 5f(x_{i-2}, y_{i-2}))$$

The forth order Adams-Bashforth ($m = 4$):

$$y_{i+1} = y_i + \frac{h}{24}(55\, f(x_i, y_i) - 59\, f(x_{i-1}, y_{i-1}) + 37f(x_{i-2}, y_{i-2}) - 9f(x_{i-3}, y_{i-3}))$$

The advantages of Adams-Bashforth methods

- Efficiency: Adams-Bashforth methods are efficient for long-term integration since they use multiple previous points to estimate future values.
- Explicit nature: Being explicit, these methods do not require solving systems of equations, simplifying their implementation.

The disadvantages of Adams-Bashforth methods

- Not self-starting: Adams-Bashforth methods need a set of initial values to start, often provided by a different method like Runge-Kutta.
- Stability: Explicit methods can be less stable, especially for stiff differential equations, compared to implicit methods.

2.4 Adams- Moulton Method (AMM)

The Adam-Moulton formulas are implicit multistep methods, to derive the second order Adams-Moulton formula, we first start with a first degree polynomial $p_1(x) = \alpha x + \beta$, and we determine the coefficients α and β using the points (x_n , y_n) and (x_{n+1} , y_{n+1}).

the polynomial $p_1(x)$ must satisfy:

$$p_1(x_n) = y_n \text{and} p_1(x_{n+1}) = y_{n+1}$$

plugging in the values:

at x_n,

$$p_1(x_n) = \alpha x_n + \beta = y_n$$

at x_{n+1},

$$p_1(x_{n+1}) = \alpha(x_n + h\,) + \beta = y_{n+1}$$

where$x_{n+1} = x_n + h$

this gives two equations:

$$\alpha x_n + \beta = y_n \tag{2.23}$$

$$\alpha(x_n + h\,) + \beta = y_{n+1} \tag{2.24}$$

subtract equation (2.23) from equation (2.24)

$$\alpha(x_n + h\,) + \beta - (\alpha x_n + \beta) = y_{n+1} - y_n$$

simplifying this, we get

$$\alpha h = y_{n+1} - y_n$$

we have

$$\alpha = \frac{y_{n+1} - y_n}{h} \tag{2.25}$$

substituteα from (2.25) into equation (2.23) to find β

$$\frac{y_{n+1} - y_n}{h} x_n + \beta = y_n$$

rearranging gives

$$\beta = y_n - \frac{y_{n+1} - y_n}{h}$$

the Adams-Moulton method is gives by

$$y_{n+1} = y_n + \frac{h}{2}\,(f_n + f_{n+1})$$

where$f_n = f(x_n\,, y_n)$ and $f_{n+1} = f(x_{n+1}\,, y_{n+1})$

the fourth-order Adams-Moulton (AM4) formula:

$$y_{i+1} = y_i + \tfrac{h}{24}\big(9\,f(x_{i+1}\,, y_{i+1}) + 19\,f(x_i\,, y_i) - 5f(x_{i-1}\,, y_{i-1}) + f(x_{i-2}\,, y_{i-2})\big) \tag{2.26}$$

The advantages of Adams-Moulton method

- Increased stability: Adams-Moulton methods tend to be more stable than explicit methods, making them suitable for stiff differential equations.
- Higher accuracy: These methods can be more accurate, particularly when higher-order variants are used.

The disadvantages of Adams-Moulton method

- Computational cost: The need to solve implicit equations at each step increases computational complexity and requires iterative solvers.
- Not self-starting: Similar to Adams-Bashforth methods, Adams-Moulton methods need initial values from a single-step method for the starting values.

2.5 Predictor-Corrector Method (PCM)

The Predictor-Corrector method is a class of numerical techniques used to solve ordinary differential equations. These methods combine two steps: a prediction step to estimate the next value and a correction step to refine this estimate.

A prediction step is an explicit method that uses information from previous steps to predict the next value, it requires four evaluations from previous steps but only needs one new function evaluation per step, the explicit methods are derived using Newton's backward-difference polynomial, which provides formulas like the Adams-Bashforth methods.

A correction step is an implicit method that refines the predication using both current and previous values, it typically requires two function evaluations per step.

To illustrate the derivation of an implicit formula consider the integral form of the solution to the ordinary differential equations

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y)\, dx$$

to approximate this integral, the trapezoidal rule can be used. The trapezoidal rule approximates the integral as follows

$$\int_{x_i}^{x_{i+1}} f(x, y)\, dx \approx \frac{h}{2}[\, f(x_i, y_i) + f(x_{i+1}, y_{i+1})]$$

where$h = x_{i+1} - x_i$ is the step size

substituting this approximation into the integral equation

$$y_{i+1} = y_i + \frac{h}{2}[\, f(x_i, y_i) + f(x_{i+1}, y_{i+1})]$$

The advantages of Predictor-Corrector method

- Improved accuracy: predictor-corrector methods often achieve higher accuracy compared to using either method alone.

- Flexibility: predictor-corrector methods can be adjusted based on the desired accuracy and stability.
- Stability: the corrector step, being implicit, can improve stability, especially for differential equations.

The disadvantages of Predictor-Corrector method

- Computational cost: corrector step requires solving an implicit equation, which can increase computational complexity and effort.
- Not self-starting: like many multistep methods, predictor-corrector methods need initial values provided by a single-step method.

2.6 Error Analysis

Error analysis in the context of solving systems of linear differential equations involves understanding how numerical methods approximate solutions and how errors accumulate and affect these solutions.

Euler's method has local truncation error is $O(h^2)$, this means the error made in a single step is proportional to the square of the step size (h), also for the global truncation error is $O(h)$ because the global truncation error accumulates over $\frac{1}{h}$steps, leading to an overall error proportional to the step size (h), for stability Euler's method is conditionally stable, for stiff system, Euler's method might exhibit numerical instability unless small step sizes are used.

The Runge-Kutta methods has local truncation error is $O(h^5)$, this indicates that the error is a single step is very small for a small (h) for the global truncation error for $O(h^4)$, because the error accumulates over $\frac{1}{h}$ steps, leading to an overall error proportional to h^4. RK4is generally more stable compared to Euler's method, especially for non-differential systems. It can be applied with larger step size without significant loss of accuracy.

The Adams-Bashforth (Predictor) and Adams-Moulton (Corrector) methods are multistep methods that use previous values to predict and correct future values. For these methods, Local Truncation Error decreases with the order of the method. For example, Adams-Bashforth method of order 4 has Local Truncation Error $O(h^5)$,the global truncation error for Adams-Bashforth methods is $O(h^p)$ where $p$is the order of the method. Adams-

Moulton method also have a similar order of accuracy but typically require solving nonlinear equations for correction.The multistep methods can be more stable for certain problems but can be challenging to apply directly to differential systems. They generally require a good initial guess or values provided by a single-step method.

2.7 Stability and Convergence

Stability: For differential systems, implicit methods (like Adams-Moulton) or implicit Runge-Kutta methods are preferred due to their better stability properties.

Convergence: Ensuring that the numerical method converges to the true solution as the step size decreases is crucial. Methods of higher order generally converge faster, but for differential systems, implicit methods or specialized techniques might be required.

Chapter Three

Numerical Examples and Results

In this chapter, we will implement the numerical methods discussed in Chapter Two to solve several numerical examples. This implementation will involve using appropriate algorithms and MATLAB software. We will compare the exact solutions with the approximate solutions obtained through these methods. The results of this comparison will be presented both in tabular form and through graphical illustrations.

Example 3.1

Consider the system of linear differential equations

$$\frac{dx}{dt} = 2x - y \tag{3.1}$$

$$\frac{dy}{dt} = x$$

subject to the initial conditions

$$x(0) = 6 \tag{3.2}$$

$$y(0) = 2$$

To find the exact solution of system (3.1), we have

$$x(t) = y'(t)$$

now, substitute $x = y'$ into the equation $x' = 2x - y$ yields

$$y'' - 2y' + y = 0$$

solving this equations gives

$$y(t) = (C_1 + C_2 t)e^t$$

Now find C_1 and C_2 with initial condition gives

$C_1 = 2$ and $C_2 = 4$

the exact solution to the system (3.1) is

$$x(t) = 6\,e^t + 4\,te^t$$

$$y(t) = 2\,e^t + 4\,te^t$$

3.1 The numerical approximation of system (3.1) using the fourth order Rung Kutta method

The following algorithm implements the fourth order Rung Kutta method using the matlab software.

Algorithm 3.1

In order to approximate the solution of the general initial value problem

$$x'(t) = f(t,x)\,, \qquad a \le t \le b$$

and

$$x(a) = x_0$$

We follow the algorithm 3.1

$$\text{Set } h = \tfrac{a-b}{n};$$

$$t_0 = a;$$

$$k_1 = f(t_n, x_n);$$

$$k_2 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right);$$

$$k_3 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2\right);$$

$$k_4 = f(t_n + h\,, x_n + hk_3);$$

$$x_{n+1} = x_n + \tfrac{h}{6}(k_1 + 2\,k_2 + \,2\,k_3 + k_4);$$

$$t_{k+1} = a + \frac{k+1}{h};$$

we repeated the code for$k = 0,1, \dots , n-1$.

After these steps the value x_k is an approximated value of solution $x(t_k)$ of the differential at t_k.

We illustrate the computational of x_1 and $y_1$with step size $h = 0.2$

$f(t,x,y) = 2\,x - y$, $g(t,x,y) = x$, $t_0 = 0$, $x_0 = 6$ and $y_0 = 2$

$$k_{1,1} = h\,f(t_0\,, x_0\,, y_0) = 0.2\,f(0\,,6\,,2) = 0.2\,(10) = 2$$

$$k_{2,1} = h\,g(t_0\,, x_0\,, y_0) = 0.2\,g(0\,,6\,,2) = 0.2\,(6) = 1.2$$

$$k_{1,2} = h\,f\left(t_0 + \frac{h}{2}, x_0 + \frac{k_{1,1}}{2}, y_0 + \frac{k_{2,1}}{2}\right) = 0.2\,f(0.1\,, 6 + 1\,, 2 + 0.6\,)$$

$$= \ 0.2\,f(0.1\,, 7\,, 2.6\,) = 0.2\,(11.4) = \ 2.28$$

$$k_{2,2} = h\,g\left(t_0 + \frac{h}{2}, x_0 + \frac{k_{1,1}}{2}, y_0 + \frac{k_{2,1}}{2}\right) = 0.2\,g(0.1\,, 6 + 1\,, 2 + 0.6\,)$$

$$= 0.2\,g(0.1\,, 7\,, 2.6\,) = 0.2\,(7) = 1.4$$

$$k_{1,3} = h\,f\left(t_0 + \frac{h}{2}, x_0 + \frac{k_{1,2}}{2}, y_0 + \frac{k_{2,2}}{2}\right) = 0.2\,f(0.1\,, 6 + 1.14\,, 2 + 0.7\,)$$

$$= 0.2\,f(0.1\,, 7.14\,, 2.7\,) = 0.2\,(11.58) = 2.316$$

$$k_{2,3} = h\,g\left(t_0 + \frac{h}{2}, x_0 + \frac{k_{1,2}}{2}, y_0 + \frac{k_{2,2}}{2}\right) = 0.2\,g(0.1\,, 6 + 1.14\,, 2 + 0.7\,)$$

$$= 0.2\,g(0.1\,, 7.14\,, 2.7\,) = 0.2\,(7.14) = 1.428$$

$$k_{1,4} = h\,f(t_0 + h\,, x_0 + k_{1,3}\,, y_0 + k_{2,3}) = 0.2\,f(0.2\,, 6 + 2.316\,, 2 + 1.428\,)$$

$$= \ 0.2\,f(0.2\,, 8.316\,, 3.428\,) = 0.2\,(13.204) = \ 2.6408$$

$$k_{2,4} = h\,g(t_0 + h\,, x_0 + k_{1,3}\,, y_0 + k_{2,3}) = 0.2\,g(0.2\,, 6 + 2.316\,, 2 + 1.428\,)$$

$$= \ 0.2\,g(0.2\,, 8.316\,, 3.428\,) = 0.2\,(8.316) = \ 1.6632$$

$$x_1 = \ x_0 + \frac{1}{6}\left(\,k_{1,1} + \ 2\,k_{1,2} + 2\,k_{1,3} + k_{1,4}\right)$$

$$= 6 + \frac{1}{6}(2 + 2\,(2.28) + 2\,(2.316) + 2.6408)$$

$$= 6 + \frac{1}{6}(13.8328) = 8.3054$$

$$y_1 = y_0 + \frac{1}{6}\left(k_{2,1} + 2\,k_{2,2} + 2\,k_{2,3} + k_{2,4}\right)$$

$$= 2 + \frac{1}{6}(1.2 + 2\,(1.4) + 2\,(1.428) + 1.6632)$$

$$= 2 + \frac{1}{6}(8.5192) = 3.4198$$

These numbers give us the approximation $x_1 \approx x(0.2) = 8.3054$ and $y_1 \approx y(0.2) = 3.4198$

Now to find $x(0.4)$ and $y(0.4)$ we need to find $t_1 = t_0 + h = 0 + 0.2 = 0.2$

For $t_1 = 0.2$ we need to find

$$k_{1,1} = h\,f(t_1\,, x_1\,, y_1) = 0.2\,f(0.2\,, 8.3054\,, 3.4198) = 0.2\,(13.191) = 2.6382$$

$$k_{2,1} = h\,g(t_1\,, x_1\,, y_1) = 0.2\,g(0.2\,, 8.3054\,, 3.4198) = 0.2\,(8.3054) = 1.661$$

$$k_{1,2} = h\,f\left(t_1 + \frac{h}{2}, x_1 + \frac{k_{1,1}}{2}, y_1 + \frac{k_{2,1}}{2}\right)$$

$$= 0.2\,f\left(0.2 + 0.1\,, 8.3054 + \frac{2.6382}{2}, 3.4198 + \frac{1.661}{2}\right)$$

$$= 0.2\,f(0.3\,, 9.6245\,, 4.2503\,) = 0.2\,(14.9987) = 2.9997$$

$$k_{2,2} = h\,g\left(t_1 + \frac{h}{2}, x_1 + \frac{k_{1,1}}{2}, y_1 + \frac{k_{2,1}}{2}\right)$$

$$= 0.2\,g\left(0.2 + 0.1\,, 8.3054 + \frac{2.6382}{2}, 3.4198 + \frac{1.661}{2}\right)$$

$$= 0.2\,g(0.3\,, 9.6245\,, 4.2503\,) = 0.2\,(9.6245) = 1.9249$$

$$k_{1,3} = h\,f\left(t_1 + \frac{h}{2}, x_1 + \frac{k_{1,2}}{2}, y_1 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\,f\left(0.2 + 0.1\,, 8.3054 + \frac{2.9997}{2}, 3.4198 + \frac{1.9249}{2}\right)$$

$$= 0.2\, f(0.3\,, 9.8052\,, 4.3822\,) = 0.2\,(15.2282) = 3.0456$$

$$k_{2,3} = h\, g\left(t_1 + \frac{h}{2}, x_1 + \frac{k_{1,2}}{2}, y_1 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\, g\left(0.2 + 0.1\,, 8.3054 + \frac{2.9997}{2}, 3.4198 + \frac{1.9249}{2}\right)$$

$$= 0.2\, g(0.3\,, 9.8052\,, 4.3822\,) = 0.2\,(9.8052) = 1.961$$

$$k_{1,4} = h\, f(t_1 + h\,, x_1 + k_{1,3}\,, y_1 + k_{2,3})$$

$$= 0.2\, f(0.2 + 0.2\,, 8.3054 + 3.0456\,, 3.4198 + 1.961\,)$$

$$= 0.2\, f(0.4\,, 11.351\,, 5.3808\,) = 0.2\,(17.3212) = 3.4642$$

$$k_{2,4} = h\, g(t_1 + h\,, x_1 + k_{1,3}\,, y_1 + k_{2,3})$$

$$= 0.2\, g(0.2 + 0.2\,, 8.3054 + 3.0456\,, 3.4198 + 1.961\,)$$

$$= 0.2\, g(0.4\,, 11.351\,, 5.3808\,) = 0.2\,(11.351) = 2.2702$$

$$x_2 = x_1 + \frac{1}{6}\left(k_{1,1} + 2\,k_{1,2} + 2\,k_{1,3} + k_{1,4}\right)$$

$$= 8.3054 + \frac{1}{6}(2.6382 + 2\,(2.9997) + 2(3.0456) + 3.4642)$$

$$= 8.3054 + \frac{1}{6}(18.193) = 11.3376$$

$$y_2 = y_1 + \frac{1}{6}\left(k_{2,1} + 2\,k_{2,2} + 2\,k_{2,3} + k_{2,4}\right)$$

$$= 3.4198 + \frac{1}{6}(1.661 + 2\,(1.9249) + 2\,(1.961) + 2.2702)$$

$$= 3.4198 + \frac{1}{6}(11.703) = 5.3704$$

These numbers give us the approximation $x_2 \approx x(0.4) = 11.3376$ and $y_2 \approx y(0.4) = 5.3704$

Now to find $x(0.6)$ and $y(0.6)$ we need to find $t_2 = t_1 + h = 0.2 + 0.2 = 0.4$

For $t_2 = 0.4$ we need to find

$$k_{1,1} = h\, f(t_2\,, x_2\,, y_2) = 0.2\, f(0.4\,, 11.3376\,, 5.3704) = 0.2\ (17.3047) = 3.4609$$

$$k_{2,1} = h\, g(t_2\,, x_2\,, y_2) = 0.2\, g(0.4\,, 11.3376\,, 5.3704) = 0.2\ (11.3375) = 2.2675$$

$$k_{1,2} = h\, f\left(t_2 + \frac{h}{2}\,, x_2 + \frac{k_{1,1}}{2}\,, y_2 + \frac{k_{2,1}}{2}\right)$$

$$= 0.2\, f\left(0.4 + 0.1\,, 11.3376 + \frac{3.4609}{2}\,, 5.3704 + \frac{2.2675}{2}\right)$$

$$= \ 0.2\, f(0.5\,, 13.0679\,, 6.504\,) = 0.2\ (19.6318) = \ 3.9264$$

$$k_{2,2} = h\, g\left(t_2 + \frac{h}{2}\,, x_2 + \frac{k_{1,1}}{2}\,, y_2 + \frac{k_{2,1}}{2}\right)$$

$$= 0.2\, g\left(0.4 + 0.1\,, 11.3376 + \frac{3.4609}{2}\,, 5.3704 + \frac{2.2675}{2}\right)$$

$$= \ 0.2\, g(0.5\,, 13.0679\,, 6.504\,) = 0.2\ (13.0679) = \ 2.6136$$

$$k_{1,3} = h\, f\left(t_2 + \frac{h}{2}\,, x_2 + \frac{k_{1,2}}{2}\,, y_2 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\, f\left(0.4 + 0.1\,, 11.3376 + \frac{3.9264}{2}\,, 5.3704 + \frac{2.6136}{2}\right)$$

$$= \ 0.2\, f(0.5\,, 13.3006\,, 6.677\,) = 0.2\ (19.9242) = \ 3.9849$$

$$k_{2,3} = h\, g\left(t_2 + \frac{h}{2}\,, x_2 + \frac{k_{1,2}}{2}\,, y_2 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\, g\left(0.4 + 0.1\,, 11.3376 + \frac{3.9264}{2}\,, 5.3704 + \frac{2.6136}{2}\right)$$

$$= \ 0.2\, g(0.5\,, 13.3006\,, 6.677\,) = 0.2\ (13.3006) = \ 2.6601$$

$$k_{1,4} = h\, f(t_2 + h\,, x_2 + k_{1,3}\,, y_2 + k_{2,3})$$

$$= 0.2\, f(0.4 + 0.2\,, 11.3376 + 3.9849\,, 5.3704 + 2.6601\,)$$

$$= \ 0.2\, f(0.6\,, 15.3223\,, 8.0304\,) = 0.2\ (22.6142) = \ 4.5229$$

$$k_{2,4} = h\, g(t_2 + h\,, x_2 + k_{1,3}\,, y_2 + k_{2,3})$$

$$= 0.2\, g(0.4 + 0.2\,, 11.3376 + 3.9849\,, 5.3704 + 2.6601\,)$$

$$= 0.2\, g(0.6\,, 15.3223\,, 8.0304\,) = 0.2\,(15.3223) = 3.0645$$

$$x_3 = x_2 + \tfrac{1}{6}\left(k_{1,1} + 2\,k_{1,2} + 2\,k_{1,3} + k_{1,4}\right)$$

$$= 11.3376 + \tfrac{1}{6}(3.4609 + 2\,(3.9264) + 2\,(3.9849) + 4.5229)$$

$$= 11.3376 + \tfrac{1}{6}(23.8059) = 15.3054$$

$$y_3 = y_2 + \tfrac{1}{6}\left(k_{2,1} + 2\,k_{2,2} + 2\,k_{2,3} + k_{2,4}\right)$$

$$= 5.3704 + \tfrac{1}{6}(2.2675 + 2\,(2.6136) + 2\,(2.6601) + 3.0645)$$

$$= 5.3704 + \tfrac{1}{6}(15.8791) = 8.017$$

These numbers give us the approximation $x_3 \approx x(0.6) = 15.3054$ and $y_3 \approx y(0.6) = 8.017$

Now to find $x(0.8)$ and $y(0.8)$ we need to find $t_3 = t_2 + h = 0.4 + 0.2 = 0.6$

For $t_3 = 0.6$ we need to find

$$k_{1,1} = h\, f(t_3\,, x_3\,, y_3) = 0.2\, f(0.6\,, 15.3054\,, 8.017) = 0.2\,(22.5934) = 4.5187$$

$$k_{2,1} = h\, g(t_3\,, x_3\,, y_3) = 0.2\, g(0.6\,, 15.3054\,, 8.017) = 0.2\,(15.3051) = 3.061$$

$$k_{1,2} = h\, f\left(t_3 + \tfrac{h}{2}\,, x_3 + \tfrac{k_{1,1}}{2}\,, y_3 + \tfrac{k_{2,1}}{2}\right)$$

$$= 0.2\, f\left(0.6 + 0.1\,, 15.3051 + \tfrac{4.5187}{2}\,, 8.0168 + \tfrac{3.061}{2}\right)$$

$$= 0.2\, f(0.7\,, 17.5644\,, 9.5473\,) = 0.2\,(25.5815) = 5.1164$$

$$k_{2,2} = h\, g\left(t_3 + \tfrac{h}{2}\,, x_3 + \tfrac{k_{1,1}}{2}\,, y_3 + \tfrac{k_{2,1}}{2}\right)$$

$$= 0.2\, g\left(0.6 + 0.1\,, 15.3054 + \tfrac{4.5187}{2}\,, 8.017 + \tfrac{3.061}{2}\right)$$

$$= 0.2\, g(0.7\,, 17.5644\,, 9.5473\,) = 0.2\,(17.5644) = 3.5129$$

$$k_{1,3} = h\, f\left(t_3 + \frac{h}{2}, x_3 + \frac{k_{1,2}}{2}, y_3 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\, f\left(0.6 + 0.1\,, 15.3054 + \frac{5.1164}{2}, 8.017 + \frac{3.5129}{2}\right)$$

$$= \ 0.2\, f(0.7\,, 17.8632\,, 9.7732\,) = 0.2\ (25.9532) = \ 5.1907$$

$$k_{2,3} = h\, g\left(t_3 + \frac{h}{2}, x_3 + \frac{k_{1,2}}{2}, y_3 + \frac{k_{2,2}}{2}\right)$$

$$= 0.2\, g\left(0.6 + 0.1\,, 15.3054 + \frac{5.1164}{2}, 8.017 + \frac{3.5129}{2}\right)$$

$$= \ 0.2\, g(0.7\,, 17.8632\,, 9.7732\,) = 0.2\ (17.8632) = \ 3.5727$$

$$k_{1,4} = h\, f(t_3 + h\,, x_3 + k_{1,3}\,, y_3 + k_{2,3})$$

$$= 0.2\, f(0.6 + 0.2\,, 15.3054 + 5.1907\,, 8.017 + 3.5727\,)$$

$$= \ 0.2\, f(0.8\,, 20.4957\,, 11.5894\,) = 0.2\ (29.402) = \ 5.8805$$

$$k_{2,4} = h\, g(t_3 + h\,, x_3 + k_{1,3}\,, y_3 + k_{2,3})$$

$$= 0.2\, g(0.6 + 0.2\,, 15.3054 + 5.1907\,, 8.017 + 3.5727\,)$$

$$= \ 0.2\, g(0.8\,, 20.4957\,, 11.5894\,) = 0.2\ (20.4957) = \ 4.0992$$

$$x_4 = \ x_3 + \frac{1}{6}\left(\ k_{1,1} + \ 2\ k_{1,2} + 2\ k_{1,3} + k_{1,4}\right)$$

$$= 15.3051 + \frac{1}{6}(4.5187 + \ 2(5.1164) + 2(5.1907) + 5.8805)$$

$$= \ 15.3051 + \frac{1}{6}(31.0128) = 20.4744$$

$$y_4 = \ y_3 + \frac{1}{6}\left(\ k_{2,1} + \ 2\ k_{2,2} + 2\ k_{2,3} + k_{2,4}\right)$$

$$= 8.017 + \frac{1}{6}(3.061 + \ 2(3.5129) + 2(3.5727) + 4.0992)$$

$$= \ 8.0168 + \frac{1}{6}(21.3309) = 11.5723$$

These numbers give us the approximation $x_4 \approx x(0.8) = 20.4744$ and

$y_4 \approx y(0.8) = 11.5723.$

Table 3.1(a) and 3.2(b), contain the k's value when applying algorithm 3.1 for system 3.1

Table 3.1(a)
The k's values for $x(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	2.000000000	2.280000000	2.316000000	2.640800000
0.4	2.638213333	2.999746666	3.045671200	3.464268213
0.6	3.460991610	3.926436208	3.984915212	4.522921522
0.8	4.518776273	5.116422417	5.190763883	5.880535156

Table 3.1(b)
The k's values for $y(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	1.200000000	1.400000000	1.428000000	1.663200000
0.4	1.661093333	1.924914666	1.961068000	2.270227573
0.6	2.267537242	2.613636403	2.660180863	3.064520285
0.8	3.061091108	3.512968735	3.572733350	4.099243885

Table 3.2 (a), the exact and numerical solution for $x(t)$ when applying RKM on system (3.1) and showing the resulting error.

Table 3.2(a)
The exact and numerical solution of $x(t)$ by applying RKM

t	**Exact solution**	**Approximate solution**	**Absolute error**
0	6.000000000	6.000000000	0.000000000
0.2	8.305538755	8.305466660	0.000072095
0.4	11.33786770	11.33768621	0.00018149
0.6	15.30579792	15.30545554	0.00034238
0.8	20.47497654	20.47440288	0.00057366

From table 3.2(a) we conclude the maximum error $= 0.00057366$.

Figure 3.1(a): Compares the exact solution $x(t)$ and approximate solution with time t.

Figure 3.1(a)
The exact and numerical solutions of $x(t)$

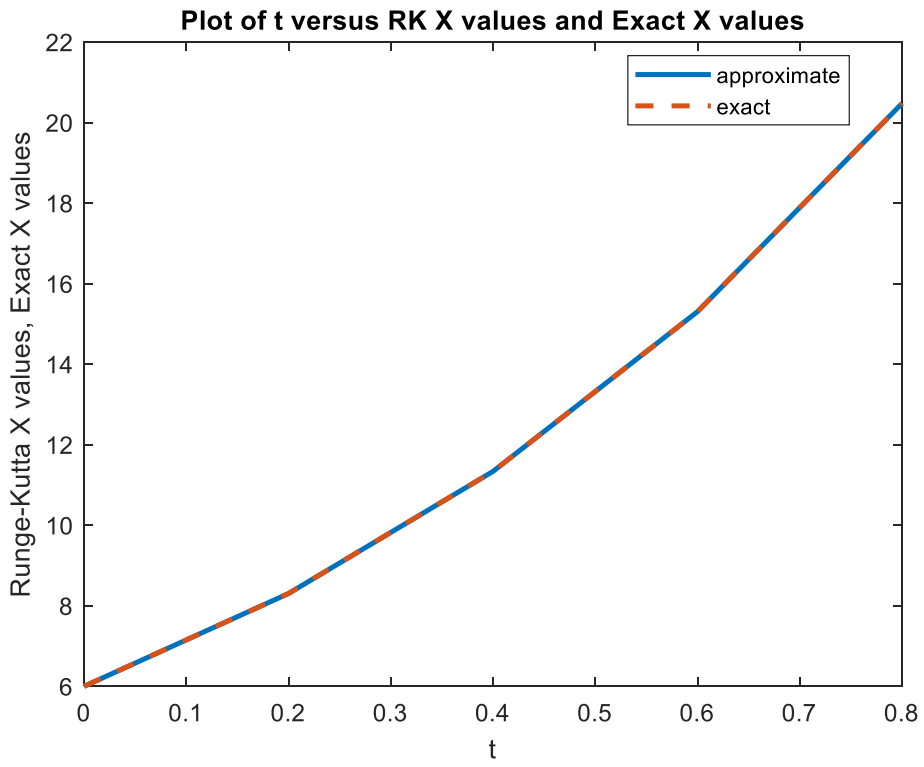


Figure 3.1(b)
the absolute error resulting of applying RKM of $x(t)$

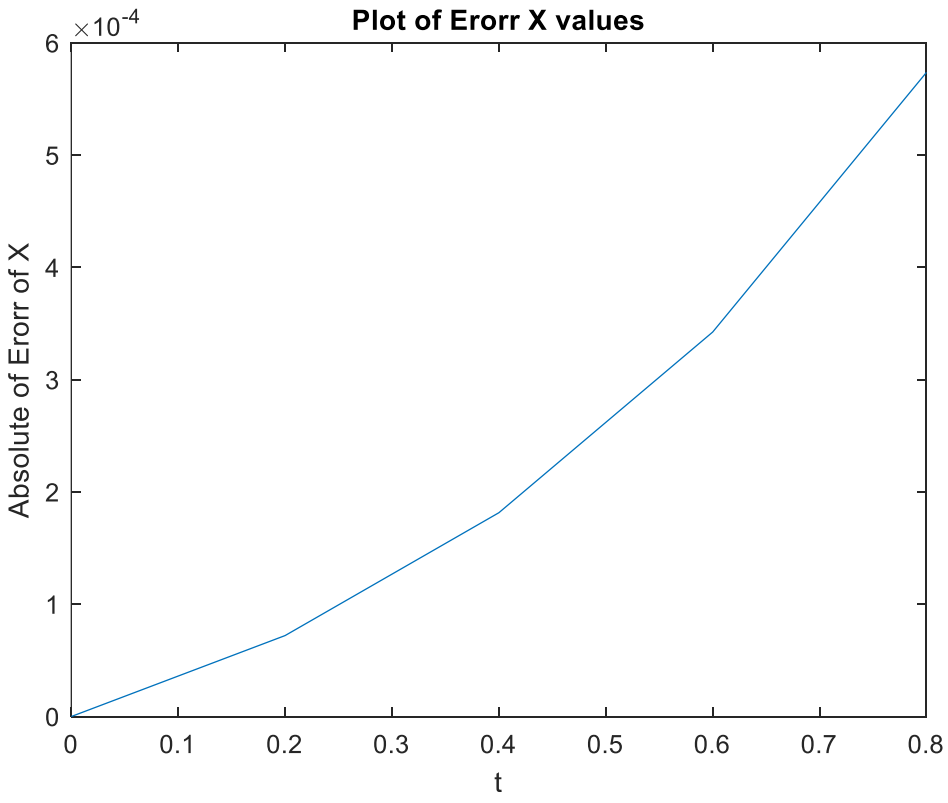


Figure 3.1(b): Shows the absolute error resulting of$x(t)$.

Table 3.2 (b), the exact and numerical solution for $y(t)$ when applying RKM on system (3.1) and showing the resulting error.

Table 3.2(b)
The exact and numerical solution of $y(t)$ by applying RKM

t	Exact solution	Approximate solution	Absolute error
0	2.00000000	2.000000000	0.000000000
0.2	3.419927723	3.419866666	0.000061057
0.4	5.370568912	5.370414373	0.000154539
0.6	8.017322722	8.017029717	0.000293005
0.8	11.57281283	11.57231957	0.00049326

From table 3.2(b) we conclude the maximum error $= 0.00049326$.

Figure 3.2(a): Compares the exact solution $y(t)$ and approximate solution.

Figure 3.2(a)
The exact and numerical solutions of $y(t)$

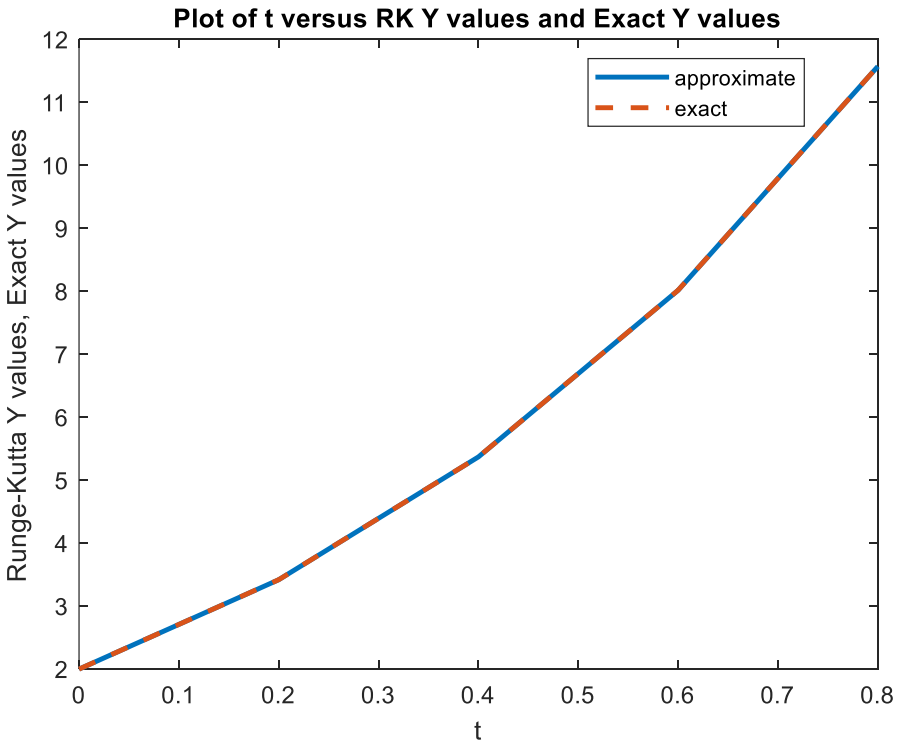


Figure 3.2(b)
absolute error resulting of applying RKM of $y(t)$

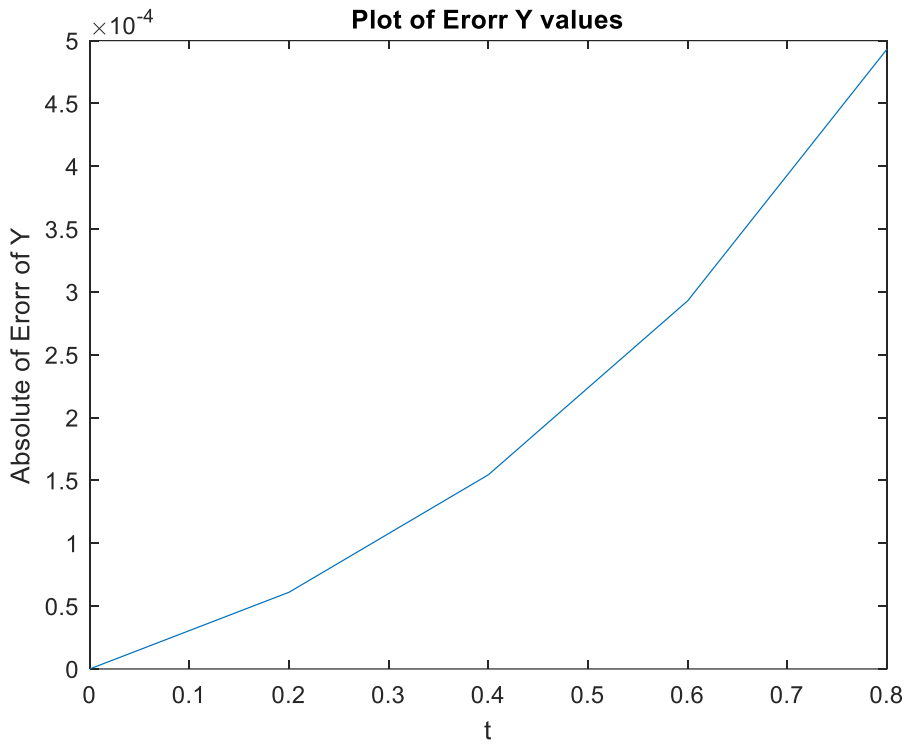


Figure 3.2(b): Shows the absolute error resulting of $y(t)$..

3.2 The numerical approximation of system (3.1) using the Adams-Bashforth Method

The following algorithm implements the Adams-Bashforth Method using the matlab software.

Algorithm 3.2

An algorithm for the Fourth-order Adams Bashforth method is given below

Set $h = \frac{a-b}{n}$;

$$t_0 = a;$$

$$x'_n = f(t_n, x_n);$$

$$x'_{n-1} = f(t_{n-1}, x_{n-1});$$

$$x'_{n-2} = f(t_{n-2}, x_{n-2});$$

$$x'_{n-3} = f(t_{n-3}, x_{n-3});$$

$$x_{n+1}^{*} = x_n + \frac{h}{24}(55x_n' - 59x_{n-1}' + 37x_{n-2}' - 9x_{n-3}');$$

$$t_{k+1} = a + \frac{k+1}{h};$$

we repeated the code for$k = 0, 1, \dots , n-1$.

After these steps the value x_k^{*} is an approximated value of solution $x(t_k)$ of the differential at t_k.

The Adams-Bashforth Method to approximate $x(0.8)$ and $y(0.8)$ with $h = 0.2$ in the system (3.1).

$$x_0' = f(t_0 , x_0 , y_0) = f(0 , 6, 2) = 10$$

$$y_0' = g(t_0 , x_0 , y_0) = g(0 , 6 , 2) = 6$$

$$x_1' = f(t_1 , x_1 , y_1) = f(0.2 , 8.30546666, 3.419866666) = 13.19106665$$

$$y_1' = g(t_1 , x_1 , y_1) = g(0.2 , 8.30546666, 3.419866666) = 8.30546666$$

$$x_2' = f(t_2 , x_2 , y_2) = f(0.4 , 11.33768621, 5.370414373) = 17.30495805$$

$$y_2' = g(t_2 , x_2 , y_2) = g(0.4 , 11.33768621, 5.370414373) = 11.33768621$$

$$x_3' = f(t_3 , x_3 , y_3) = f(0.6 , 15.30545554 , 8.017029717) = 22.59388136$$

$$y_3' = g(t_3 , x_3 , y_3) = g(0.6 , 15.30545554 , 8.017029717) = 15.30545554$$

$$x_4^{*} = x_3 + \frac{h}{24}(55x_3' - 59x_2' + 37x_1' - 9x_0')$$

$$= 15.30545554 + \frac{0.2}{24}(55(22.59388136) - 59(17.30495805) + 37(13.19106665) - 9(10))$$

$$= 15.30545554 + \frac{0.2}{24}(619.7404164)$$

$$= 15.30545554 + 5.16450347 = 20.46995901$$

$$y_4^* = y_3 + \frac{h}{24}(55y_3' - 59y_2' + 37y_1' - 9y_0')$$

$$= 8.017029717 + \frac{0.2}{24}(55(15.30545554) - 59(11.33768621) + 37(8.30546666) - 9(6))$$

$$= 8.017029717 + \frac{0.2}{24}(426.1788347)$$

$$= 8.017029717 + 3.551490289 = 11.56852001$$

Table 3.3(a), shows the exact and numerical solutions when using ABM on system (3.1) and showing the resulting error.

Table 3.3(a)
The exact and numerical solution of applying Fourth Adams-Bashforth method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	20.47497654	20.46995901	0.00501753
$y(t)$	11.57281283	11.56852001	0.00429282

These results show the ABM is accuracy with exact solution.

3.3 The numerical approximation of system (3.1) using the Adams-Moulton method

The following algorithm implements the Adams-Moulton methodusing the matlab software.

Algorithm 3.3

An algorithm for the Fourth-order Adams-Moulton method is given below

Set $h = \frac{a-b}{n}$;

$$x_{n+1}' = f(t_{n+1}, x_{n+1}^*);$$

$$x_{n+1} = x_n + \frac{h}{24}(9x_{n+1}' + 19x_n' - 5x_{n-1}' + x_{n-2}');$$

$$t_{k+1} = a + \frac{k+1}{h};$$

we repeated the code for$k = 0, 1, \ldots, n-1$.

After these steps the value x_k is an approximated value of solution $x(t_k)$ of the differential at t_k.

Consider the RK4 method with $x_3 = 15.30545554$, $y_3 = 8.017029717$ and $t_4 = 0.8$at system (3.1) we have

$$x_4' = f(t_4\,, x_4^*\,, y_4^*) = f(0.8\,, 20.46995901\,, 11.56852001) = 29.37139801$$

$$y_4' = g(t_4\,, x_4^*\,, y_4^*) = g(0.8\,, 20.46995901\,, 11.56852001) = 20.46995901$$

now

$$y_{n+1} = y_n + \frac{h}{24}(\,9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

$$x_4 = x_3 + \frac{h}{24}(\,9x_4' + 19x_3' - 5x_2' + x_1')$$

$$= 15.30545554 \;\; + \frac{0.2}{24}\;(\,9(29.37139801) + 19(22.59388136)$$
$$- 5(17.30495805) + (13.19106665))$$

$$= 15.30545554 \;\; + \frac{0.2}{24}(620.2926044)$$

$$= 15.30545554 \;\; + 5.169105037$$

$$x_4 = 20.47456058$$

$$y_4 = y_3 + \frac{h}{24}(\,9y_4' + 19y_3' - 5y_2' + y_1')$$

$$= 8.017029717 + \frac{0.2}{24}(9(20.46995901) + 19(15.30545554)$$
$$- 5(11.33768621) + (8.30546666))$$

$$= 8.017029717 + \frac{0.2}{24}(426.650322)$$

$$= 8.017029717 + 3.55541935$$

$$y_4 \;\; = 11.57244907$$

Table 3.3(b), the exact and numerical solution when applying AMM on system (3.1) and showing the resulting error.

Table 3.3(b)
The exact and numerical solution of applying Fourth Adams-Moulton method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	20.47497654	20.47456058	0.00041596
$y(t)$	11.57281283	11.57244907	0.00036376

These results show the AMM is accuracy with exact solution.

3.4 The numerical approximation of system (3.1) using the Predictor-Corrector method

The following algorithm implements the Predictor-Corrector method using the matlab software.

Algorithm 3.4

$$\text{Set } h = \tfrac{a-b}{n};$$

$$\hat{x_{n+1}} = x_n + h\, f(t_n, x_n);$$

$$x_{n+1} = x_n + \frac{h}{2}\Big(f(t_n, x_n) + f\big(t_{n+1}, \hat{x_{n+1}}\big)\Big);$$

$$t_{k+1} = a + \frac{k+1}{h};$$

we repeated the code for$k = 0,1, \dots, n-1$.

After these steps the value x_k is an approximated value of solution $x(t_k)$ of the differential at t_k.

Consider the RK4 method with $x_3 = 15.30545554$, $y_3 = 8.017029717$ and $t_4 = 0.8$ at system (3.1) we have

$$\hat{x_{n+1}} = x_n + h\, f(t_n, x_n, y_n)$$

$$x_{n+1} = x_n + \frac{h}{2}\Big(f(t_n, x_n, y_n) + f\big(t_{n+1}, \hat{x_{n+1}}, \hat{y_{n+1}}\big)\Big)$$

now

$$\hat{x_4} = x_3 + h\, f(t_3\,, x_3\,, y_3)$$

$$= 15.30545554 + 0.2\, f(0.6\,, 15.30545554\,, 8.017029717)$$

$$= 15.30545554 + 0.2\,(22.59388136)$$

$$= 15.30545554 + 4.518776272 = 19.82423181$$

$$\hat{y_4} = y_3 + h\, g(t_3\,, x_3\,, y_3)$$

$$= 8.017029717 + 0.2\, g(0.6\,, 15.30545554\,, 8.017029717)$$

$$= 8.017029717 + 0.2\,(15.30545554)$$

$$= 8.017029717 + 3.061091108 = 11.07812083$$

$$x_4 = x_3 + \frac{h}{2}\Big(\, f(t_3\,, x_3\,, y_3) + f\big(t_4\,, \hat{x_4}\,, \hat{y_4}\big)\Big)$$

$$= 15.30545554 + \frac{0.2}{2}(f(0.6\,, 15.30545554\,, 8.017029717)$$
$$+ f(0.8\,, 19.82423181\,, 11.07812083\,))$$

$$= 15.30545554 + \frac{0.2}{2}(\,22.59388136 + 28.57034279)$$

$$= 15.30545554 + 0.1(\,51.16422415)$$

$$= 20.42187796$$

$$y_4 = y_3 + \frac{h}{2}\Big(\, g(t_3\,, x_3\,, y_3) + g\big(t_4\,, \hat{x_4}\,, \hat{y_4}\big)\Big)$$

$$= 8.017029717 + \frac{0.2}{2}(g(0.6\,, 15.30545554\,, 8.017029717)$$
$$+ g(0.8\,, 19.82423181\,, 11.07812083\,))$$

$$= 8.017029717 + 0.1(15.30545554 + 19.82423181)$$

$$= 8.017029717 + 0.1(\,35.12968735)$$

$$= 11.52999845$$

Table 3.3(c), the exact and numerical solution when applying PCM on system (3.1) and showing the resulting error.

Table3.3(c)
The exact and numerical solution of applying Predictor-Corrector method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	20.47497654	20.42187796	0.05309858
$y(t)$	11.57281283	11.52999845	0.04281438

these results show the PCM is accuracy with exact solution.

Example 3.2

Consider the system of linear differential equations

$$\frac{dx}{dt} = 3x - y - z \tag{3.3}$$

$$\frac{dy}{dt} = x + y - z$$

$$\frac{dz}{dt} = x - y + z$$

subject to the initial conditions

$$x(0) = 6 \tag{3.4}$$

$$y(0) = 3$$

$$z(0) = 4$$

to find the exact solution of System (3.3),we have $\det(A - \lambda I) = (1 - \lambda)(\lambda - 2)^2 = 0$. For $\lambda_1 = 1$ we obtain

$$K_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

for $\lambda_2 = 2$ we obtain

$$K_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ and } K_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

then

$$X(t) = c_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} e^t + c_2 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} e^{2t} + c_3 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} e^{2t}.$$

The exact solution of system (3.3) is

$$x(t) = e^t + 5\, e^{2t}$$

$$y(t) = e^t + 2\, e^{2t}$$

$$z(t) = e^t + 3\, e^{2t}$$

3.5 The numerical approximation of system (3.3) using the fourth order Rung Kutta methods

Table 3.4(a), 3.4(b) and 3.4(c), contain the k's value when applying RKM for system 3.3

Table 3.4(a)
The k's value for $x(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	2.200000000	2.620000000	2.702000000	3.236400000
0.4	3.227746666	3.848868000	3.970649466	4.761776293
0.6	4.748900267	5.668843961	5.849849064	7.022603176
0.8	7.003435201	8.367680112	8.636884881	10.37728762

Table 3.4(b)
The k's value for $y(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	1.000000000	1.180000000	1.214000000	1.441200000
0.4	1.437666666	1.700772000	1.750950266	2.083816613
0.6	2.078578262	2.464457555	2.538649778	3.027801456
0.8	3.020026855	3.587590097	3.697458532	4.418108741

Table 3.4(c)
The k's value for $z(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	2.200000000	2.620000000	2.702000000	3.236400000
0.4	3.227746666	3.848868000	3.970649466	4.761776293
0.6	4.748900267	5.668843961	5.849849064	7.022603176
0.8	7.003435201	8.367680112	8.636884881	10.37728762

Table 3.5(a), -Appendix D- the exact and numerical solution for $x(t)$ when applying RKM on system (3.3) and showing the resulting error.

From table 3.5(a) we conclude the maximum error $= 0.006086350$.

Figure 3.3(a): Compares the exact solution $x(t)$ and approximate solution with time t .

Figure 3.3(a)
The exact and numerical solutions of $x(t)$

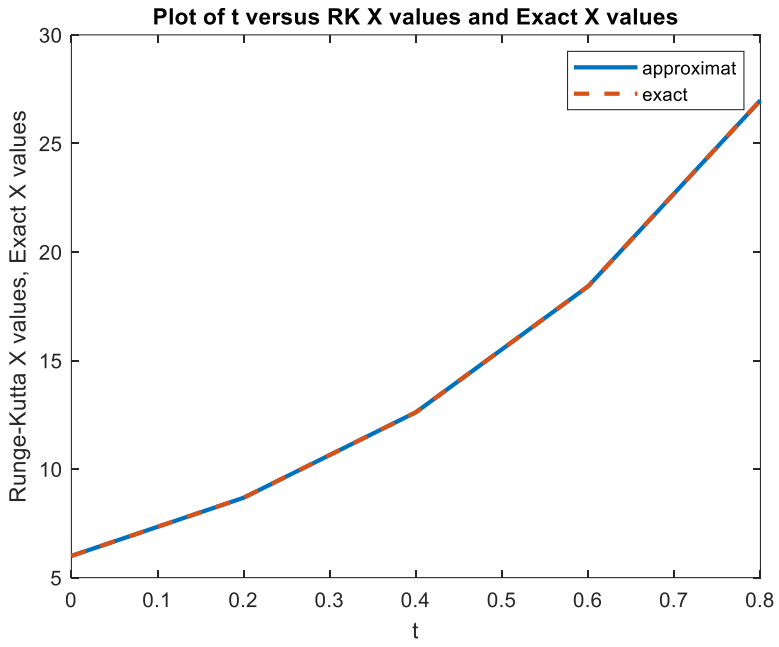


Figure 3.3(b)
the absolute error resulting of applying RKM.

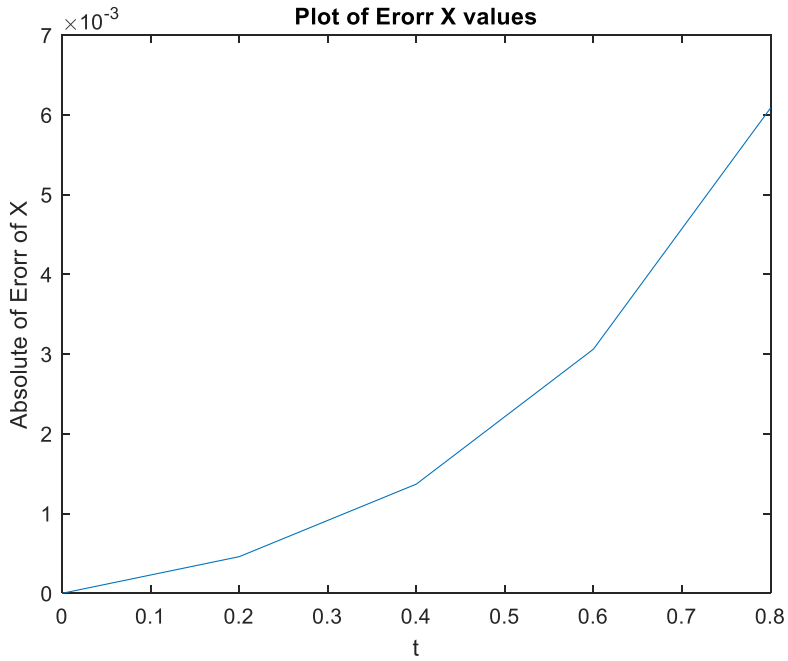


Figure 3.3(b): Shows the absolute error resulting of system 3.3.

Table 3.5(b), -Appendix D- the exact and numerical solution for $y(t)$ when applying RKM on system (3.3) and showing the resulting error.

From table 3.5(b) we conclude the maximum error $= 0.002446607$

Figure 3.4(a): Compares the exact solution $y(t)$ and approximate solution with time t .

Figure 3.4(a)
The exact and numerical solutions $y(t)$

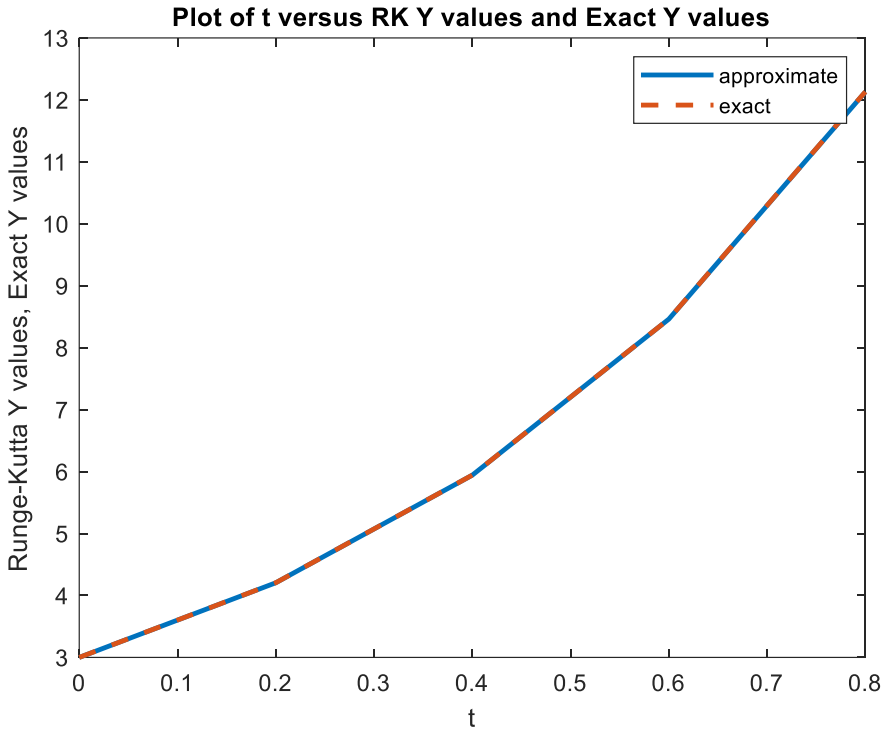


Figure 3.4(b)
the absolute error resulting of applying RKM of $y(t)$

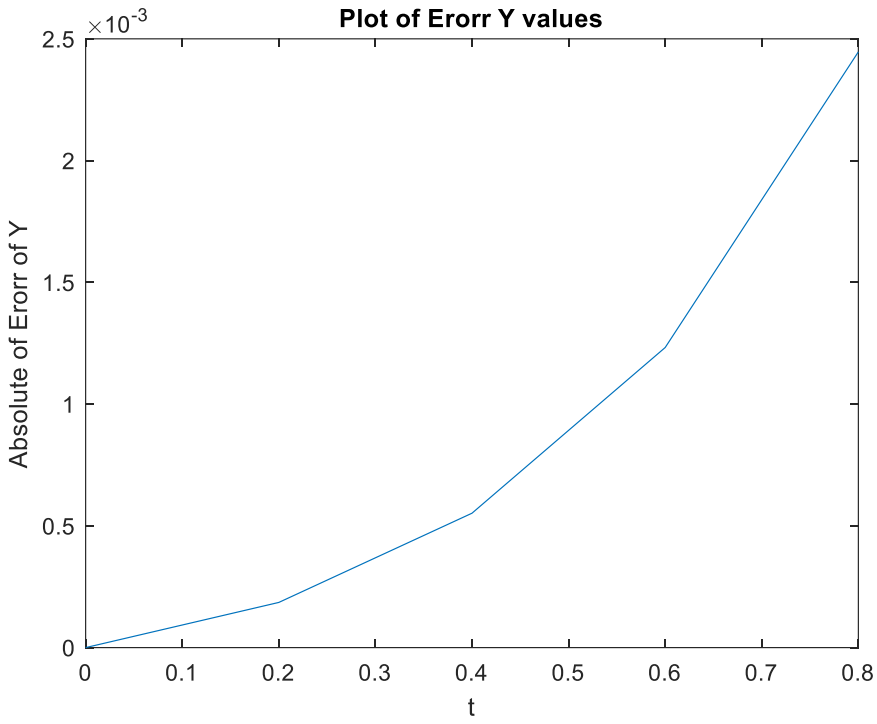


Figure 3.4(b): Shows the absolute error resulting of $y(t)$.

Table 3.5(c), -Appendix D- the exact and numerical solution for $z(t)$ when applying RKM on system (3.3) and showing the resulting error.

From table 3.5(c) we conclude the maximum error $= 0.003659851$

Figure 3.5(a): -Appendix E- Compares the exact solution $z(t)$ and approximate solution with time t .

Figure 3.5(b): -Appendix E- Shows the absolute error resulting of $z(t)$.

3.6 The numerical approximation of system (3.3) using the Fourth Adams-Bashforth Method

The Adams-Bashforth Method to approximate $x(0.8)$, $y(0.8)$ and $z(0.8)$with $h = 0.2$ in the system (3.3).

$$x_0' = f(t_0\,, x_0\,, y_0\,, z_0) = f(\,0\,, 6\,, 3\,, 4\,) = 11$$

$$y_0' = g(t_0\,, x_0\,, y_0\,, z_0) = g(\,0\,, 6\,, 3\,, 4\,) = 5$$

$$z_0' = q(t_0\,, x_0\,, y_0\,, z_0) = q(\,0\,, 6\,, 3\,, 4\,) = 7$$

$$x_1' = f(t_1\,, x_1\,, y_1\,, z_1) = f(\,0.2\,, 8.680066666, 4.504866666, 5.6966)$$
$$= 16.13873331$$

$$y_1' = g(t_1\,, x_1\,, y_1\,, z_1) = g(0.2\,, 8.680066666, 4.504866666, 5.6966)$$
$$= 7.188333326$$

$$z_1' = q(t_1\,, x_1\,, y_1\,, z_1) = q(0.2\,, 8.680066666, 4.504866666, 5.6966)$$
$$= 10.17179999$$

$$x_2' = f(t_2\,, x_2\,, y_2\,, z_2)$$

$$= f(\,0.4\,, 12.61815964, 5.942354635, 8.167622973) = 2374450131$$

$$y_2' = g(t_2\,, x_2\,, y_2\,, z_2) = g(\;0.4\,, 12.61815964, 5.942354635, 8.167622973)$$
$$= 10.3928913$$

$$z_2' = q(t_2\,, x_2\,, y_2\,, z_2)$$

$$= q(\,0.4\,, 12.61815964, 5.942354635, 8.167622973) = 14.84342798$$

$$x_3' = f(t_3\,, x_3\,, y_3\,, z_3)$$

$$= f(\,0.6, 18.41964123, 8.461120366, 11.78062732) = 35.017176$$

$$y_3' = g(t_3\,, x_3\,, y_3\,, z_3)$$

$$= g(\,0.6, 18.41964123, 8.461120366, 11.78062732) = 15.10013428$$

$$z_3' = q(t_3\,, x_3\,, y_3\,, z_3)$$

$$= q(\,0.6, 18.41964123, 8.461120366, 11.78062732) = 21.73914818$$

$$x_4^* = x_3 + \frac{h}{24}(\,55x_3' - 59x_2' + 37x_1' - 9x_0')$$

$$= 18.41964123\; + \frac{0.2}{24}(55(35.017176\,) - 59(23.74450131)$$
$$+ 37(16.13873331\,) - 9(11))$$

$$= 18.41964123 + \frac{0.2}{24}(1023.152235)$$

$$= 18.41964123 + 8.526268625 = 26.94590986$$

$$y_4^* = y_3 + \frac{h}{24}(55y_3' - 59y_2' + 37y_1' - 9y_0')$$

$$= 8.461120366 + \frac{0.2}{24}(55(15.10013428) - 59(10.3928913) + 37(7.188333326) - 9\,(5))$$

$$= 8.461120366 + \frac{0.2}{24}(438.2951314)$$

$$= 8.461120366 + 3.652459428 = 12.11357979$$

$$z_4^* = z_3 + \frac{h}{24}(55z_3' - 59z_2' + 37z_1' - 9z_0')$$

$$= 11.78062732 + \frac{0.2}{24}(55(21.73914818) - 59(14.84342798) + 37(10.17179999) - 9(7))$$

$$= 11.78062732 + \frac{0.2}{24}(633.2474992)$$

$$= 11.78062732 + 5.277062493 = 17.05768981$$

Table 3.6(a), -Appendix D- the exact and numerical solution when applying ABM on system (3.3) and showing the resulting error.

These results show the ABM is accuracy with exact solution.

3.7 The numerical approximation of system (3.3) using the Adams-Moulton method

Consider the RK4 method with $x_3 = 18.41964123$, $y_3 = 8.461120366$, $z_3 = 11.78062732$and $t_4 = 0.8$ at system (3.3) we have

$x_4' = f(t_4\,, x_4^*\,, y_4^*\,, z_4^*)$

$$= f(0.8\,, 26.94590986\,, 12.11357979\,, 17.05768981) = 51.66645998$$

$$y_4' = g(t_4\,, x_4^*\,, y_4^*\,, z_4^*)$$

$$= g(0.8\,, 26.94590986\,, 12.11357979\,, 17.05768981) = 22.00179984$$

$$z_4' = q(t_4\,, x_4^*\,, y_4^*\,, z_4^*)$$

$$= q(0.8\,, 26.94590986\,, 12.11357979\,, 17.05768981) = 31.89001988$$

now

$$x_{n+1} = x_n + \frac{h}{24}(\,9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

$$x_4 = x_3 + \frac{h}{24}(\,9x_4' + 19x_3' - 5x_2' + x_1')$$

$$x_4 = 18.41964123 + \frac{0.2}{24}(9(51.66645998) + 19(35.017176) - 5(23.74450131) + (16.13873331))$$

$$= 18.41964123 + \frac{0.2}{24}(1027.74071)$$

$$= 18.41964123 + 8.564505917$$

$$x_4 = 26.98414715$$

$$y_4 = y_3 + \frac{h}{24}(\,9y_4' + 19y_3' - 5y_2' + y_1')$$

$$y_4 = 8.461120366 + \frac{0.2}{24}(9(22.00179984) + 19(15.10013428) - 5(10.3928913) + (7.188333326))$$

$$= 8.461120366 + \frac{0.2}{24}(440.142627)$$

$$= 8.461120366 + 3.667855225$$

$$y_4 = 12.12897559$$

$$z_4 = z_3 + \frac{h}{24}(\,9z_4' + 19z_3' - 5z_2' + z_1')$$

$$z_4 = 11.78062732 + \frac{0.2}{24}(9(31.89001988) + 19(21.73914818) - 5(14.84342798) + (10.17179999))$$

$$= 11.78062732 + \frac{0.2}{24}(636.00865)$$

$$= 11.78062732 + 5.300072083$$

$$z_4 = 17.0806994$$

Table 3.6 (b), -Appendix D- the exact and numerical solution when applying AMM on system (3.3) and showing the resulting error.

These results show the AMM is accuracy with exact solution.

3.8 The numerical approximation of system (3.3) using the Predictor-Corrector method

$$x_4 = x_3 + \frac{h}{2}\Big(f(t_3, x_3, y_3) + f\big(t_4, \hat{x_4}, \hat{y_4}\big)\Big)$$

$$\hat{x_4} = x_3 + h\, f(t_3, x_3, y_3, z_3)$$

$$= 18.41964123 + 0.2\, f(0.6, 18.41964123, 8.461120366, 11.78062732)$$

$$= 18.41964123 + 0.2\,(35.017176)$$

$$= 18.41964123 + 7.0034352 = 25.42307643$$

$$\hat{y_4} = y_3 + h\, g(t_3, x_3, y_3, z_3)$$

$$= 8.461120366 + 0.2\, g(0.6, 18.41964123, 8.461120366, 11.78062732)$$

$$= 8.461120366 + 0.2\,(15.10013428)$$

$$= 8.461120366 + 3.020026856 = 11.48114722$$

$$\hat{z_4} = z_3 + h\, q(t_3, x_3, y_3, z_3)$$

$$= 11.78062732 + 0.2\, q(0.6, 18.41964123, 8.461120366, 11.78062732)$$

$$= 11.78062732 + 0.2\ (21.73914818)$$

$$= 11.78062732 + 4.347829636 = 16.12845696$$

$$x_4 = x_3 + \frac{h}{2}\Big(\ f(t_3\ , x_3\ , y_3, z_3) + f\big(t_4\ , \hat{x_4}\ , \hat{y_4}, \hat{z_4}\big)\Big)$$

$$x_4 = 18.41964123 + \frac{0.2}{2}\big(f(0.6, 18.41964123, 8.461120366, 11.78062732) + f(0.8\ , 25.42307643\ , 11.48114722\ , 16.12845696\ )\big)$$

$$= 18.41964123\ + \frac{0.2}{2}(35.017176 + 48.65962511)$$

$$= 18.41964123 + 0.1(83.67680111)$$

$$= 18.41964123 + 8.367680111$$

$$=\ 26.78732134$$

$$y_4 = y_3 + \frac{h}{2}\Big(\ g(t_3\ , x_3\ , y_3, z_3) + g\big(t_4\ , \hat{x_4}\ , \hat{y_4}, \hat{z_4}\big)\Big)$$

$$y_4 = 8.461120366 + \frac{0.2}{2}\big(g(0.6, 18.41964123, 8.461120366, 11.78062732) + g(0.8\ , 25.42307643\ , 11.48114722\ , 16.12845696\ )\big)$$

$$= 8.461120366 + 0.1(15.10013428 + 20.77576669)$$

$$= 8.461120366 + 0.1(35.87590097)$$

$$= 8.461120366 + 3.587590097$$

$$= 12.04871046$$

$$z_4 = 11.78062732 + \frac{0.2}{2}\big(q(0.6, 18.41964123, 8.461120366, 11.78062732) + q\ (0.8\ , 25.42307643\ , 11.48114722\ , 16.12845696\ )\big)$$

$$= 11.78062732 + 0.1(21.73914818 + 30.07038617)$$

$$= 11.78062732 + 0.1(51.80953435)$$

$$= 11.78062732 + 5.180953435$$

$$= 16.96158076$$

Table 3.6(c), -Appendix D- the exact and numerical solution when applying PCM on system (3.3) and showing the resulting error.

These results show the PCM is accuracy with exact solution.

Example 3.3:

Consider the system of linear differential equations

$$\frac{dx}{dt} = 3x - y + 4e^{2t} \tag{3.5}$$

$$\frac{dy}{dt} = -x + 3y + 4e^{4t}$$

subject to the initial conditions

$$x(0) = 1 \tag{3.6}$$

$$y(0) = 1$$

To find the exact solution of system (3.5), we have $\lambda = 2\,, 4$

we obtain $\Phi(t) = \begin{bmatrix} -e^{4t} & e^{2t} \\ e^{4t} & e^{2t} \end{bmatrix}$, $\Phi^{-1}(t) = \begin{bmatrix} -\frac{1}{2}e^{-4t} & \frac{1}{2}e^{-4t} \\ \frac{1}{2}e^{-2t} & \frac{1}{2}e^{-2t} \end{bmatrix}$

$$X = \Phi\Phi^{-1}(0)X(0) + \Phi\int_0^t \Phi^{-1}F\,ds$$

$$= \Phi.\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \Phi.\begin{pmatrix} e^{-2t} + 2t - 1 \\ e^{2t} + 2t - 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2 \\ 2 \end{pmatrix} te^{2t} + \begin{pmatrix} -1 \\ 1 \end{pmatrix} e^{2t} + \begin{pmatrix} -2 \\ 2 \end{pmatrix} te^{4t} + \begin{pmatrix} 2 \\ 0 \end{pmatrix} e^{4t}$$

The exact solution to the system (3.5) is

$$x(t) = 2te^{2t} - e^{2t} - 2te^{4t} + 2e^{4t}$$

$$y(t) = 2te^{2t} + e^{2t} + 2te^{4t}$$

3.9 The numerical approximation of system (3.5) using the fourth order Rung Kutta methods

The fourth order Runge-Kutta method to approximate $x(0.8)$ and $y(0.8)$ with $h = 0.2$ in the system (3.5).

Table 3.7 (a) and 3.7 (b), -Appendix D- contain the k's value when applying algorithm 3.1 on system 3.5

Table 3.8(a), -Appendix D- the exact and numerical solution for $x(t)$ when applying RKM on system (3.5) and showing the resulting error.

From table 3.8(a) we conclude the maximum error $= 0.094200336$

Figure 3.6(a): -Appendix E- Compares the exact solution $x(t)$ and approximate solution with time t .

Figure 3.6(b): -Appendix E- Shows the absolute error resulting of $x(t)$.

Table 3.8(b), -Appendix D- the exact and numerical solution for $y(t)$ when applying RKM on system (3.5) and showing the resulting error.

From table 3.8(b) we conclude the maximum error $= 0.119853838$

Figure 3.7(a): -Appendix E- Compares the exact solution $y(t)$ and approximate solution with time t.

Figure 3.7(b): -Appendix E- Shows the absolute error resulting of $y(t)$.

3.10 The numerical approximation of system (3.5) using the Fourth Adams-Bashforth Method

The Adams-Bashforth Method to approximate $x(0.8)$ and $y(0.8)$ with $h = 0.2$ in the system (3.5).

$$x_0' = f(t_0\,, x_0\,, y_0) = f(\,0\,, 1\,, 1\,) = 6$$

$$y_0' = g(t_0\,, x_0\,, y_0) = g(\,0\,, 1\,, 1) = 6$$

$$x_1' = f(t_1\,, x_1\,, y_1) = f(0.2\,, 2.666086766, 2.977368648) = 10.98819044$$

$$y_1' = g(t_1\,, x_1\,, y_1) = g(0.2\,, 2.666086766, 2.977368648) = 15.16818289$$

$$x_2' = f(t_2\,, x_2\,, y_2) = f(0.4, 5.502624064, 7.960421513) = 17.44961439$$

$$y_2' = g(t_2\,, x_2\,, y_2) = g(0.4, 5.502624064, 7.960421513) = 38.19077018$$

$$x_3' = f(t_3\,, x_3\,, y_3) = f(0.6\,, 9.505252676, 20.49885426) = 21.29737145$$

$$y_3' = g(t_3\,, x_3\,, y_3) = g(0.6\,, 9.505252676, 20.49885426) = 96.0841564$$

$$x_4^* = x_3 + \frac{h}{24}(\,55x_3' - 59x_2' + 37x_1' - 9x_0')$$

$$= 9.505252676 + \frac{0.2}{24}(55(\,21.29737145\,) - 59(17.44961439) + 37(10.98819044) - 9(6))$$

$$= 9.505252676 + \frac{0.2}{24}(494.3912272)$$

$$= 9.505252676 + 4.119926893$$

$$= 13.62517957$$

$$y_4^* = y_3 + \frac{h}{24}(\,55y_3' - 59y_2' + 37y_1' - 9y_0')$$

$$= 20.49885426 + \frac{0.2}{24}(55(96.0841564) - 59(38.19077018) + 37(15.16818289) - 9(6))$$

$$= 20.49885426 + \frac{0.2}{24}(3538.588187)$$

$$= 20.49885426 \ - 29.48823489 = 49.98708915$$

Table 3.9(a), -Appendix D- the exact and numerical solution when applying ABM on system (3.5) and showing the resulting error.

These results show the ABM is accuracy with exact solution.

3.11 The numerical approximation of system (3.5) using the Adams-Moulton method

$$x_4' = f(t_4\,, x_4^*\,, y_4^*) = f(0.8\,, 13.62517957\,, 49.98708915) = 10.70057926$$

$$y_4' = g(t_4\,, x_4^*\,, y_4^*) = g(0.8\,, 13.62517957\,, 49.98708915) = 234.4662087$$

now

$$y_{n+1} = y_n + \frac{h}{24}(\,9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

$$x_4 = x_3 + \frac{h}{24}(\,9x_4' + 19x_3' - 5x_2' + x_1')$$

$$= 9.505252676 \ + \frac{0.2}{24}(\,9(10.70057926) + 19(21.29737145) - 5(17.44961439) + (10.98819044))$$

$$= 9.505252676 \ + \frac{0.2}{24}(424.6953894)$$

$$= 9.505252676 \ + 3.539128245$$

$$x_4 = 13.04438092$$

$$y_4 = y_3 + \frac{h}{24}(\,9y_4' + 19y_3' - 5y_2' + y_1')$$

$$= 20.49885426 + \frac{0.2}{24}(9(234.4662087) + 19(96.0841564) - 5(38.19077018) + (15.16818289))$$

$$= 20.49885426 + \frac{0.2}{24}(3760.006507)$$

$$= 20.49885426 + 31.33338756$$

$$y_4 = 51.83224182$$

Table 3.9(b), -Appendix D- the exact and numerical solution when applying AMM on system (3.5) and showing the resulting error.

These results show the AMM is accuracy with exact solution

3.12 The numerical approximation of system (3.5) using the Predictor-Corrector method

$$x_4 = x_3 + \frac{h}{2}\Big(f(t_3 , x_3 , y_3) + f\big(t_4 , \hat{x_4} , \hat{y_4}\big)\Big)$$

and

$$\hat{x_4} = x_3 + h\, f(t_3 , x_3 , y_3)$$

$$= 9.505252676 + 0.2\, f(0.6 , 9.505252676 , 20.49885426)$$

$$= 9.505252676 + 0.2(21.29737145)$$

$$= 9.505252676 + 0.391535305 = 13.76472697$$

$$\hat{y_4} = y_3 + h\, g(t_3 , x_3 , y_3)$$

$$= 20.49885426 + 0.2\, g(0.6 , 9.505252676 , 20.49885426)$$

$$= 20.49885426 + 0.2\ (96.0841564)$$

$$= 20.49885426 + 19.21683128 = 39.71568554$$

$$x_4 = x_3 + \frac{h}{2}\Big(f(t_3 , x_3 , y_3) + f\big(t_4 , \hat{x_4} , \hat{y_4}\big)\Big)$$

$$= 9.505252676 + \frac{0.2}{2}(f(0.6 , 9.505252676 , 20.49885426)$$
$$+ f(0.8 , 13.76472697 , 39.71568554))$$

$$= 9.505252676 \quad + \frac{0.2}{2}(21.29737145 + 21.39062507)$$

$$= 9.505252676 \quad + 0.1(42.68799652)$$

$$= 13.77405233$$

$$y_4 = y_3 + \frac{h}{2}\Big(g(t_3 , x_3 , y_3) + g\big(t_4 , \hat{x_4} , \hat{y_4}\big)\Big)$$

$$= 20.49885426 + \frac{0.2}{2}(g(0.6 , 9.505252676 \ , 20.49885426) + g(0.8 , 13.76472697 , 39.71568554))$$

$$= 20.49885426 + 0.1(96.0841564 + 203.5124504)$$

$$= 20.49885426 + 0.1(299.5966068)$$

$$= 50.45851494$$

Table 3.9(c), -Appendix D- the exact and numerical solution when applying PCM on system (3.5) and showing the resulting error.

These results show the PCM is accuracy with exact solution.

Chapter Four
Comparison of Numerical Methods

In this chapter, we will compare the proposed numerical methods in the examples with exact solution and show the conclusions.

In this chapter, we will carry some comparison between the proposed numerical methods, namely; Rung-Kutta Method, Adams-Bashforth Method, Adams-Moulton Method and Predictor-Corrector Method for solving the numerical examples presented in chapter three.

For example 3.1 the exact and numerical solutions when applying Rung-Kutta Method, Adams-Bashforth Method, Adams-Moulton Method and Predictor-Corrector Method for solving system (3.1) and the error associated with their numerical methods are shown in Table 4.1 -Appendix D-.

In fact, we see clearly that the Adams-Moulton Methods is more accurate with less absolute error than its counterparts. On the other hand, the Predictor-Corrector method requires less Cup-time than the other methods.

For example 3.2 the exact and numerical solutions when applying Rung-Kutta Method, Adams-Bashforth Method, Adams-Moulton Method and Predictor-Corrector Method for solving system (3.3) and the error associated with their numerical methods are shown in Table 4.2 -Appendix D-.

In fact, we see clearly that the Rung-Kutta Method is more accurate with less absolute error than its counterparts. On the other hand, the Predictor-Corrector method requires less Cup-time than the other methods.

For example 3.3 the exact and numerical solutions when applying Rung-Kutta Method, Adams-Bashforth Method, Adams-Moulton Method and Predictor-Corrector Method for solving system (3.5) and the error associated with their numerical methods are shown in Table 4.3 -Appendix D-.

In fact, we see clearly that the Rung-Kutta Method is more accurate with less absolute error than its counterparts. On the other hand, the Adams-Moulton Method requires less Cup-time than the other methods.

4.1 Conclusions

This work explores the numerical treatment of systems of ordinary differential equations (ODEs), which has wide rang of applications in mathematical physics, chemistry, biology, stereology, heat conduction, and engineering models.

We implemented four numerical methods namely; Rung-Kutta Method, Adams-Bashforth Method, Adams-Moulton Method and Predictor-Corrector Method to solve systems of ordinary differential equations.

Numerical results presented in Tables and Figures show clearly that the Rung-Kutta Method is one of the efficient method in comparison with its counterparts.

References

[1] G. Adomian, Applications of Nonlinear Stochastic Systems Theory to Physics. Kulwer Academic, Dordrecht, 1989.

[2] G. Adomian, A Review of The Decomposition Method in Applied Mathematics, J. Math. Anal. Appl. 135, 501-544, (1988).

[3] G. Adomian, Nonlinear Stochastic Operator Equations. Academic Press. 1986.

[4] G. Adomian, Solving Frontier Problems of Physics, The Decomposition Method. Kluwer Academic Publisher, Dordrecht, 1994.

[5] F. Bashforth and J. Adams, An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drop, Cambridge University Press, Cambridge, 1883.

[6] J. Biazar, E. Babolian and R. Islam, Solution of a System of Ordinary Differential Equations with Adomian Decomposition Method, Appl. Math. Comput., 147(3), 2004.

[7] J. Biazar, Solution of a System Integral-Differential Equations by Adomian Decomposition Method. Appl. Math. Comput. 168. 2005.

[8] W. Boyce, R. Diprima and D. Meade, Elementary Differential Equations and Boundary Value Problems, USA, 11th, (2017).

[9] R. Burden, J. Faires and A. Burden, Numerical Analysis, 10th edition, Cengage Learning, 2014.

[10] R. Burden and J. Faires, Numerical Methods, Brooks Cole, 2002.

[11] J. Butcher, Numerical Method for Ordinary Differential Equations, 3rd ed., Wiley, UK, 2016.

[12] J. Butcher, "Numerical Methods ForOrdinary Differential Equations in The 20thCentury", Journal of Computational and Applied Mathematics, 125, 2000.

[13] S. Goode and S. Annin, Differential Equations and Linear Algebra, California State University, Fullerton, 4th ed., (2015), ppl.580-598.

[14] H. Jafari, A. Borhanifar, and S. Karimi, New Solitary Wave Solution for the bad Boussinesq and good Boussinesq Equations, Numer. Methods Partial Differential Equations, 25(2009), pp.1231-1237.

[15] H. Jafari and V. Daftardar-Gejji, Revised Adomian Decomposition Method for Solving Systems of Ordinary and Fractional Differential Equations, Applied Mathematics and Computation 181 (2006) pp.598-608.

[16] H. Khalil, Nonlinear Systems, 2nd.ed, Michigan State University, 1996.

[17] A. Kharab and R. Guenther, An Introduction to Numerical Methods: A MATLAB Approach, 5th ed., Boca Raton, Florida, 2023.

[18] W. Kutta, "Beitragzur Naherungsweisen Integration Totaler Differentialgleichungen". Z. Math. Phys., 46, 1901.

[19] W. Milne, A note on the Numerical Integration of Differential Equations, J. Res. Nat. Bur. Standards, 43, 1949.

[20] F. Moulton, New Method in Exterior Ballistics, University of Chicago, Chicago, 1926.

[21] R. Nagle, B. Edward and A. Sinder, Fundamentals of Differential Equations, 9th ed., Boston, 2017, pp.(132-139).

[22] C. Nuno, J. Cristina and N. Joao, Ordinary Differential Equations with Singular Coefficients: An Intrinsic Formulation with Applications to the Euler Bernoulli Beam Equation. Journal of Dynamics and Differential Equations, 2020.

[23] K. Prem, Green's Functions and Linear Differential Equations, USA, 2011.

[24] C. Runge, "Über dic numerische Auflosung von Differentialgleichungen", Math. Ann., 46, 1895.

[25] M. Searcoid, Metric Spaces, University College Dublin, Springer, Ireland, 2006.

[26] E. Süli and D. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, 2003.

[27] E. Süli, Numerical Solution of Ordinary Differential Equations, Mathematical Institute, University of Oxford, 2022.

[28] G. Thomas, J. Hess, C. Heil, P. Bogackl and M. Weir, Thomas Calculus Early Transcendentals, 15th ed., Pearson, 2024.

[29] G. Ying-Qin, A Class of Stable Algorithms for Stiff Ordinary Differential Equation System, Journal of Advances in Applied Mathematics.Vol.5, No.1, 2020.

[30] L. Zheng and X. Zhang, Modeling and Analysis of Modern Fluid Problems, Academic Press, 2017, pp.13-15.

[31] D. Zill, A First Course Differential Equations with Modeling Applications, Loyola Marymount University, USA, 2024.

Appendices

Appendix A

Matlab code for example 3.1

$$x' = 2x - y\,, \quad x(0) = 6$$

$$y' = x \qquad , \quad y(0) = 2$$

The enters required to matlab code

Enter the desired step size h required for computation: 0.2

Enter the t0 IC: 0

Enter the x0 IC: 6

Enter the y0 IC: 2

```
% RUN WITH MATLAB2019a / LAST MODIFCATION: 30/7/2023

% LAST MODIFCATION: 17/8/2023 (NO 3D NEEDED) RESTORE LAST ITERATION

formatLONGG

% ALL THIS CODE HAS BEEN EXPORTED TO FILE: RKversion2 WITH ALL THE COMMENTS

clc, clear

%% USER INPUT

H= input('Enter the desired step size h required for computation: ');

% Identifications or Functions

f=@(t,x,y)(2*x-y);

g=@(t,x,y)(x);
```

```
% Exact Solution: Input by user (Example)

EXACT_X=[6,8.3055,11.3378,15.3057,20.4749];

EXACT_Y=[2,3.4199,5.3705,8.0173,11.5728];

% erorr Solution: Input by user (Example)

ERORR_X=[0,0.0001,0.0002,0.0003,0.0005];

ERORR_Y=[0,0.0001,0.0001,0.0003,0.0005];

% IC: Inital Conditions For t,x,y

t0 = input('Enter the t0 IC: ');

x0 = input('Enter the x0 IC: ');

y0 = input('Enter the y0 IC: ');

fprintf('\n')

tic

%%

% Independent Variables

t = [t0 0 0 0 0];

x = [x0 0 0 0 0]; %[x0 x(0.2) x(0.4) x(0.6) x(0.8)]

y = [y0 0 0 0 0];

k = [0 0 0 0;0 0 0 0]; % Initialization for k11,k12,k13,14,k21,k22,k23,k24

O = randi([0,0],8,4);

A=[];B=[];C=[];D=[]; O=[];

% A I   iteration k for x1
```

```
% B II  iteration k for x2

% C III iteration k for x3

% D IV  iteration k for x4

%% [ONE-STEP] RK4: Runge-Kutta Method - One-Step Method for Solving ODE
(Approximating ODE Solutions)

%% (RK4): First Iteration to Obtain x1 = x(0.2)(mathnotation), x(2) (Matlab) ------ ; y1
= y(0.2)(mathnotation), y(2) (Matlab)

% MATH-WISE: using i Variable for Iterations where each iteration (i) changes the

% notation (or index) of t,x,y in substitution

fori=1:4

% i stands for iteration I which is to calcaulte x1,x2,x3,x4 in RK Method

for c=1:4

for r=1:2

if mod(r,2) % odd then sub f

if c==4

    k(r,c)=H*f(t(i)+H,x(i)+k(r,c-1),y(i)+k(r+1,c-1));

end

if c==1

k(r,c)=H*f(t(i),x(i),y(i));

elseif c<4

    k(r,c)=H*f(t(i)+H/2,x(i)+k(1,c-1)/2,y(i)+k(r+1,c-1)/2);

end
```

```
else% even then sub g
if c==4
    k(r,c)=H*g(t(i)+H,x(i)+k(r-1,c-1),y(i)+k(r,c-1));
end
if c==1
k(r,c)=H*g(t(i),x(i),y(i));
elseif c<4
    k(r,c)=H*g(t(i)+H/2,x(i)+k(r-1,c-1)/2,y(i)+k(r,c-1)/2);
end
end% END-IF EVEN OR ODD CHECK FOR F(t,x,y) or G(t,x,y) SUB
end% for r
switchi
case 1
   A = k;
case 2
   B = k;
case 3
   C = k;
case 4
   D = k;
end
```

```
end%for c

t(i+1)=t(i)+H;  % time update equation

% CALCULATION

x(i+1)= x(i) + (1/6)*(k(1,1)+2*k(1,2)+2*k(1,3)+k(1,4));

y(i+1)= y(i) + (1/6)*(k(2,1)+2*k(2,2)+2*k(2,3)+k(2,4));

end

x4rk = x(5);

y4rk = y(5);

fprintf('ONE-STEP:[Runga-Kutta RK4]  @t=(0.8) x4=%f ,y4=%f \n',x4rk,y4rk)

toc

fprintf('\n')

O = [A;B;C;D];

% CALL OUT

x,y

%fprintf('\n')

%disp('_____')

%disp('column: t , k1 , k2 , k3 , k4 , x , exact , error')

%fprintf('\n')

disp('_____')

disp('column: t , k1 , k2 , k3 , k4 , x')

P = [t(1),0,0,0,0,x(1)
```

```
    t(2),A(1,1),A(1,2),A(1,3),A(1,4),x(2)

    t(3),B(1,1),B(1,2),B(1,3),B(1,4),x(3)

    t(4),C(1,1),C(1,2),C(1,3),C(1,4),x(4)

    t(5),D(1,1),D(1,2),D(1,3),D(1,4),x(5)];

%disp('t k1 k2 k3 k4 x')

disp(P)

disp('_____')

disp('column: t , k1 , k2 , k3 , k4 , y')

PP = [t(1),0,0,0,0,y(1)

     t(2),A(2,1),A(2,2),A(2,3),A(2,4),y(2)

     t(3),B(2,1),B(2,2),B(2,3),B(2,4),y(3)

     t(4),C(2,1),C(2,2),C(2,3),C(2,4),y(4)

     t(5),D(2,1),D(2,2),D(2,3),D(2,4),y(5)];

disp(PP)

plot(t,y,t,EXACT_Y)

title('Plot of t versus RK Y values and Exact Y values')

xlabel('t')

ylabel('Runge-Kutta Y values, Exact Y values')

figure

plot(t,x,t,EXACT_X)

title('Plot of t versus RK X values and Exact X values')
```

```
xlabel('t')

ylabel('Runge-Kutta X values, Exact X values')

%% [TWO-STEP] Adam-Bashforth Method

tic

x_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)

y_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)

% x0' y0'

x_prime(1)=f(t(1),x(1),y(1));

y_prime(1)=g(t(1),x(1),y(1));

% x1' y1'

x_prime(2)=f(t(2),x(2),y(2));

y_prime(2)=g(t(2),x(2),y(2));

% x2' y2'

x_prime(3)=f(t(3),x(3),y(3));

y_prime(3)=g(t(3),x(3),y(3));

% x3' y3'

x_prime(4)=f(t(4),x(4),y(4));

y_prime(4)=g(t(4),x(4),y(4));

% x4' y4' (calculated by formula)

x_prime(5)= x(4)+(H/24)*(55*x_prime(4)-59*x_prime(3)+37*x_prime(2)-
9*x_prime(1));
```

```
y_prime(5)= y(4)+(H/24)*(55*y_prime(4)-59*y_prime(3)+37*y_prime(2)-
9*y_prime(1));

x4STAR=x_prime(5);

y4STAR=y_prime(5);

% CALL OUT

x_prime;

y_prime;

fprintf('TWO-STEP:[Adam-Bashforth] @t=(0.8) x4=%f ,y4=%f
\n',x_prime(5),y_prime(5))

toc

fprintf('\n')

%% [TWO-STEP] Adam-Moulton Method

% MATH-WISE: with step sie of h=0.2, x(0.8), y(0.8) will be approxmiated by x4 and
y4

% we use the RK4 method with x3 and y3,and t4=0.8 by previous calculation

tic

% Array of X & Y Values for Adam-Moulton Method

x_AM = [x(1) x(2) x(3) x(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

y_AM = [y(1) y(2) y(3) y(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

% values of x in Adam-Moulton are the same of these RK4

%{

DOCUMENTATION NOTE:
```

```
Formula: yn+1 = yn + h/24 (9fn+1 19fn -5fn-1 + fn-2)

To calculate x4,

x4 = x3 + h/24 (9*x4'+19*x3'-5*x2'+x1')

where:

x3: from RK4

x3',x2',x1 from Adam-Bashforth (primes)

x4': PREPARATION X4 From Adam-Bashforth

To calculate y4,

y4 = y3 + h/24 (9*y4'+19*y3'-5*y2'+y1')

where:

y3: from RK4

y3',y2',y1 from Adam-Bashforth (primes)

y4' PREPARATION Y4 From Adam-Bashforth

%}

% PREPARATION for X4 and Y4 NEW PRIME FOR NEW SUBSTITUTION

X_NEW_PRIME = f(t(5),x4STAR,y4STAR);

% the new x4' prime for Adam-Moulton

Y_NEW_PRIME = g(t(5),x4STAR,y4STAR);

% the new y4' prime for Adam-Moulton

% NEW SUBSTITUTION

x_AM(5)= x(4)+(H/24)*(9*X_NEW_PRIME + 19*x_prime(4) -5*x_prime(3) +
x_prime(2));
```

```
y_AM(5)= y(4)+(H/24)*(9*Y_NEW_PRIME + 19*y_prime(4) -5*y_prime(3) +
y_prime(2));

% CALL OUT

x_AM;

y_AM;

fprintf('TWO-STEP:[Adam-Mouton]  @t=(0.8) x4=%f ,y4=%f \n',x_AM(5),y_AM(5))

toc

fprintf('\n')

%% [TWO-STEP] Predictor-Corrector Method

% MATH-WISE: use the predictor-corrector method to obtain an approximation value
of

% x(0.8) and y(0.8)  for the solution of the system (rk4) with h=0.2

tic

% Array of X & Y Values for Predictor-Corrector Method

X_PC = [x(1) x(2) x(3) x(4) 0];

Y_PC = [y(1) y(2) y(3) y(4) 0];

%{

DOCUMENTATION NOTE:

PREPARATION for this method:

x4^ (x4 HAT)

y4^ (y4 HAT)

WHERE:
```

```
x4^ = x3 + h f(t3,x3,y3)

VALUES FROM RK4

AND:

y4^ = y3 + h g(t3,x3,y3)

VALUES FROM RK4

x4 = x3 + (H/2)(f(t3,x3,y3)+f(t4,x4^,y4^))

AND:

y4 = y3 + (H/2)(g(t3,x3,y3)+g(t4,x4^,y4^))

f(t3,x3,y3) = PREDICTOR

f(t4,x4^,y4^)= CORRECTOR

%}

% PREPARATION for X4 and Y4 HAT

XHAT = x(4) + H*f(t(4),x(4),y(4));

YHAT = y(4) + H*g(t(4),x(4),y(4));

% Obtain Approx Value of x(0.8) and y(0.8)

X_PC(5) = x(4) + (H/2)*(f(t(4),x(4),y(4))+f(t(5),XHAT,YHAT));

Y_PC(5) = y(4) + (H/2)*(g(t(4),x(4),y(4))+g(t(5),XHAT,YHAT));

fprintf('TWO-STEP:[Predictor-Corrector Method]  @t=(0.8) x4=%f ,y4=%f
\n',X_PC(5),Y_PC(5))

toc

% CALL OUT

X_PC;
```

```
Y_PC;

%% COMPARISON PLOT

plot(t,ERORR_X)

title('Plot of Erorr X values')

xlabel('t')

ylabel('Absolute of Erorr of X')

figure

plot(t,ERORR_Y)

title('Plot of Erorr Y values')

xlabel('t')

ylabel('Absolute of Erorr of Y')

%% COMPARISON PLOT

plot(0.8,x_prime(5),'bx',0.8,x_AM(5),'r+',0.8,X_PC(5),'go',0.8,EXACT_X(5),'kd')

figure

plot(0.8,y_prime(5),'bx',0.8,y_AM(5),'r+',0.8,Y_PC(5),'go',0.8,EXACT_Y(5),'kd')
```

Appendix B

Matlab code for example 3.2

$$x' = 3x - y - z \quad , \quad x(0) = 6$$

$$y' = x + y - z \quad , \quad y(0) = 3$$

$$z' = x - y + z \quad , \quad z(0) = 4$$

The enters required to matlab code

Enter the desired step size h required for computation: 0.2

Enter the t0 IC: 0

Enter the x0 IC: 6

Enter the y0 IC: 3

Enter the z0 IC: 4

```
% RUN WITH MATLAB2019a / LAST MODIFCATION: 30/7/2023

% LAST MODIFCATION: 17/8/2023 (NO 3D NEEDED) RESTORE LAST ITERATION

formatLONGG

% ALL THIS CODE HAS BEEN EXPORTED TO FILE: RKversion2 WITH ALL THE COMMENTS

clc, clear

%% USER INPUT

H= input('Enter the desired step size h required for computation: ');

% Identifications or Functions

f=@(t,x,y,z)(3*x-y-z);
```

```
g=@(t,x,y,z)(x+y-z);

m=@(t,x,y,z)(x-y+z);

% Exact Solution: Input by user (Example)

EXACT_X=[6,8.680526246,12.61952934,18.42270341,26.99070305];

EXACT_Y=[3,4.205052153,5.942906555,8.462352646,12.13160578];

EXACT_Z=[4,5.696876851,8.168447483,11.78246957,17.0846382];

% erorr Solution: Input by user (Example)

ERORR_X=[0,0.00045958,0.0013697,0.003062184,0.00608635];

ERORR_Y=[0,0.000185487,0.000551919,0.001232279,0.002446607];

ERORR_Z=[0,0.000276851,0.00082451,0.001842248,0.003659851];

% IC: Inital Conditions For t,x,y

t0 = input('Enter the t0 IC: ');

x0 = input('Enter the x0 IC: ');

y0 = input('Enter the y0 IC: ');

z0 = input('Enter the z0 IC: ');

fprintf('\n')

tic

%%

% Independent Variables

t = [t0 0 0 0 0];

x = [x0 0 0 0 0]; %[x0 x(0.2) x(0.4) x(0.6) x(0.8)]
```

```
y = [y0 0 0 0 0];

z = [z0 0 0 0 0];

k = [0 0 0 0;0 0 0 0;0 0 0 0]; % Initialization for k11,k12,k13,14,k21,k22,k23,k24

O = randi([0,0],8,4);

A=[];B=[];C=[];D=[]; O=[];

% A I   iteration k for x1

% B II  iteration k for x2

% C III iteration k for x3

% D IV  iteration k for x4

%% [ONE-STEP] RK4: Runge-Kutta Method - One-Step Method for Solving ODE
(Approximating ODE Solutions)

%% (RK4): First Iteration to Obtain x1 = x(0.2)(mathnotation), x(2) (Matlab) ------ ; y1
= y(0.2)(mathnotation), y(2) (Matlab)

% MATH-WISE: using i Variable for Iterations where each iteration (i) changes the

% notation (or index) of t,x,y,z in substitution

fori=1:4

% i stands for iteration I which is to calcaulte x1,x2,x3,x4 in RK Method

for c=1:4

for r=1:3

if r==1 % substitute function f

if c==4

    k(r,c)=H*f(t(i)+H,x(i)+k(r,c-1),y(i)+k(r+1,c-1),z(i)+k(r+2,c-1));
```

```
end
if c==1
    k(r,c)=H*f(t(i),x(i),y(i),z(i));
elseif c<4
    k(r,c)=H*f(t(i)+H/2,x(i)+k(r,c-1)/2,y(i)+k(r+1,c-1)/2,z(i)+k(r+2,c-1)/2);
end
end
if r==2 % substitute function g
if c==4
    k(r,c)=H*g(t(i)+H,x(i)+k(r-1,c-1),y(i)+k(r,c-1),z(i)+k(r+1,c-1));
end
if c==1
    k(r,c)=H*g(t(i),x(i),y(i),z(i));
elseif c<4
    k(r,c)=H*g(t(i)+H/2,x(i)+k(r-1,c-1)/2,y(i)+k(r,c-1)/2,z(i)+k(r+1,c-1)/2);
end
end
if r==3 % substitute function m
if c==4
    k(r,c)=H*m(t(i)+H,x(i)+k(r-2,c-1),y(i)+k(r-1,c-1),z(i)+k(r,c-1));
end
```

```
if c==1
        k(r,c)=H*m(t(i),x(i),y(i),z(i));
elseif c<4
        k(r,c)=H*m(t(i)+H/2,x(i)+k(r-2,c-1)/2,y(i)+k(r-1,c-1)/2,z(i)+k(r,c-1)/2);
end
end
end% for r
switchi
case 1
    A = k;    % 3x4 [k11, k12, k13, k14; k21, k22, k23, k24; k31, k32, k33, k34]
case 2
    B = k;
case 3
    C = k;
case 4
    D = k;
end
end%for c
t(i+1)=t(i)+H;  % time update equation
% CALCULATION
x(i+1)= x(i) + (1/6)*(k(1,1)+2*k(1,2)+2*k(1,3)+k(1,4));
```

```
y(i+1)= y(i) + (1/6)*(k(2,1)+2*k(2,2)+2*k(2,3)+k(2,4));

z(i+1)= z(i) + (1/6)*(k(3,1)+2*k(3,2)+2*k(3,3)+k(3,4));

end

%%

x4rk = x(5);

y4rk = y(5);

z4rk = z(5);

fprintf('ONE-STEP:[Runga-Kutta RK4]  @t=(0.8) x4=%f ,y4=%f ,z4=%f
\n',x4rk,y4rk,z4rk)

toc

fprintf('\n')

O = [A;B;C;D];

% CALL OUT

x;

y;

z;

%fprintf('\n')

%disp('_______')

%disp('column: t , k1 , k2 , k3 , k4 , x , exact , error')

%fprintf('\n')

disp('_______')

disp('column: t , k1 , k2 , k3 , k4 , x')
```

```
P = [t(1),0,0,0,0,x(1)
    t(2),A(1,1),A(1,2),A(1,3),A(1,4),x(2)
    t(3),B(1,1),B(1,2),B(1,3),B(1,4),x(3)
    t(4),C(1,1),C(1,2),C(1,3),C(1,4),x(4)
    t(5),D(1,1),D(1,2),D(1,3),D(1,4),x(5)];
%disp('t k1 k2 k3 k4 x')
disp(P)
disp('_______')
disp('column: t , k1 , k2 , k3 , k4 , y')
PP = [t(1),0,0,0,0,y(1)
     t(2),A(2,1),A(2,2),A(2,3),A(2,4),y(2)
     t(3),B(2,1),B(2,2),B(2,3),B(2,4),y(3)
     t(4),C(2,1),C(2,2),C(2,3),C(2,4),y(4)
     t(5),D(2,1),D(2,2),D(2,3),D(2,4),y(5)];
disp(PP)
disp('_______')
disp('column: t , k1 , k2 , k3 , k4 , z')
PPP = [t(1),0,0,0,0,z(1)
    t(2),A(1,1),A(1,2),A(1,3),A(1,4),z(2)
    t(3),B(1,1),B(1,2),B(1,3),B(1,4),z(3)
    t(4),C(1,1),C(1,2),C(1,3),C(1,4),z(4)
```

```
    t(5),D(1,1),D(1,2),D(1,3),D(1,4),z(5)];
%disp('t k1 k2 k3 k4 z')
disp(PPP)
plot(t,x,t,EXACT_X)
title('Plot of t versus RK X values and Exact X values')
xlabel('t')
ylabel('Runge-Kutta X values, Exact X values')
figure
plot(t,z,t,EXACT_Z)
title('Plot of t versus RK Z values and Exact Z values')
xlabel('t')
ylabel('Runge-Kutta Z values, Exact Z values')
figure
plot(t,y,t,EXACT_Y)
title('Plot of t versus RK Y values and Exact Y values')
xlabel('t')
ylabel('Runge-Kutta Y values, Exact Y values')
%% [TWO-STEP] Adam-Bashforth Method
tic
x_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)
y_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)
```

```
z_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)

% x0' y0' z0'

x_prime(1)=f(t(1),x(1),y(1),z(1));

y_prime(1)=g(t(1),x(1),y(1),z(1));

z_prime(1)=m(t(1),x(1),y(1),z(1));

% x1' y1' z1'

x_prime(2)=f(t(2),x(2),y(2),z(2));

y_prime(2)=g(t(2),x(2),y(2),z(2));

z_prime(2)=m(t(2),x(2),y(2),z(2));

% x2' y2' z2'

x_prime(3)=f(t(3),x(3),y(3),z(3));

y_prime(3)=g(t(3),x(3),y(3),z(3));

z_prime(3)=m(t(3),x(3),y(3),z(3));

% x3' y3' z3'

x_prime(4)=f(t(4),x(4),y(4),z(4));

y_prime(4)=g(t(4),x(4),y(4),z(4));

z_prime(4)=m(t(4),x(4),y(4),z(4));

% x4' y4' z4'(calculated by formula)

x_prime(5)= x(4)+(H/24)*(55*x_prime(4)-59*x_prime(3)+37*x_prime(2)-
9*x_prime(1));

y_prime(5)= y(4)+(H/24)*(55*y_prime(4)-59*y_prime(3)+37*y_prime(2)-
9*y_prime(1));
```

```
z_prime(5)= z(4)+(H/24)*(55*z_prime(4)-59*z_prime(3)+37*z_prime(2)-
9*z_prime(1));

x4STAR=x_prime(5);

y4STAR=y_prime(5);

z4STAR=z_prime(5);

% CALL OUT

x_prime;

y_prime;

z_prime;

fprintf('TWO-STEP:[Adam-Bashforth] @t=(0.8) x4=%f ,y4=%f ,z4=%f
\n',x_prime(5),y_prime(5),z_prime(5))

toc

fprintf('\n')

%% [TWO-STEP] Adam-Moulton Method

% MATH-WISE: with step sie of h=0.2, x(0.8), y(0.8) will be approxmiated by x4 and
y4

% we use the RK4 method with x3 and y3,and t4=0.8 by previous calculation

tic

% Array of X & Y Values for Adam-Moulton Method

x_AM = [x(1) x(2) x(3) x(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

y_AM = [y(1) y(2) y(3) y(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

z_AM = [z(1) z(2) z(3) z(4) 0]; %x0 x1 x2 x3 x4 (5 elements)
```

```
% values of x in Adam-Moulton are the same of these RK4
%{
DOCUMENTATION NOTE:
Formula: yn+1 = yn + h/24 (9fn+1 19fn -5fn-1 + fn-2)
To calculate x4,
x4 = x3 + h/24 (9*x4'+19*x3'-5*x2'+x1')
where:
x3: from RK4
x3',x2',x1 from Adam-Bashforth (primes)
x4': PREPARATION X4 From Adam-Bashforth
To calculate y4,
y4 = y3 + h/24 (9*y4'+19*y3'-5*y2'+y1')
where:
y3: from RK4
y3',y2',y1 from Adam-Bashforth (primes)
y4' PREPARATION Y4 From Adam-Bashforth
%}
% PREPARATION for X4 and Y4 NEW PRIME FOR NEW SUBSTITUTION
X_NEW_PRIME = f(t(5),x4STAR,y4STAR,z4STAR);
% the new x4' prime for Adam-Moulton
Y_NEW_PRIME = g(t(5),x4STAR,y4STAR,z4STAR);
```

```
% the new y4' prime for Adam-Moulton

Z_NEW_PRIME = m(t(5),x4STAR,y4STAR,z4STAR);

% the new z4' prime for Adam-Moulton

% NEW SUBSTITUTION

x_AM(5)= x(4)+(H/24)*(9*X_NEW_PRIME + 19*x_prime(4) -5*x_prime(3) +
x_prime(2));

y_AM(5)= y(4)+(H/24)*(9*Y_NEW_PRIME + 19*y_prime(4) -5*y_prime(3) +
y_prime(2));

Z_AM(5)= z(4)+(H/24)*(9*Z_NEW_PRIME + 19*z_prime(4) -5*z_prime(3) +
z_prime(2));

% CALL OUT

x_AM;

y_AM;

Z_AM;

fprintf('TWO-STEP:[Adam-Mouton]  @t=(0.8) x4=%f ,y4=%f ,z4=%f
\n',x_AM(5),y_AM(5),Z_AM(5))

toc

fprintf('\n')

%% [TWO-STEP] Predictor-Corrector Method

% MATH-WISE: use the predictor-corrector method to obtain an approximation value
of

% x(0.8) and y(0.8)  for the solution of the system (rk4) with h=0.2

tic
```

```
% Array of X & Y Values for Predictor-Corrector Method

X_PC = [x(1) x(2) x(3) x(4) 0];

Y_PC = [y(1) y(2) y(3) y(4) 0];

Z_PC = [z(1) z(2) z(3) z(4) 0];

%{

DOCUMENTATION NOTE:

PREPARATION for this method:

x4^ (x4 HAT)

y4^ (y4 HAT)

WHERE:

x4^ = x3 + h f(t3,x3,y3)

VALUES FROM RK4

AND:

y4^ = y3 + h g(t3,x3,y3)

VALUES FROM RK4

x4 = x3 + (H/2)(f(t3,x3,y3)+f(t4,x4^,y4^))

AND:

y4 = y3 + (H/2)(g(t3,x3,y3)+g(t4,x4^,y4^))

f(t3,x3,y3) = PREDICTOR

f(t4,x4^,y4^)= CORRECTOR

%}
```

```
% PREPARATION for X4 and Y4 HAT

XHAT = x(4) + H*f(t(4),x(4),y(4),z(4));

YHAT = y(4) + H*g(t(4),x(4),y(4),z(4));

ZHAT = z(4) + H*m(t(4),x(4),y(4),z(4));

% Obtain Approx Value of x(0.8) and y(0.8)

X_PC(5) = x(4) + (H/2)*((f(t(4),x(4),y(4),z(4)))+f(t(5),XHAT,YHAT,ZHAT));

Y_PC(5) = y(4) + (H/2)*((g(t(4),x(4),y(4),z(4)))+g(t(5),XHAT,YHAT,ZHAT));

Z_PC(5) = z(4) + (H/2)*((m(t(4),x(4),y(4),z(4)))+m(t(5),XHAT,YHAT,ZHAT));

fprintf('TWO-STEP:[Predictor-Corrector Method]  @t=(0.8) x4=%f ,y4=%f ,z4=%f
\n',X_PC(5),Y_PC(5),Z_PC(5))

toc

% CALL OUT

X_PC;

Y_PC;

Z_PC;

%% COMPARISON PLOT

plot(t,ERORR_Y)

title('Plot of Erorr Y values')

xlabel('t')

ylabel('Absolute of Erorr of Y')

figure

plot(t,ERORR_Z)
```

```
title('Plot of Erorr Z values')

xlabel('t')

ylabel('Absolute of Erorr of Z')

figure

plot(t,ERORR_X)

title('Plot of Erorr X values')

xlabel('t')

ylabel('Absolute of Erorr of X')

%% COMPARISON PLOT

plot(0.8,x_prime(5),'bx',0.8,x_AM(5),'r+',0.8,X_PC(5),'go',0.8,EXACT_X(5),'kd')

figure

plot(0.8,y_prime(5),'bx',0.8,y_AM(5),'r+',0.8,Y_PC(5),'go',0.8,EXACT_Y(5),'kd')

figure

plot(0.8,z_prime(5),'bx',0.8,Z_AM(5),'r+',0.8,Z_PC(5),'go',0.8,EXACT_Z(5),'kd')
```

Appendix C

Matlab code for example 3.3

$$x' = 3x - y + 4e^{2t}, \quad x(0) = 1$$

$$y' = -x + 3y + 4e^{4t}, \quad y(0) = 1$$

The enters required to matlab code

Enter the desired step size h required for computation: 0.2

Enter the t0 IC: 0

Enter the x0 IC: 1

Enter the y0 IC: 1

```
%% USER INPUT

H= input('Enter the desired step size h required for computation: ');

e=exp(1)

% Identifications or Functions

f=@(t,x,y)(4*e^(2*t)+3*x-y);

g=@(t,x,y)(4*e^(4*t)-x+3*y);

% Exact Solution: Input by user (Example)

EXACT_X=[1,2.665770667,5.498530724,9.482564489,12.78483153];

EXACT_Y=[1,2.978770948,7.968399611,20.53206889,52.12993262];

% erorr Solution: Input by user (Example)

ERORR_X=[0,0.00316099,0.00409334,0.022688187,0.094200336];

ERORR_Y=[0,0.0014023,0.007978098,0.03321463,0.119853838];

% IC: Inital Conditions For t,x,y
```

```
t0 = input('Enter the t0 IC: ');

x0 = input('Enter the x0 IC: ');

y0 = input('Enter the y0 IC: ');

fprintf('\n')

tic

%%

% Independent Variables

t = [t0 0 0 0 0];

x = [x0 0 0 0 0]; %[x0 x(0.2) x(0.4) x(0.6) x(0.8)]

y = [y0 0 0 0 0];

k = [0 0 0 0;0 0 0 0]; % Initialization for k11,k12,k13,14,k21,k22,k23,k24

O = randi([0,0],8,4);

A=[];B=[];C=[];D=[]; O=[];

% A I   iteration k for x1

% B II  iteration k for x2

% C III iteration k for x3

% D IV  iteration k for x4

%% [ONE-STEP] RK4: Runge-Kutta Method - One-Step Method for Solving ODE
(Approximating ODE Solutions)

%% (RK4): First Iteration to Obtain x1 = x(0.2)(mathnotation), x(2) (Matlab) ------ ; y1
= y(0.2)(mathnotation), y(2) (Matlab)

% MATH-WISE: using i Variable for Iterations where each iteration (i) changes the
```

```
% notation (or index) of t,x,y in substitution
fori=1:4
% i stands for iteration I which is to calcaulte x1,x2,x3,x4 in RK Method
for c=1:4
for r=1:2
if mod(r,2) % odd then sub f
if c==4
    k(r,c)=H*f(t(i)+H,x(i)+k(r,c-1),y(i)+k(r+1,c-1));
end
if c==1
k(r,c)=H*f(t(i),x(i),y(i));
elseif c<4
    k(r,c)=H*f(t(i)+H/2,x(i)+k(1,c-1)/2,y(i)+k(r+1,c-1)/2);
end
else% even then sub g
if c==4
    k(r,c)=H*g(t(i)+H,x(i)+k(r-1,c-1),y(i)+k(r,c-1));
end
if c==1
k(r,c)=H*g(t(i),x(i),y(i));
elseif c<4
```

```
        k(r,c)=H*g(t(i)+H/2,x(i)+k(r-1,c-1)/2,y(i)+k(r,c-1)/2);
end
end% END-IF EVEN OR ODD CHECK FOR F(t,x,y) or G(t,x,y) SUB
end% for r
switchi
case 1
   A = k;
case 2
   B = k;
case 3
   C = k;
case 4
   D = k;
end
end%for c
t(i+1)=t(i)+H;  % time update equation
% CALCULATION
x(i+1)= x(i) + (1/6)*(k(1,1)+2*k(1,2)+2*k(1,3)+k(1,4));
y(i+1)= y(i) + (1/6)*(k(2,1)+2*k(2,2)+2*k(2,3)+k(2,4));
end
x4rk = x(5);
```

```
y4rk = y(5);

fprintf('ONE-STEP:[Runga-Kutta RK4]  @t=(0.8) x4=%f ,y4=%f \n',x4rk,y4rk)

toc

fprintf('\n')

O = [A;B;C;D];

% CALL OUT

x,y

%fprintf('\n')

%disp('_____')

%disp('column: t , k1 , k2 , k3 , k4 , x , exact , error')

%fprintf('\n')

disp('_____')

disp('column: t , k1 , k2 , k3 , k4 , x')

P = [t(1),0,0,0,0,x(1)

    t(2),A(1,1),A(1,2),A(1,3),A(1,4),x(2)

    t(3),B(1,1),B(1,2),B(1,3),B(1,4),x(3)

    t(4),C(1,1),C(1,2),C(1,3),C(1,4),x(4)

    t(5),D(1,1),D(1,2),D(1,3),D(1,4),x(5)];

%disp('t k1 k2 k3 k4 x')

disp(P)

disp('_____')
```

```
disp('column: t , k1 , k2 , k3 , k4 , y')
PP = [t(1),0,0,0,0,y(1)
    t(2),A(2,1),A(2,2),A(2,3),A(2,4),y(2)
    t(3),B(2,1),B(2,2),B(2,3),B(2,4),y(3)
    t(4),C(2,1),C(2,2),C(2,3),C(2,4),y(4)
    t(5),D(2,1),D(2,2),D(2,3),D(2,4),y(5)];
disp(PP)
plot(t,x,t,EXACT_X)
title('Plot of t versus RK X values and Exact X values')
xlabel('t')
ylabel('Runge-Kutta X values, Exact X values')
figure
plot(t,y,t,EXACT_Y)
title('Plot of t versus RK Y values and Exact Y values')
xlabel('t')
ylabel('Runge-Kutta Y values, Exact Y values')
%% [TWO-STEP] Adam-Bashforth Method
tic
x_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)
y_prime = [0 0 0 0]; %x0 x1 x2 x3 x4 (5 elements)
% x0' y0'
```

```
x_prime(1)=f(t(1),x(1),y(1));

y_prime(1)=g(t(1),x(1),y(1));

% x1' y1'

x_prime(2)=f(t(2),x(2),y(2));

y_prime(2)=g(t(2),x(2),y(2));

% x2' y2'

x_prime(3)=f(t(3),x(3),y(3));

y_prime(3)=g(t(3),x(3),y(3));

% x3' y3'

x_prime(4)=f(t(4),x(4),y(4));

y_prime(4)=g(t(4),x(4),y(4));

% x4' y4' (calculated by formula)

x_prime(5)= x(4)+(H/24)*(55*x_prime(4)-59*x_prime(3)+37*x_prime(2)-
9*x_prime(1));

y_prime(5)= y(4)+(H/24)*(55*y_prime(4)-59*y_prime(3)+37*y_prime(2)-
9*y_prime(1));

x4STAR=x_prime(5);

y4STAR=y_prime(5);

% CALL OUT

x_prime;

y_prime;
```

```
fprintf('TWO-STEP:[Adam-Bashforth] @t=(0.8) x4=%f ,y4=%f
\n',x_prime(5),y_prime(5))

toc

fprintf('\n')

%% [TWO-STEP] Adam-Moulton Method

% MATH-WISE: with step sie of h=0.2, x(0.8), y(0.8) will be approxmiated by x4 and
y4

% we use the RK4 method with x3 and y3,and t4=0.8 by previous calculation

tic

% Array of X & Y Values for Adam-Moulton Method

x_AM = [x(1) x(2) x(3) x(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

y_AM = [y(1) y(2) y(3) y(4) 0]; %x0 x1 x2 x3 x4 (5 elements)

% values of x in Adam-Moulton are the same of these RK4

%{

DOCUMENTATION NOTE:

Formula: yn+1 = yn + h/24 (9fn+1 19fn -5fn-1 + fn-2)

To calculate x4,

x4 = x3 + h/24 (9*x4'+19*x3'-5*x2'+x1')

where:

x3: from RK4

x3',x2',x1 from Adam-Bashforth (primes)

x4': PREPARATION X4 From Adam-Bashforth
```

```
To calculate y4,

y4 = y3 + h/24 (9*y4'+19*y3'-5*y2'+y1')

where:

y3: from RK4

y3',y2',y1 from Adam-Bashforth (primes)

y4' PREPARATION Y4 From Adam-Bashforth

%}

% PREPARATION for X4 and Y4 NEW PRIME FOR NEW SUBSTITUTION

X_NEW_PRIME = f(t(5),x4STAR,y4STAR);

% the new x4' prime for Adam-Moulton

Y_NEW_PRIME = g(t(5),x4STAR,y4STAR);

% the new y4' prime for Adam-Moulton

% NEW SUBSTITUTION

x_AM(5)= x(4)+(H/24)*(9*X_NEW_PRIME + 19*x_prime(4) -5*x_prime(3) +
x_prime(2));

y_AM(5)= y(4)+(H/24)*(9*Y_NEW_PRIME + 19*y_prime(4) -5*y_prime(3) +
y_prime(2));

% CALL OUT

x_AM;

y_AM;

fprintf('TWO-STEP:[Adam-Mouton]  @t=(0.8) x4=%f ,y4=%f \n',x_AM(5),y_AM(5))

toc
```

```
fprintf('\n')

%% [TWO-STEP] Predictor-Corrector Method

% MATH-WISE: use the predictor-corrector method to obtain an approximation value
of

% x(0.8) and y(0.8)  for the solution of the system (rk4) with h=0.2

tic

% Array of X & Y Values for Predictor-Corrector Method

X_PC = [x(1) x(2) x(3) x(4) 0];

Y_PC = [y(1) y(2) y(3) y(4) 0];

%{

DOCUMENTATION NOTE:

PREPARATION for this method:

x4^ (x4 HAT)

y4^ (y4 HAT)

WHERE:

x4^ = x3 + h f(t3,x3,y3)

VALUES FROM RK4

AND:

y4^ = y3 + h g(t3,x3,y3)

VALUES FROM RK4

x4 = x3 + (H/2)(f(t3,x3,y3)+f(t4,x4^,y4^))

AND:
```

```
y4 = y3 + (H/2)(g(t3,x3,y3)+g(t4,x4^,y4^))

f(t3,x3,y3) = PREDICTOR

f(t4,x4^,y4^)= CORRECTOR

%}

% PREPARATION for X4 and Y4 HAT

XHAT = x(4) + H*f(t(4),x(4),y(4));

YHAT = y(4) + H*g(t(4),x(4),y(4));

% Obtain Approx Value of x(0.8) and y(0.8)

X_PC(5) = x(4) + (H/2)*(f(t(4),x(4),y(4))+f(t(5),XHAT,YHAT));

Y_PC(5) = y(4) + (H/2)*(g(t(4),x(4),y(4))+g(t(5),XHAT,YHAT));

fprintf('TWO-STEP:[Predictor-Corrector Method]  @t=(0.8) x4=%f ,y4=%f
\n',X_PC(5),Y_PC(5))

toc

% CALL OUT

X_PC;

Y_PC;

%% COMPARISON PLOT

plot(t,ERORR_Y)

title('Plot of Erorr Y values')

xlabel('t')

ylabel('Absolute of Erorr of Y')

figure
```

```
plot(t,ERORR_X)

title('Plot of Erorr X values')

xlabel('t')

ylabel('Absolute of Erorr of X')

%% COMPARISON PLOT

plot(0.8,x_prime(5),'bx',0.8,x_AM(5),'r+',0.8,X_PC(5),'go',0.8,EXACT_X(5),'kd')

figure

plot(0.8,y_prime(5),'bx',0.8,y_AM(5),'r+',0.8,Y_PC(5),'go',0.8,EXACT_Y(5),'kd')
```

Appendix D

Tables

Table 3.5(a)
The exact and numerical solution of $x(t)$ by applying RKM

t	Exact solution	Approximate solution	Absolute error
0	6.000000000	6.000000000	0.000000000
0.2	8.680526246	8.680066666	0.000459580
0.4	12.61952934	12.61815964	0.001369700
0.6	18.42270341	18.41964123	0.003062184
0.8	26.99070305	26.98461670	0.006086350

Table3.5(b)
The exact and numerical solution of $y(t)$ by applying RKM.

t	Exact solution	Approximate solution	Absolute error
0	3.000000000	3.000000000	0.000000000
0.2	4.205052153	4.204866666	0.000185487
0.4	5.942906555	5.942354635	0.000551919
0.6	8.462352646	8.461120366	0.001232279
0.8	12.13160578	12.12915917	0.002446607

Table 3.5(c)
The exact and numerical solution of $z(t)$ by applying RKM

t	Exact solution	Approximate solution	Absolute error
0	4.000000000	4.000000000	0.000000000
0.2	5.696876851	5.696600000	0.000276851
0.4	8.168447483	8.167622973	0.000824510
0.6	11.78246957	11.78062732	0.001842248
0.8	17.0846382	17.08097835	0.003659851

Table 3.6(a)
The exact and numerical solution of applying Fourth Adams-Bashforth method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	26.99070305	26.94590986	0.04479319
$y(t)$	12.13160578	12.11357979	0.01802599
$z(t)$	17.0846382	17.05768981	0.02694839

Table 3.6 (b)
The exact and numerical solution of applying Fourth Adams-Moulton method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	26.99070305	26.98414715	0.0065559
$y(t)$	12.13160578	12.12897559	0.00263019
$z(t)$	17.0846382	17.0806994	0.0039388

Table 3.6(c)
The exact and numerical solution of applying Predictor-Corrector method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	26.99070305	26.78732134	0.20338171
$y(t)$	12.13160578	12.04871046	0.08289532
$z(t)$	17.0846382	16.96158076	0.12305744

Table 3.7(a)
The k's value for $x(t)$

t	$k1$	$k2$	$k3$	$k4$
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	1.200000000	1.617122206	1.678912892	2.204450400
0.4	2.197638088	2.817801139	2.847251258	3.491480903
0.6	3.489922878	4.167277058	3.981356026	4.228582622
0.8	4.259474291	4.203702701	3.219344241	1.137107026

Table 3.7(b)
The k's value for $y(t)$

t	k_1	k_2	k_3	k_4
0	0.000000000	0.000000000	0.000000000	0.000000000
0.2	1.200000000	1.833459758	1.981785464	3.033721443
0.4	3.033636578	4.599624538	5.007404621	7.650622296
0.6	7.638154034	11.52942689	12.62907333	19.27544199
0.8	19.21680312	28.89307294	31.80153105	48.46133596

Table 3.8(a)
The exact and numerical solution of $x(t)$ by applying RKM

t	Exact solution	Approximate solution	Absolute error
0	1.000000000	1.000000000	0.000000000
0.2	2.665770667	2.666086766	0.003160990
0.4	5.498530724	5.502624064	0.004093340
0.6	9.482564489	9.505252676	0.022688187
0.8	12.78483153	12.87903187	0.094200336

Table 3.8(b)
The exact and numerical solution of $y(t)$ by applying RKM

t	Exact solution	Approximate solution	Absolute error
0	1.000000000	1.000000000	0.000000000
0.2	2.978770948	2.977368648	0.001402300
0.4	7.968399611	7.960421513	0.007978098
0.6	20.53206889	20.49885426	0.033214630
0.8	52.12993262	52.01007878	0.119853838

Table 3.9(a)
The exact and numerical solution of applying Fourth Adams-Bashforth methodat $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	12.78483153	13.62517957	0.840348039
$y(t)$	52.12993262	49.98708915	2.142843368

Table 3.9(b)
The exact and numerical solution of applying Fourth Adams-Moulton method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	12.78483153	13.04438092	0.259549391
$y(t)$	52.12993262	51.83224182	0.297690801

Table 3.9(c)
The exact and numerical solution of applying Predictor-Corrector method at $t = 0.8$

	Exact solution	Approximate solution	Absolute error
$x(t)$	12.78483153	13.77405233	0.989220798
$y(t)$	52.12993262	50.45851494	1.67141768

Table 4.1
The Exact and Numerical solutions at $t = 0.8$ of system (3.1)

	Exact solution	**Rung-Kutta Method**	**Adams-Bashforth Method**	**Adams-Moulton Method**	**Predictor-Corrector Method**
$x(t)$	20.47497654	20.47440288	20.46995901	20.47456058	20.42187796
Error for $x(t)$	0.000000000	0.00057366	0.00501753	0.00041596	0.05309858
$y(t)$	11.57281283	11.57231957	11.56852001	11.57244907	11.52999845
Error for $y(t)$	0.000000000	0.00049326	0.00429282	0.00036376	0.04281438
Cpu-time in second		0.021504	0.000969	0.000896	0.0008

Table 4.2
The Exact and numerical solutions of at $t = 0.8$ of system (3.3)

	Exact solution	**Rung-Kutta Method**	**Adams-Bashforth Method**	**Adams-Moulton Method**	**Predictor-Corrector Method**
$x(t)$	26.99070305	26.98461670	26.94590986	26.98414715	26.78732134
Error for $x(t)$	0.000000000	0.006086350	0.04479319	0.0065559	0.20338171
$y(t)$	12.13160578	12.12915917	12.11357979	12.12897559	12.04871046
Error for $y(t)$	0.000000000	0.002446607	0.01802599	0.00263019	0.08289532
$z(t)$	17.0846382	17.08097835	17.05768981	17.0806994	16.96158076
Error for $z(t)$	0.000000000	0.003659851	0.02694839	0.0039388	0.12305744
Cpu-time in second		0.030624	0.001688	0.002347	0.001338

Table 4.3
The Exact and numerical solutions at $t = 0.8$ of system (3.5)

	Exact solution	**Rung-Kutta Method**	**Adams-Bashforth Method**	**Adams-Moulton Method**	**Predictor-Corrector Method**
$x(t)$	12.78483153	12.87903187	13.62517957	13.04438092	13.77405233
Error for $x(t)$	0.000000000	0.094200336	0.840348039	0.259549391	0.989220798
$y(t)$	52.12993262	52.01007878	49.98708915	51.83224182	50.45851494
Error for $y(t)$	0.000000000	0.119853838	2.142843368	0.297690801	1.67141768
Cpu-time in second		0.022645	0.000687	0.000501	0.000616

Appendix E

Figures

Figure 3.5(a)
The exact and numerical solutions of $z(t)$

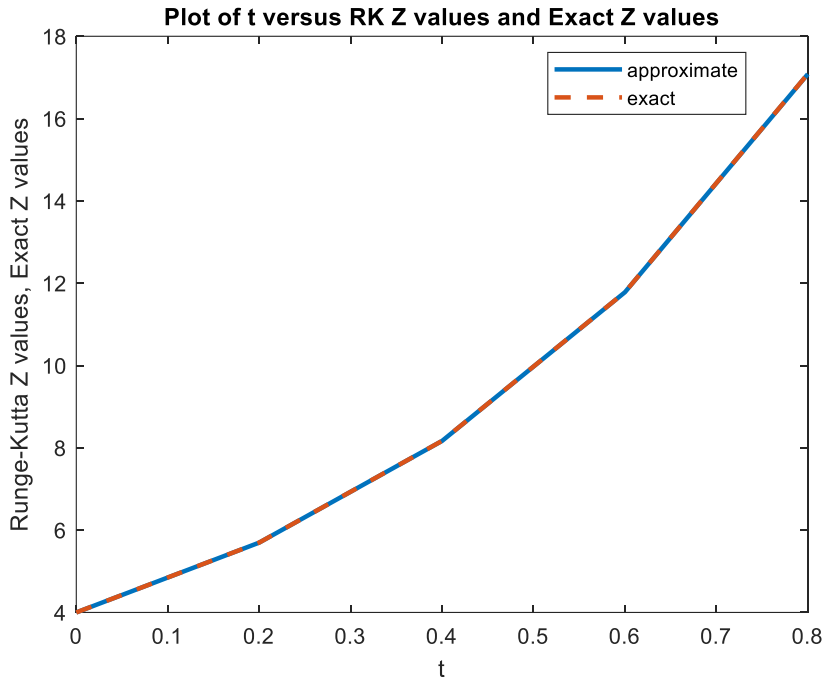


Figure 3.5(b)
the absolute error resulting of applying RKM of $z(t)$

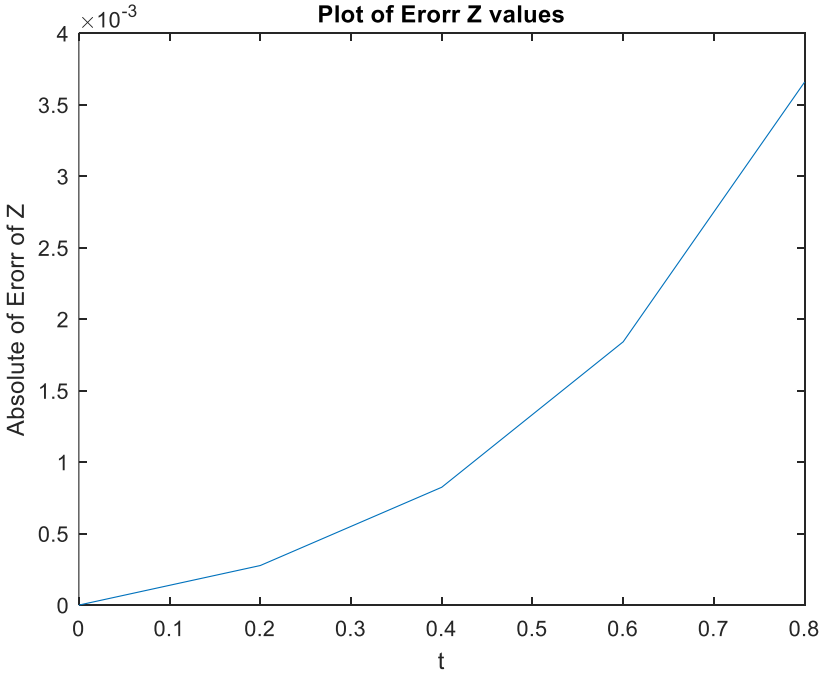


Figure 3.6(a)
the exact and numerical solution $x(t)$

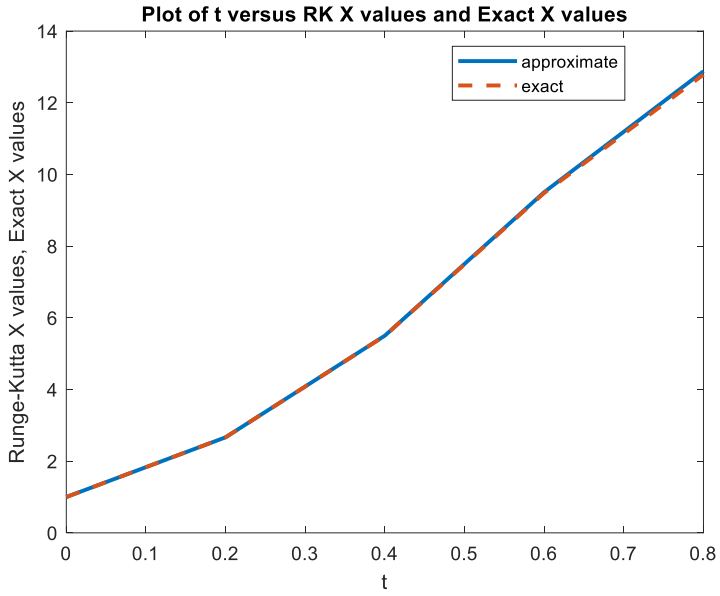


Figure 3.6 (b)
the absolute error resulting of applying RKM of $x(t)$

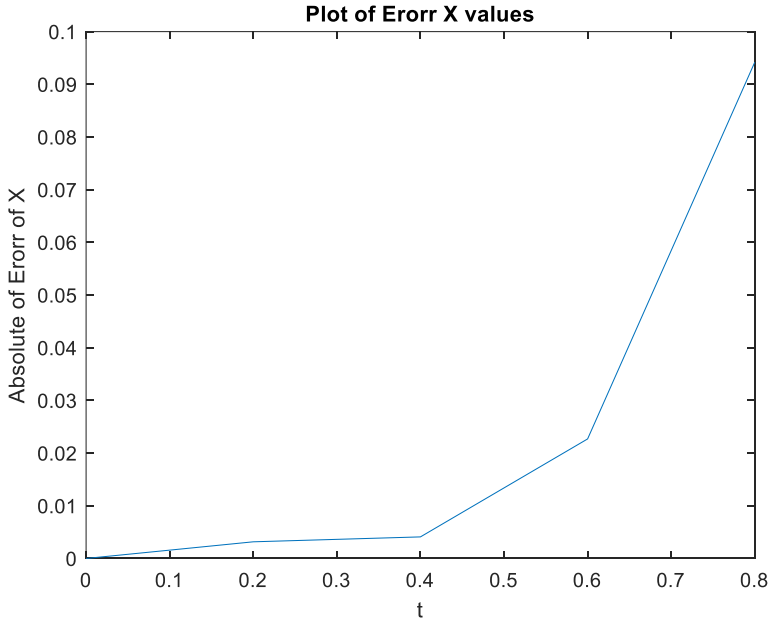


Figure 3.7(a)
the exact and numerical solutions $y(t)$

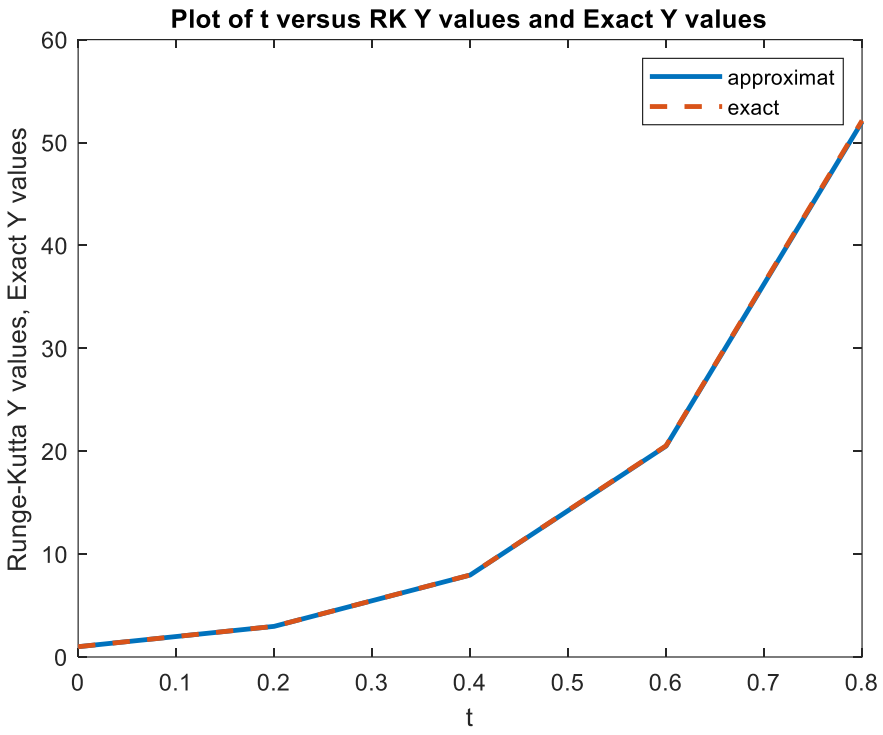
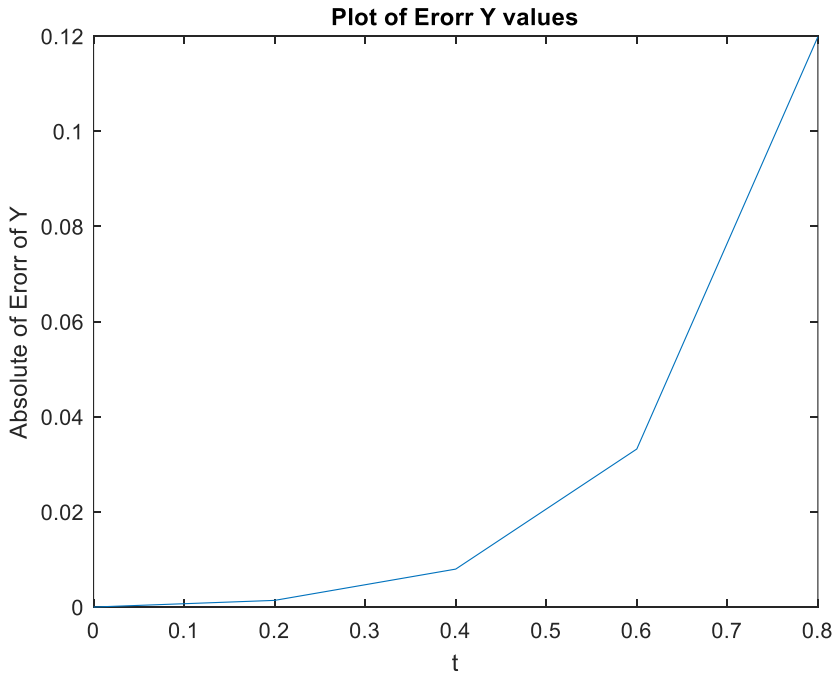


Figure 3.7(b)
the absolute error resulting of applying RKM of $y(t)$



جامعة النجاح الوطنية

كلية الدراسات العليا

مقارنة الطرق العددية لحل أنظمة المعادلات التفاضلية العادية

إعداد

عبد الفتاح بشناق

إشراف

أ. د. ناجي قطناني

قدمت هذه الرسالة/الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الرياضيات المحوسبة، من كلية الدراسات العليا، في جامعة النجاح الوطنية، نابلس– فلسطين.

2024

مقارنة الطرق العددية لحل أنظمة المعادلات التفاضلية العادية

إعداد

عبد الفتاح بشناق

إشراف

أ. د. ناجي قطناني

الملخص

في هذه الرسالة، نلقي الضوء على المعالجة العددية لأنظمة المعادلات التفاضلية العادية، هذه الأنظمة لها مجموعة واسعة من التطبيقات في الرياضيات الفيزيائية والكيميائية والحيوية والنماذج المجسمة والتوصيل الحراري والهندسة.

تم تقديم بعض الجوانب المهمة لأنظمة المعادلات التفاضلية بما في ذلك قابلية حل الأنظمة المتجانسة والغير متجانسة، كما تم التركيز على طرق العددية لحل أنظمة المعادلات التفاضلية وهي طرق ذات خطوة واحدة وتشمل طريقة Euler و Runge-Kutta وعدديدة الخطوات تشمل Adams-Bashforth method و Adams-Moulton method و Predictor-Corrector method .

تم توضيح الطرق العددية المقترحة من خلال حل بعض الأمثلة العددية وإظهار الخطأ المرتبط بهذه الطرق، وتظهر النتائج العددية بوضوح ان الطرق المتعددة الخطوات أكثر كفاءة وتعطي تقاربا اسرع من الطرق الأخرى.

الكلمات المفتاحية: المقارنة، الطرق العددية، أنظمة المعادلات.