THE COMPARISON OF RUNGE-KUTTA AND ADAMS-BASHFORH-MOULTON METHODS FOR THE FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS

By: Solomon G/giorgis

Advisor: Genanew Gofe (Ph.D.)

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF JIMMA UNIVERSITY, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN MATHEMATICS

June, 2014

Table of Contents

Content **Page**

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Genanew Gofe for his guidance and constructive comments throughout the study period. His dedication, patience, and immediate responses are deeply acknowledged.

I would also like to thank the School of Graduate Studies of Jimma University for the financial support to accomplish this research work.

Last but not least I would like to thank Mrs. Gelaye G/michael for all the undeserved helps and encouragements she provided to me.

Solomon G/giorgis

ACRONYMS

RK4 Runge-Kutta method of order 4

ABM Adams-Bashforth-Moulton

MABM Modified Adams-Bashforth-Moulton

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF APPENDICES

Page

Abstract

The performances of Runge-Kutta and Adams-Bashforth-Moulton methods were compared by considering first order ordinary differential equations using the MATLAB software. For this purpose three major programs were coded for RK4, ABM, and MABM and run using the MATLAB software. This is done by varying the step length M and sketching graphs of computation time. For the comparison of accuracy, relative errors have been calculated for each first order ordinary differential equations and represented by graphs. Moreover the effectiveness of modifiers in the Adams-Bashforth-Moulton method has been validated. The result of this research show that ABM method is the most efficient method for first order ODE but in terms of accuracy there is no one best method among RK4, ABM, and MABM. So it is not possible to make generalizations. But it is possible to conclude that the performance of a given method depend on the characteristics of the ODEs we are considering. Regarding the modifiers in the corrector and predictor formulas of the ABM method, they are effective in improving the accuracy of ABM method in most cases but this doesn't work for some problems due to the stiffness of the problem and instability of the modifier in the corrector step. Future experiments can be done by increasing the types of numerical methods and extending the first order ODEs in to higher order.

CHAPTER 1

INTRODUCTION

1.1 Background of the study

Due to the advancements in the field of computational mathematics, numerical methods are most widely being utilized to solve the equations arising in the fields of applied medical sciences, engineering and technology.

There are several numerical methods to solve differential equations related to the initial value problem, that is the single-step methods and the multiple-step methods (Chapra and Canale, 1989).

Many contributions have been made in the area of numerical methods for ordinary differential equations. By contrast, relatively little has been done assessing the merits of various methods in a reasonably definitive way. In assessing the merits of various methods, we need to consider the problem to be solved, methods to be considered, and comparison criteria (Hull *et al.*, 1972).

The major factors to be considered in comparing different numerical methods are the accuracy of the numerical solution and its computation time (Bedet *et al*., 1975). They further indicated that it is important to note that the comparison of numerical methods is not so simple because their performances may depend on the characteristic of the problem at hand. It should also be noted that there are other factors to be considered, such as stability, versatility, proof against run-time error, and so on which are being considered in most of the MATLAB built-in routines (Yang *et al*., 2005).

Performance actually depends on several factors: the computation time taken for one iteration of the algorithm, the time step for one iteration which represents the time discretization required to reach a given accuracy or numerical stability for a given method, the desired accuracy of the method, the numerical stability of the method which also limits the time step for a given method (Volino and Magnenat-Thalmann, 2000). They further indicated that accuracy increases along with time step reduction as better as the method is high-order.

In our case the methods selected are the explicit Runge-Kutta method of fourth order which is a single step method and Adams-Bashforth-Moulton predictor corrector method of fourth order which is a multistep method. The methods selected are among the best methods available (Hull *et al.*, 1972).

Runge-Kutta method of fourth order is used to approximate the solution of the initial value problem $y' = f(x, y)$ with $y(a) = y_0$ over [a, b] with step length h is given as follows:

$$(1) \qquad y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Where

$k_1 = hf(x_i, y_i)$, $k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$, $k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$, $and\ k_4 = hf(x_i + h, y_i + k_3)$

Each Runge-Kutta methods are derived from an appropriate Taylor method in such a way that the final global error is of order $O(h^N)$ (Mathews. *et al*., 2004). In this method several function evaluations is performed at each step and eliminate the necessity to compute the higher derivatives. These methods can be considered for any order N.

The Runge-Kutta method of order N = 4 is most popular. It is a good choice for common purposes because it is quite accurate, stable, and easy to program. Hence it is not necessary to go to a higher-order method because the increased accuracy is offset by additional computational effort (Mathews. *et al*., 2004). If more accuracy is required, then either a smaller step size or an adaptive method should be used.

Step halving (also called adaptive Runge-Kutta) involves halving the step size and comparing the answers with two step sizes (Hornberger and Wiberg, 2005). They further explained that a significant difference between the two answers would suggest that the step size is too big because smaller step sizes generally lead to more accurate results. So, one can see that this process can be continued until an acceptably small difference between the two estimates is obtained.

The Adams-Bashforth-Moulton method of fourth order is used to approximate the solution of the initial value problem $y' = f(x, y)\ with\ y(a) = y_0\ over\ [a, b]$ ….(*)

with predictor and corrector schemes which are given as follows:

$$(2) \qquad predictor: \quad p_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3})$$

$$(3) \qquad corrector: \quad y_{k+1} = y_k + \frac{h}{24}(9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2})$$

To obtain equations (2) and (3), the value of f(x, y) in (*) is replaced by the Newton backward interpolating polynomial after converting (*) into the appropriate integral form.

A desirable feature of a multistep method is that the local truncation error can be determined and a correction term can be included, which improves the accuracy of the answer at each step. Also it is possible if the step size is small enough to obtain an accurate value for y_{k+1}, yet large enough so that unnecessary and time-consuming calculations are eliminated. Using the combination of a predictor and corrector requires only two function evaluations of f(x, y) per step.

By using Taylor expansion of y_{k+1} about x_k and y_k about x_{k+1} , replacing the first, second, and third derivatives by their difference approximations, and assuming $f_{k+1}^{(4)} = f_k^{(4)} \cong K$ where K is a constant, we can obtain the predictor/corrector errors (Yang *et al*., 2005). As a result Adams–Bashforth–Moulton method with modification formulas will be given as follows.

$$predictor: \quad p_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3})$$

$$(4) \qquad Modifier: \quad m_{k+1} = p_{k+1} + \frac{251}{270}(c_k - p_k)$$

$$Corrector: \quad c_{k+1} = y_k + \frac{h}{24}(9f(t_{k+1}, m_{k+1}) + 19f_k - 5f_{k-1} + f_{k-2})$$

$$(5) \qquad y_{k+1} = c_{k+1} - \frac{19}{270}(c_{k+1} - p_{k+1})$$

The quantity $c_1 - p_1$ required for the modification of the first step is generally taken as $c_1 - p_1 = 0$ (Jain *et al*., 1984).

Formula (5) can be used to determine when to change the step size. This can be done by reducing the step size to h/2 or increase it to 2h by using the following criterion:

(6) If $\frac{19}{270}\frac{|Y_{j+1}-p_{j+1}|}{|Y_{j+1}|+Small} > Relative\ Error, then\ set\ h = \frac{h}{2}$.

(7) If $\frac{19}{270}\frac{|Y_{j+1}-p_{j+1}|}{|Y_{j+1}|+Small} < \frac{Relative\ \ Error}{100}, then\ set\ h = 2h.$

When the predicted and corrected values do not agree to five significant digits, then (6) reduces the step size. If they agree to seven or more significant digits then (7) increases the step size (Mathews. *et al*., 2004).

According Mathews *et al*., (2004), the step size h for a fixed-point iteration using RK4 and ABM method must satisfy the following condition.

(8) $h < \frac{0.75}{|f_y(x,y)|}$.

The condition stated in (8) is following the fact that the absolute stability interval for the explicit RK4 is $0 < |\lambda h| < 2\sqrt{2}$ when the method is applied to the test equation $y' = \lambda y$ (Jain *et al*., 1984). Similarly the intervals of absolute stability for Adams-Bashforth, Adams-Moulton, and MABM methods of order 4 are (-0.3, 0), (-3, 0), and (-0.6884, 0) respectively.

The performance of numerical methods depend on the characteristics of the ODE considered (Hull *et al*., 1972; Bedet *et al*., 1975; Butcher, 2000; Yang *et al*., 2005; Petzoid, 2006; Clement *et al*., 2009; Abdul, 2013; Polla, 2013; and Muhammmad and Arshad, 2013).

The first problem considered in this research is a first-order differential equation $y' = -y$ with $y(0) = 1$ (Hull *et al*., 1972). It has the following form of analytical solution $y = e^{-x}$. The second problem is $y' = -y^3/2,\ y(O) = 1$ is a special case of the Riccati equation and whose solution is given by $y = \frac{1}{\sqrt{x+1}}$ (Davis, 1963 as cited in Hull *et al*., 1972). The third problem is $y' = ycosx,\ y(O) = 1$ which is an oscillatory problem and whose solution is given by $y = e^{sinx}$ (Hull *et al*., 1972). The fourth problem is $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$ whose solution is given by $y = \frac{20}{1 + 19e^{-\frac{x}{4}}}$ which is the logistic curve, (Davis, 1962 as cited in Hull *et al*.,

1972). The fifth problem is $y' = \frac{y-x}{y+x}$, $y(0) = 4$ whose solution is given by $r = 4e^{-\theta+\frac{\pi}{2}}$ which is a spiral curve (Davis, 1962 as cited in Hull *et al.*, 1972).

As it is shown above the problems selected have exact solutions. This is helpful to compare the approximated values with the exact values and to calculate relative errors. The selection covers a realistically broad spectrum of problem types (Hull *et al*., 1972).

The testing of ODE solvers will be done by using MATLAB after the problems are properly coded and inserted for analysis.

Hence the purpose of this research is to compare the performance of Runge-Kutta and Adams-Bashforth-Moulton methods for first order ordinary differential equations.

1.2 Statement of the problem

The performance of numerical methods depend on the characteristics of the ODE considered (Hull *et al*., 1972; Bedet *et al*., 1975; Butcher, 2000; Yang *et al.*, 2005; Petzoid, 2006; Clement *et al*., 2009; Abdul, 2013; Polla, 2013; and Muhammmad and Arshad, 2013). While the central activity of numerical analysts is providing accurate and efficient general purpose numerical methods and algorithms, there has always been a realization that some problem types have distinctive features that they will need their own special theory and techniques (Butcher, 2000).

Hence it is important to test methods by considering first order ODEs using modern software.

Therefore, the researcher will attempt to answer the following research questions:

a. Which of the two methods register smaller computation time?
b. Which of the two methods give more accurate results?
c. What is the effect of modifiers on Adam-Bashforth-Moulton method?

1.3. General and specific objectives

1.3.1 General Objective

- To compare the performances of Runge-Kutta and Adams-Bashforth-Moulton methods for first order ordinary differential equations.

1.3.2 Specific Objectives

- To compare the computation time of each method for the first order ODEs
- To compare the accuracy of the results obtained by implementing the codes for the first order ODEs

- To validate the effectiveness of the modifiers in the accuracy of Adams-Bashforth-Moulton method for first order ODEs

1.4. Significance of the study

This research is expected to provide useful results and information for all parties, including students, faculty, and other researchers, regarding which method between the two methods, Runge-Kutta method of order 4 and Adams-Bashforth-Moulton method of order 4, is better in solving first order ordinary differential equations.

1.5. Delimitation of the study

This paper is delimited to the comparison of the performances of Runge-Kutta method of fourth order and Adams-Bashforth-Moulton method of fourth order for first order ODEs.

CHAPTER 2

REVIEW OF RELATED LITERATURES

A code is more efficient if it solves problems in less CPU time. However, this criterion is problem dependent, and hence it is necessary to test efficiency by considering problems (Hull *et al*., 1972).

The major factors to be considered in evaluating/comparing different numerical methods are the accuracy of the numerical solution and its computation time (Bedet *et al*., 1975). They further indicated that it is important to note that the evaluation/comparison of numerical methods is not so simple because their performances may depend on the characteristic of the problem at hand. It should also be noted that there are other factors to be considered, such as stability, versatility, proof against run-time error, and so on which are being considered in most of the MATLAB built-in routines (Yang *et al*., 2005).

Performance actually depends on several factors: the computation time taken for one iteration of the algorithm, the time step for one iteration which represents the time discretization required to reach a given accuracy or numerical stability for a given method, the desired accuracy of the method, the numerical stability of the method which also limits the time step for a given method (Volino and Magnenat-Thalmann, 2000). They further indicated that accuracy increases along with time step reduction as better as the method is high-order.

Jorba and Zou (2004) showed that Taylor method can be competitive, both in speed and accuracy, with the standard methods by generating a code having adaptive selection of order and step size at run time.

Each Runge-Kutta methods are derived from an appropriate Taylor method in such a way that the final global error is of order $O(h^N)$ (Mathews *et al*., 2004). In this method several function evaluations is performed at each step and eliminate the necessity to compute the higher derivatives. These methods can be considered for any order N. So from the above explanation one can understand that the Runge-Kutta method can also be competitive, both in speed and accuracy.

The double step methods require smaller step-by-step evaluation than the one-step methods. One of the double step methods is Adams-Moulton method. According to Polla (2013), Adams-Moulton method is more stable than the Millne method.

Hull *et al*., (1972) compared RK4 and Adams methods for non-stiff first order ordinary differential equations on a variety of initial value problems. The methods are compared primarily as to how well they can handle relatively routine integration steps under a variety of accuracy requirements using the software DETEST. They concluded that Adams method is efficient than RK4.

Clement *et al*. (2008), compared the orbital estimations made by a series of common numerical integration schemes such as Euler's method, Runge-Kutta method of Order 2, Runge-Kutta method of Order 4, Adams Bashforth method, and Gill's method based on stability with varying time steps and accuracy requirements. The result of their study revealed that Gill's Method is slightly less stable, but more accurate with smaller time steps and hence it is the best numerical integration scheme to implement when estimating orbits in a celestial mechanics problem.

Muhammad and Arshad (2013) described membrane gas separation in chemical engineering using a model and identified proper numerical methods for the simulation of the model by adjusting the step size and tolerance level in MATLAB. The methods they considered are Bogacki–Shampine method, Dormand–Prince method, and Adams-Bashforth-Moulton method. They concluded that Adams-Bashforth-Moulton method showed stable and fast behavior and hence it is a proper numerical method giving accurate results and requiring an acceptable computational effort for the simulation of the model.

Abdul (2013) investigated the efficiency of improved Heun's (IH) method against the classical Runge-Kutta (RK4) and Mid-point (MP) methods for Unforced Van der Pol's Equation and concluded that RK4 method has better accuracy over IH and MP methods. The solutions by the RK4 are relatively better as expected.

According to Polla (2013) the Adams-Moulton method has more rigorous accuracy than the Runge-Kutta Fehlberg method in solving linear ordinary differential equations of first order and second order. The comparison of accuracy is obtained through comparing the value of differential equations results numerically with differential equations result obtained from

MATLAB (version 5.3). A number of experiments on the completion of linear ordinary differential equations of first order and second order are done through computerization to compare the accuracy between the Runge-Kutta Fehlberg and Adams-Moulton methods. In addition, accuracy is being pointed out through relative error.

Stiffness is a subtle, difficult, and important concept in the numerical solution of ordinary differential equations. It depends on the differential equation, the initial conditions, and the numerical method. Dictionary definitions of the word "stiff" involve terms like "not easily bent," "rigid," and "stubborn."

A problem is stiff if the solution being sought varies slowly, but there are nearby solutions that vary rapidly, so the numerical method must take small steps to obtain satisfactory results.

Stiffness is a special problem that can arise in the solution of ordinary differential equations. A stiff system is one involving rapidly changing components together with slowly changing ones (Chapra and Canale, 1989). In many cases, the rapidly varying components are ephemeral transients that die away quickly, after which the solution becomes dominated by the slowly varying components.

While the central activity of numerical analysts is providing accurate and efficient general purpose numerical methods and algorithms, there has always been a realization that some problem types have such distinctive features that they will need their own special theory and techniques. Stiff ODEs were recognized as such a problem types and received considerable attention (Butcher, 2000).

Yatim *et al*., (2011) derived a variable step of the implicit block methods based on the backward differentiation formulae (BDF) namely the 5th-order variable step BBDF for solving stiff initial value problems. They included a simplified strategy in controlling the step size with the aim of optimizing the performance in terms of precision and computation time. They also underlined that numerical results obtained support the enhancement of the method proposed as compared to MATLAB's suite of ODE solvers, namely, ode15s and ode23s.

CHAPTER 3

METHODOLOGY

3.1 Study site and period

The study is conducted in Jimma University, which is Ethiopia's first innovative community-oriented educational institution of higher learning. The research was conducted from October, 2014 up to June, 2014.

3.2 Source of information

The source of information for this research is the computation times and relative errors of the first order ordinary differential equations considered which are resolved using RK4, ABM, and MABM methods.

3.3 Method of the study

The study involves entirely laboratory work with the help of a laptop and a MATLAB software. So it is an experimental research. The methods are coded and run using MATLAB software by properly inserting the problems and as a result numerical results are automatically generated.

All algorithms have been made in the same condition, which use the same type of processor, having the same memory size, the same operating system, and using the same function. The processor used is Intel(R) 2.10 GHz, with 2 GB memory, with the 32-bit operating system (windows 7 home premium). The language program used is MATLAB version 7.14.

Three major programs (codes) have been written to solve first order ordinary differential equations using RK4, ABM, and MABM methods. The codes contain function definition line, input arguments, commands (function body), and output arguments which are written in the script file of MATLAB. The function definition line contains type of numerical method, function, left end point and right end point of an interval, initial condition, and number of steps. The input arguments are written in order to insert the input values after the codes are saved and debugged using MATLAB. In the function body the formula for the step size, the formulas for the methods, the exact solution, and the formula for relative errors have been coded. In the output argument appropriate notation of the outputs such as the partitions of the independent

variable(in our case x), the corresponding numerical values of the dependent variable(in our case y), and the relative error values (ε_r) have been written.

The following procedures will be followed to measure the accuracy and computation time.

a) Computation time

MATLAB has two convenient commands that let us measure how long an operation takes to solve a given problem after certain iterations. To start (and reset) a timer, use the command tic;. To stop the timer and display the amount of time elapsed, use toc;.

Computation times for RK4, ABM, MABM using the five problems have been calculated by varying the number of steps.

b) Accuracy

To find which numerical method gives a more accurate approximation this study compared the relative errors obtained by using RK4, ABM, MABM for the five problems selected.

Relative error (ε_r) is calculated using the formula $\varepsilon_r = \left|\frac{Exact\ Value - Approximated\ \ Value}{Exact\ Value}\right|$.

According Mathews *et al*., (2004), the step size h for a fixed-point iteration using RK4 and ABM method must satisfy the following condition $h < \frac{0.75}{|f_y(x,y)|}$. Hence the value of h is in such a way that it satisfies this condition.

Finally, data obtained by using MATLAB version 7.14 software about computation times and relative errors were analyzed after calculating the average and standard deviations. More over graphs of computation times and relative errors have been sketched for the purpose of analysis.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Result

4.1.1 Comparison of computation times

Data about computation times (in seconds) obtained using RK4, ABM, and MABM methods by varying the number of steps for the five problems.

ABM method has the greatest speed than RK4 and MABM and MABM is the second faster in approximating the solution of the ordinary differential equation $y' = -y, y(0) = 1$ (fig.1).

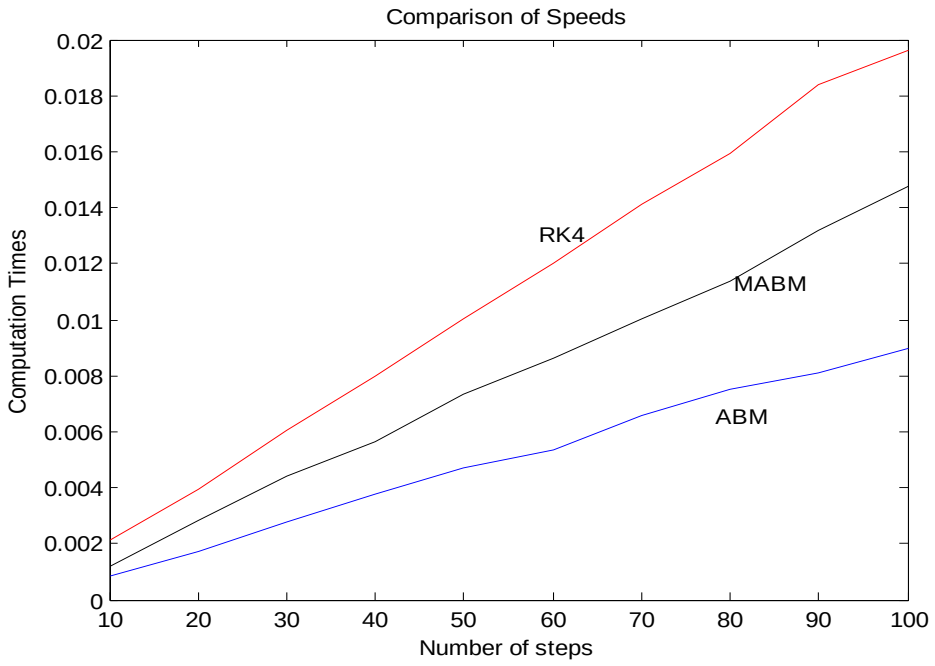


Figure 1: Plot of computation times of the three RK4, ABM, MABM methods by using the problem $y' = -y, y(0) = 1$

ABM method has the greatest speed than RK4 and MABM and MABM is the second faster in approximating the solution of the ordinary differential equation $y' = -y^3/2,\ y(O) = 1$ (fig.2).

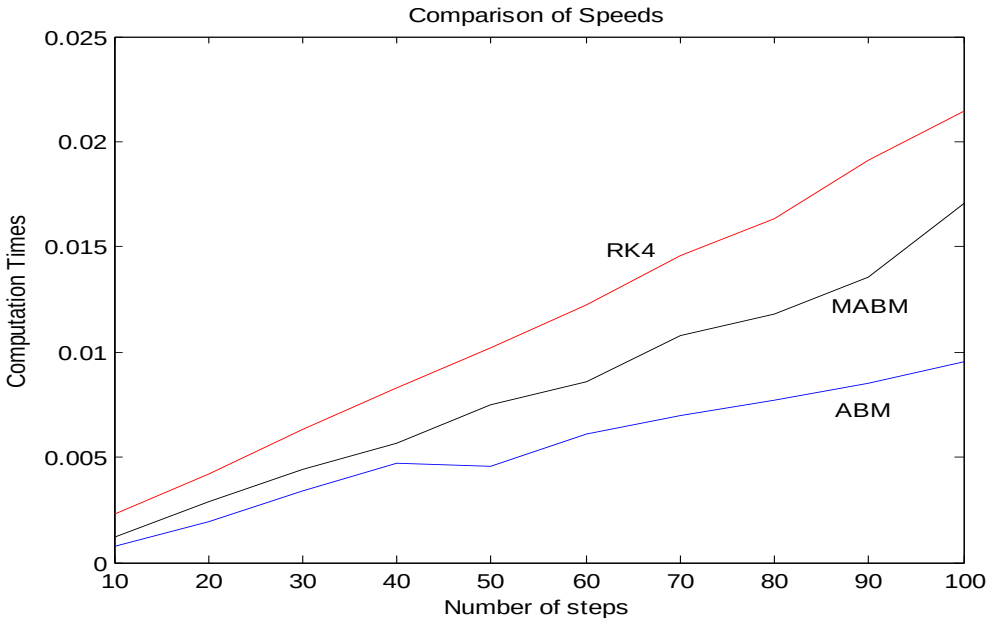


Figure 2: Plot of computation times of RK4, ABM, MABM methods by using the problem $y' = -y^3/2,\ y(0) = 1$

ABM method has the greatest speed than RK4 and MABM and MABM is the second faster in approximating the solution of the ordinary differential equation $y' = ycosx,\ y(0) = 1$ (fig.3).

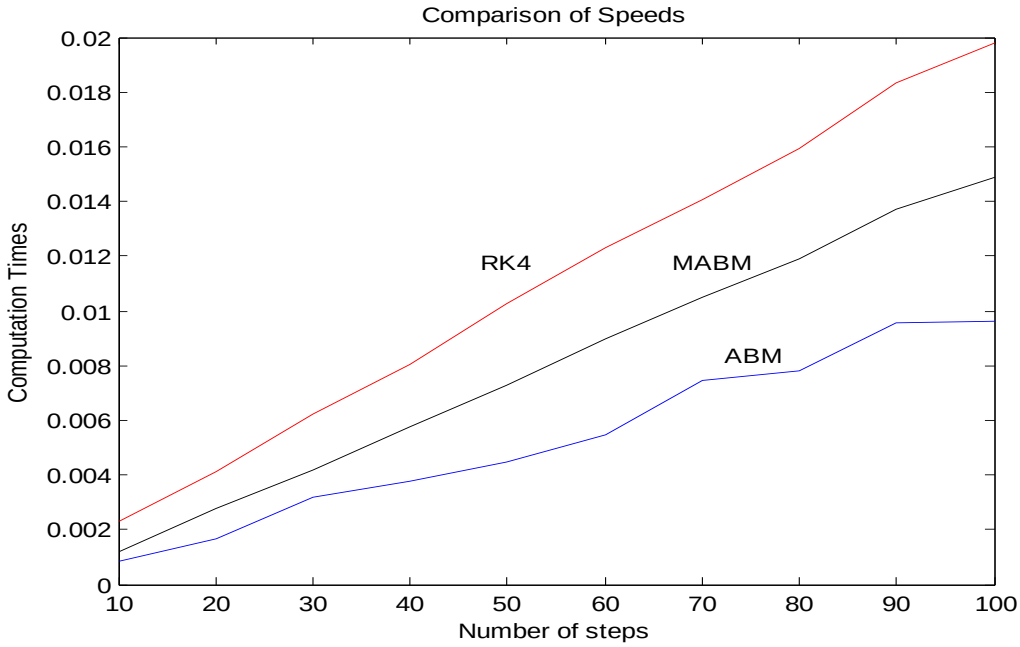


Figure 3: Plot of computation times of RK4, ABM, MABM methods by using the problem $y' = ycosx,\ y(0) = 1$

ABM method has the greatest speed than RK4 and MABM and MABM is the second faster in approximating the solution of the ordinary differential equation $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right)$, $y(0) = 1$ (fig.4).

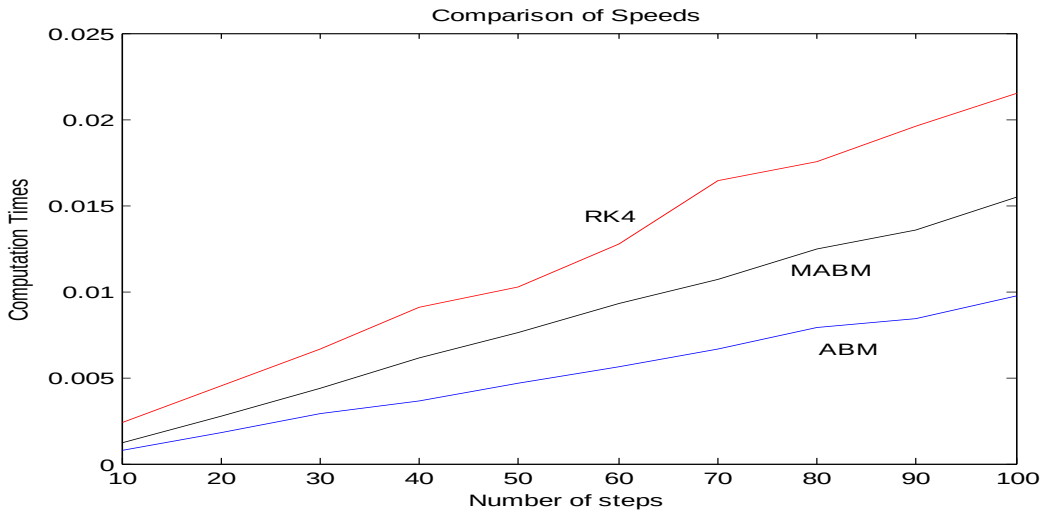


Figure 4: Plot of computation times of RK4, ABM, MABM methods by using the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right), y(0) = 1$

ABM method has the greatest speed than RK4 and MABM and MABM is the second faster (fig.5) in approximating the solution of the ordinary differential equation$y' = \frac{y-x}{y+x}$, $y(0) = 4$.

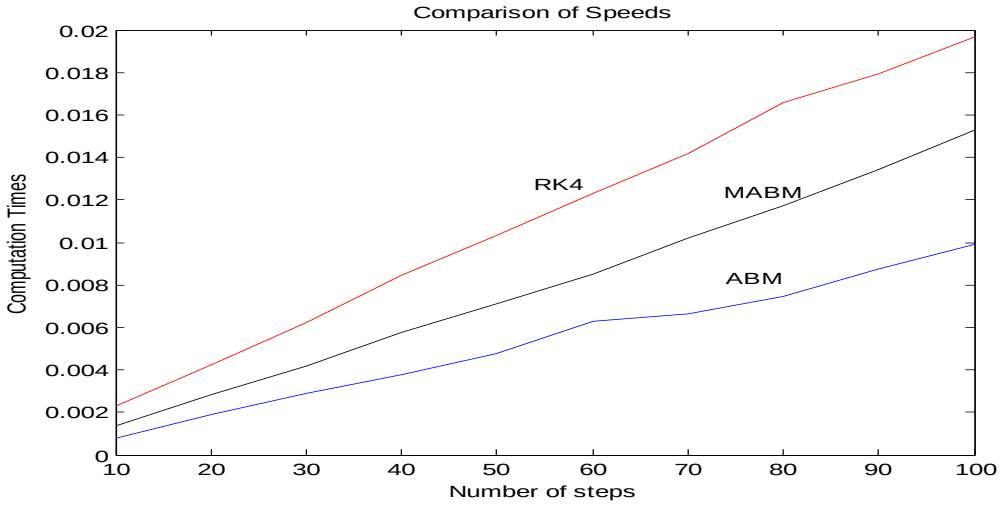


Figure 5: Plot of computation times of RK4, ABM, MABM methods by using the problem $y' = \frac{y-x}{y+x}$, $y(0) = 4$

4.1.2 Comparison of accuracy

To compare the accuracies of the RK4, ABM, and MABM methods, relative errors are computed by taking the number of steps M = 40 for the five problems considered.

RK4 method has a better accuracy than ABM method for initial value problem $y' = -y$, $y(0) = 1$ but with the help of modifiers the ABM method has a better accuracy than RK4 (table 1 and fig.6).

Table 1: Average and Standard Deviation of the relative errors obtained by RK4, ABM, and MABM methods by using the problem $y' = -y, y(0) = 1$

Method	**Average**	**Standard Deviation**
RK4	0.366583438693253	0.188061591686641
ABM	0.366583442029061	0.188061593431763
MABM	0.363985053128503	0.186498033450762

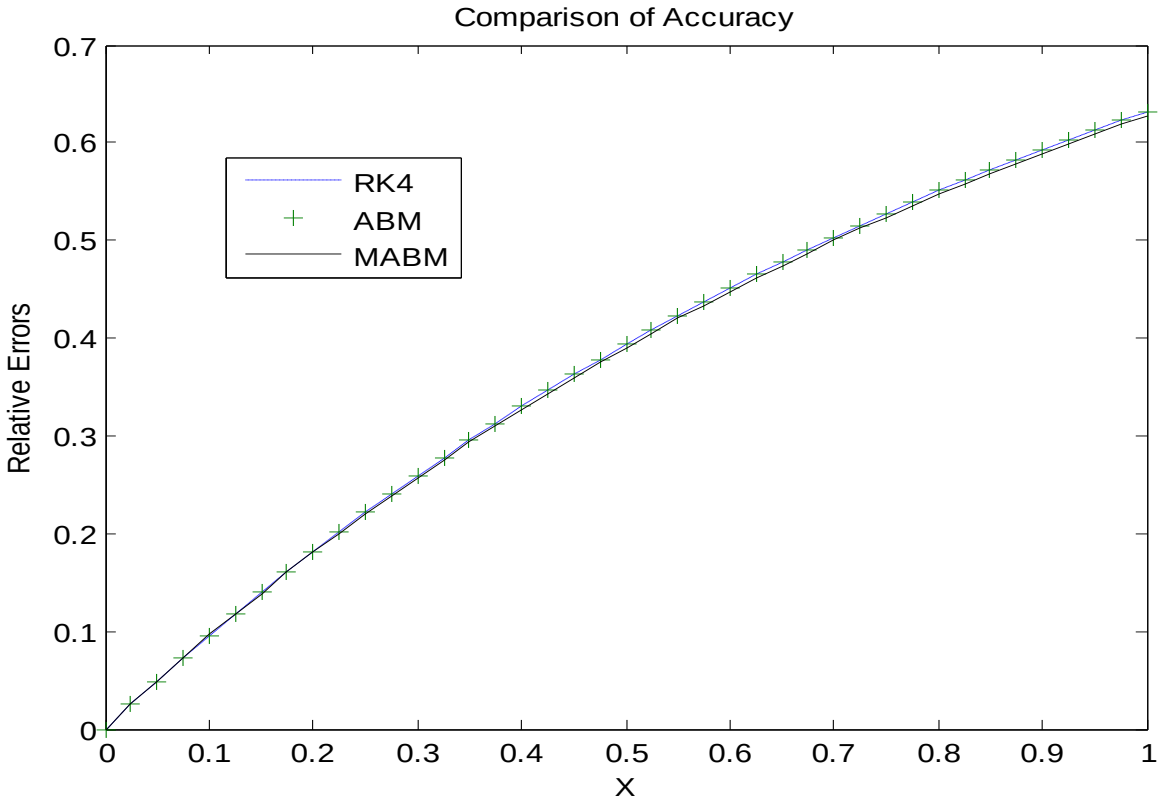


Figure 6: Comparison of accuracies of the RK4, ABM, MABM methods by using the problem $y' = -y$, $y(0) = 1$

RK4 method has a better accuracy than ABM method for initial value problem $y' = -y^3/2$, $y(0) = 1$ but with the help of modifiers the ABM method has a better accuracy than RK4 (table 2 and fig.7).

Table 2: Average and Standard Deviation of the relative errors obtained by RK4, ABM, and MABM methods by using the problem $y' = -y^3/2,\ y(0) = 1$

Method	**Average**	**Standard Deviation**
RK4	0.170943616384583	0.086130080327070
ABM	0.170943644534010	0.086130089474416
MABM	0.169627961298122	0.085395757090507

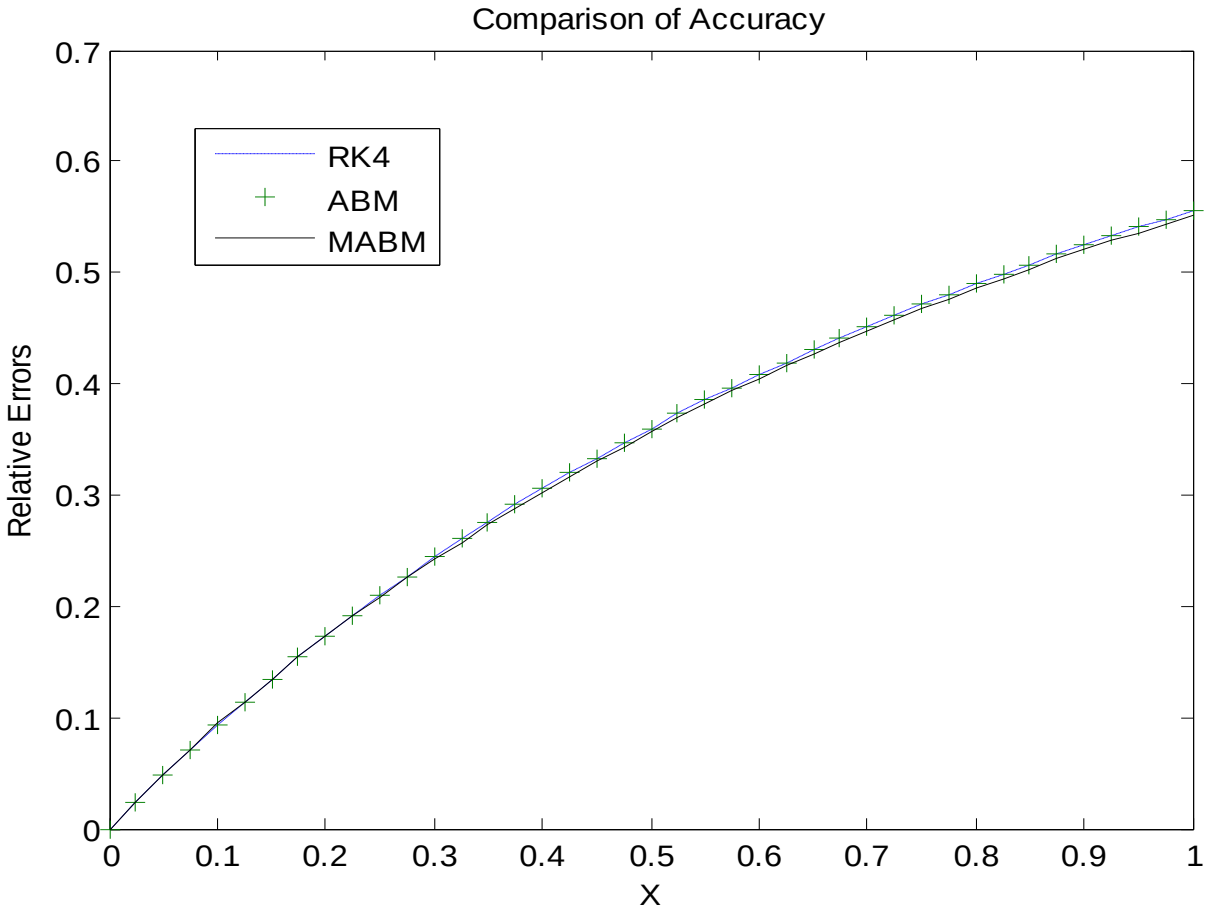


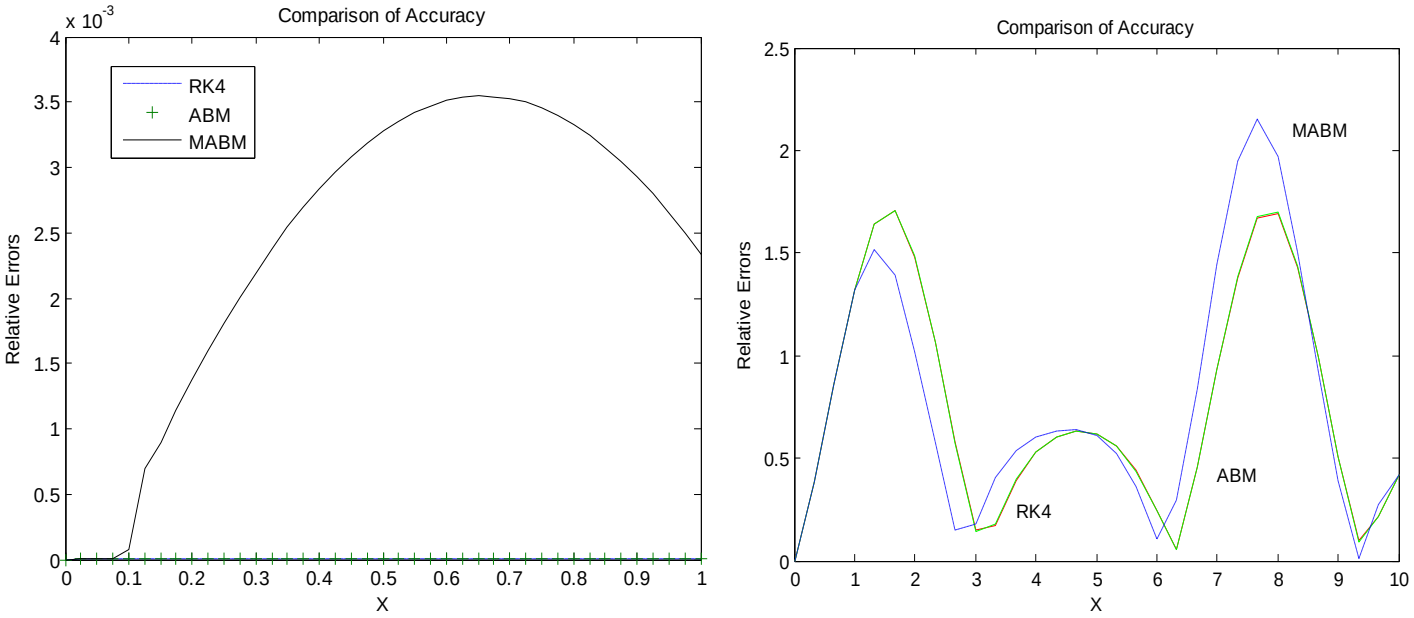
Figure 7: Comparison of accuracies of RK4, ABM, MABM methods by using the problem $y' = -y^3/2,\ y(0) = 1$

RK4 method has the greatest accuracy than both ABM and MABM methods on the interval [0, 0.475] but ABM method is more accurate than RK4 on the interval (0.475, 1] for the problem

$y' = ycosx,\ y(0) = 1$. Both RK4 and ABM methods have the same accuracy up to eight decimal places (Appendix 10). More over the usage of modifiers decreases accuracy of ABM (table 3 and fig.8a). So there is no dominant method on the interval [0, 1]. The intensity of the problem becomes much greater when we increase the interval to [0, 10] (fig.8b).

Table 3: Average and Standard Deviation of the relative errors obtained by RK4, ABM, and MABM methods by using the problem $y' = ycosx,\ y(0) = 1$

Method	**Average**	**Standard Deviation**
RK4	1.094372953711693e-09	5.046520829601356e-10
ABM	9.933909000207731e-10	6.967874828411938e-10
MABM	0.002415176583061	0.001181061887224



(a) (b)

Figure 8. Comparison of accuracies of RK4, ABM, MABM methods by using the problem $y' = ycosx,\ y(0) = 1$ on the intervals [0, 1] and [0, 10].

RK4 method has the same accuracy as the ABM method up to twelve decimal places but ABM method has a better accuracy for the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$ (Appendix 11) . More over MABM method has the least accuracy (table 4 and fig.9).

Only the graph of the MABM method is clearly shown (fig.9). The values of the relative errors for RK4 and ABM methods is very close to zero (Appendix 11). That is why their graphs seem to coincide with the x-axis.

Table 4: Average and Standard Deviation of the relative errors obtained by RK4, ABM, and MABM methods by using the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$

Method	**Average**	**Standard Deviation**
RK4	1.126954380748191e-12	6.642862614622915e-13
ABM	1.060348498426429e-12	6.939560861558076e-13
MABM	2.889908354791904e-04	1.893425057964920e-04

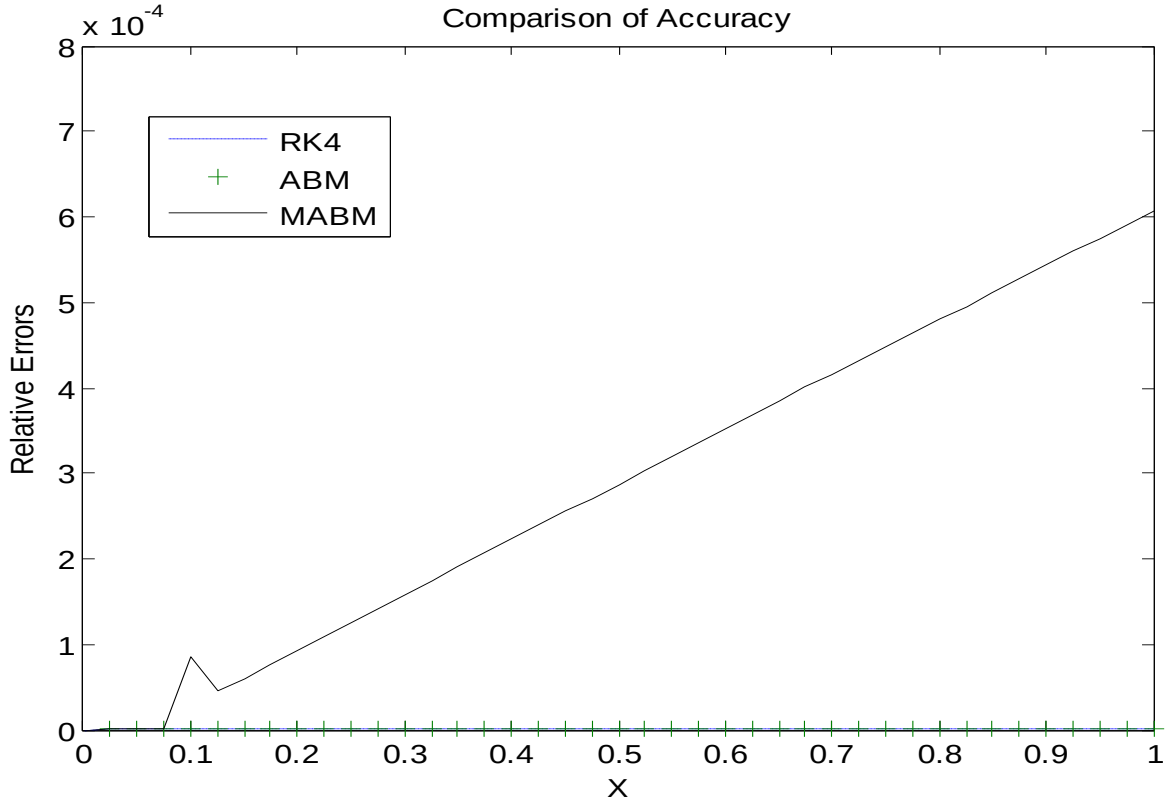


Figure 9 : Comparison of accuracies of RK4, ABM, MABM methods by using the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$

The accuracy of RK4 method is better than ABM method without modifiers but the usage of modifiers on ABM method improves the accuracy and becomes more accurate than RK4 for the

problem $y' = \frac{y-x}{y+x},\ y(0) = 4$ (table 5 and fig.10). The accuracies of RK4 and ABM are equal up to seven decimal places (Appendix 12).

Table 5: Average and Standard Deviation of the relative errors obtained by RK4, ABM, and MABM methods by using the problem $y' = \frac{y-x}{y+x},\ y(0) = 4$

Method	**Average**	**Standard Deviation**
RK4	0.107831530637488	0.060281657417218
ABM	0.107831530679423	0.060281657400593
MABM	0.075308300507318	0.035248994175990

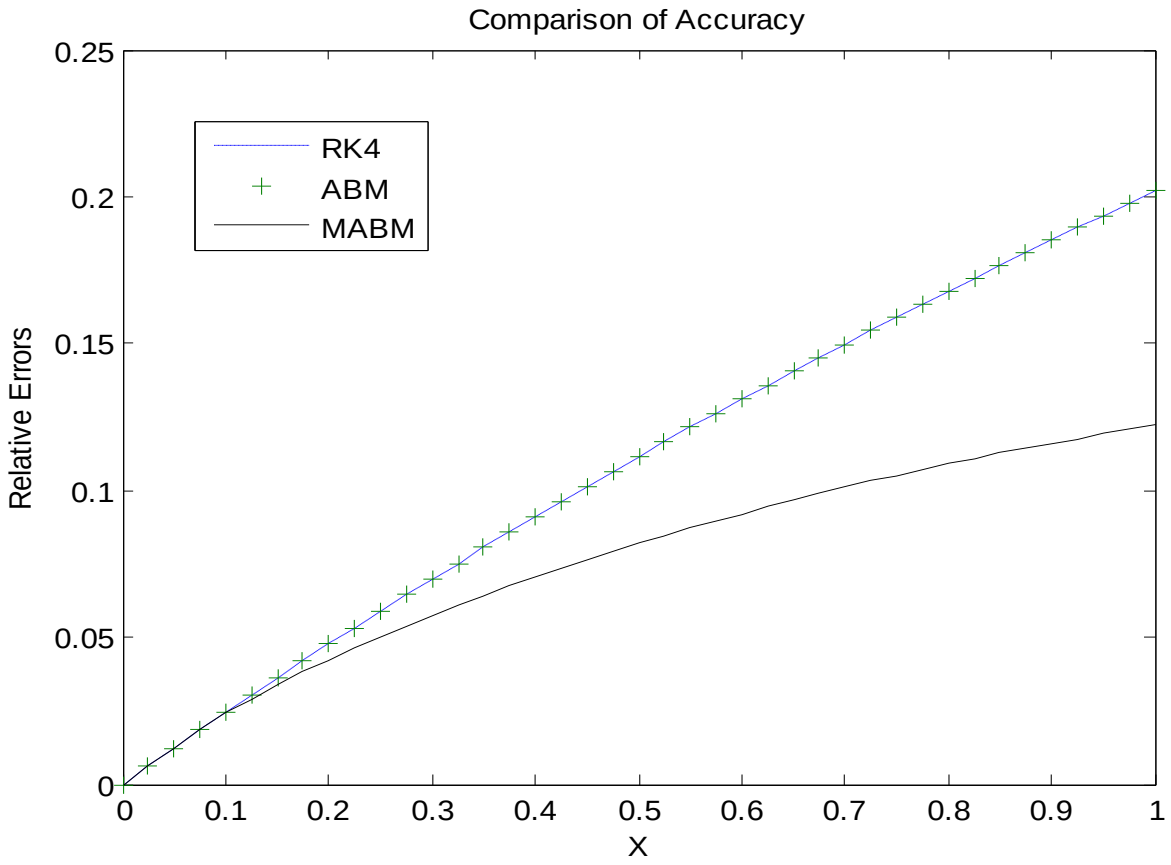


Figure 10 : Comparison of accuracies of RK4, ABM, MABM methods by using the problem

$y' = \frac{y-x}{y+x},\ y(0) = 4$

4.2 DISCUSSION

Using the combination of a predictor and corrector requires only two function evaluations of f(x, y) per step and hence unnecessary and time-consuming calculations are eliminated (Mathews. *et al*., 2004).

The ABM method registers the smallest computation time than RK4 and MABM methods. MABM method is slower than ABM method due to the addition of modifier formulas on the predictor and corrector parts but MABM is still faster than the RK4 method.

The number of iterations of the corrector is highly dependent on the accuracy of the initial prediction. Consequently, if the prediction is modified properly, we might reduce the number of iterations required to converge on the ultimate value of the corrector (Chapra and Canale, 1989).

Stiffness is a special problem that can arise in the solution of ordinary differential equations. A stiff system is one involving rapidly changing components together with slowly changing ones (Chapra and Canale, 1989). In many cases, the rapidly varying components are ephemeral transients that die away quickly, after which the solution becomes dominated by the slowly varying components. Although the transient phenomena exist for only a short part of the integration interval, they can dictate the time step for the entire solution.

The problem $y' = ycosx,\ y(0) = 1$ is a stiff differential equation as can be illustrated by the graph of its solution which showed a fast transient from y = 0 to 1 that occurs in less than 0.001166 time unit. This transient is perceptible only when the response is viewed on the finer timescale in the inset (fig.11).

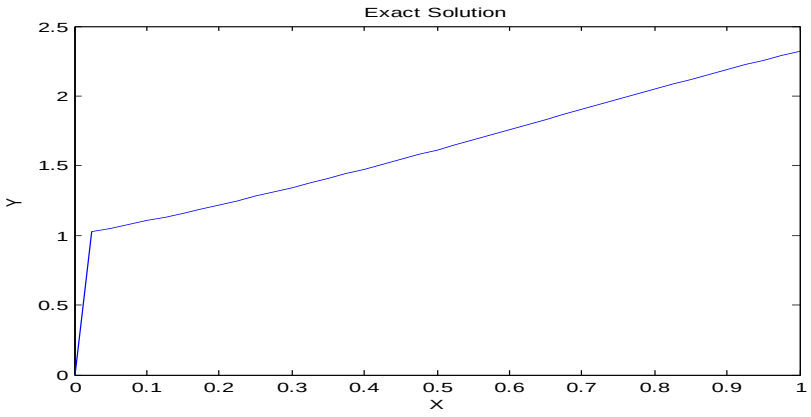
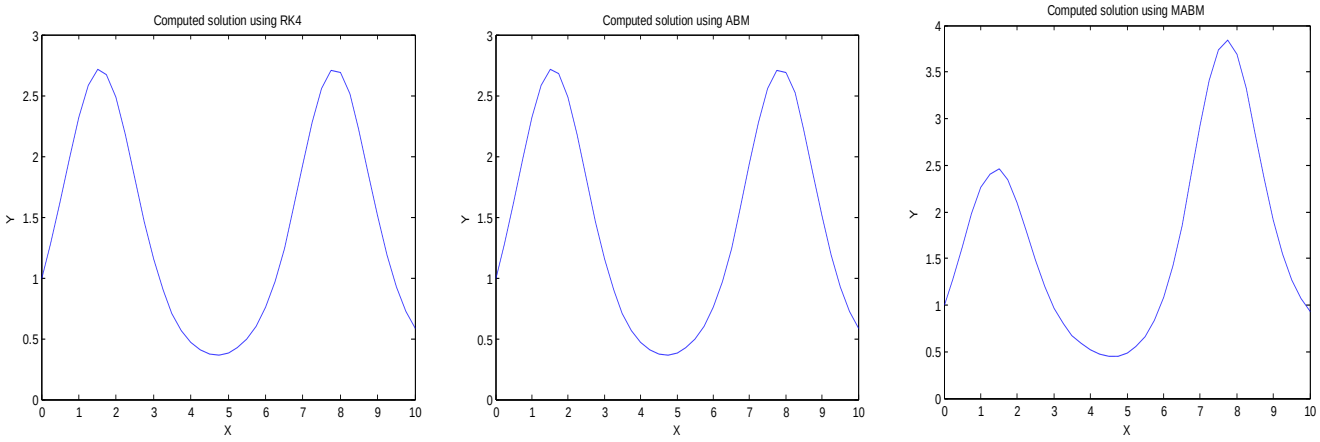


Figure 11: Plot of a stiff solution of $y' = ycosx,\ y(0) = 1$

Stiffness causes instability in uniform interval methods like RK4 unless many very small intervals are used. Stiff equations are problems for which explicit methods don't work (Hairer and Wanner as cited in Higham and Trefethen, 1993).

The high instability resulted due to the stiff nature of the given differential equation can also be clearly seen by increasing the interval from [0, 1] to [0, 10] and sketching the approximated solutions by RK4, ABM, MABM methods (fig.12). So it is not possible to compare the accuracies of the three methods as their approximated solutions manifest oscillations.



(a) **(b)** **(c)**

Figure 12: (a) Oscillations in the computed solution by using RK4; (b) Oscillations in the computed solution by using ABM; (c) Oscillations in the computed solution by using MABM

Even though the addition of the modifiers increases both the efficiency and accuracy of multistep methods, there are situations where the corrector modifier will affect the stability of the corrector iteration process (Chapra and Canale, 2010). The problem of determining when a method is stable is more complicated in the case of multistep methods, due to the interplay of previous approximations at each step (Faires and Burden, 2002).

It is due to these reasons that the use of modifiers in the ABM method decreases its accuracy for the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right)$, $y(0) = 1$. But for the problems $y' = -y$ with $y(0) = 1$, $y' = -y^3/2$ with $y(O) = 1$ and $y' = \frac{y-x}{y+x}$ with $y(0) = 4$, the modifiers are effective as they improve the accuracy of ABM method to be better in accuracy than RK4 which was previously inferior to it.

Fig. 6, fig. 7, and fig. 10 show us an interesting fact that, although the ABM method, even without modifiers, are theoretically expected to have better accuracy than the RK4 method, they turn out to work better than RK4 only with modifiers. Of course, it is not always the case, as shown in Appendix 11 where the accuracy of ABM method is better than RK4 method.

CHAPTER 5

CONCLUSION AND FUTURESCOPE

Based on the results obtained the following conclusions can be derived:

1. The computation time of ABM method is the lowest compared to RK4 and MABM method. So ABM is more efficient than RK4 method.
2. The modifiers in the corrector and predictor formulas of the ABM method are effective in improving the accuracy of ABM method in most cases but this doesn't work for some problems due to the stiffness of the problem and instability caused by the modifier in the corrector step. So the argument that modifiers are always effective for all first order ODE is invalid. But it can be concluded that such argument is valid for non-stiff first order ODEs.
3. RK4 method is more accurate than the ABM method for some problems. But this is not always the case as there is a problem where the accuracy of ABM method is greater than the RK4 method. In addition RK4, ABM, and MABM are inaccurate for some problems. So it is not possible to make generalizations. However it is possible to conclude that the performance of a given method depend on the characteristics of the ODE we are considering.
4. Explicit numerical methods do not work for stiff first order ODEs.

Future experiments can be done by increasing the types of numerical methods and extending the first order ODEs in to higher order.

REFERENCES

Abdul , S.S., Gurudeo, A.T., and Muhammad, M.S. (2013). A Comparison of Numerical Methods for Solving the Unforced Van Der Pol's Equation. *ISSN*. **3**(2):66

Atkinson, K. E. (1978). An Introduction to Numerical Analysis, 2nd ed. John Wiley & Sons, Inc, pp. 325-355.

Bedet, R. A., Enright, W. H., and Hall, T. E. (1975). STIFF DETEST: A program for comparing numerical methods for stiff ordinary differential equations. Tech. Rep. 81, Dept. of Computer Science, Univ. of Toronto, Ont., Canada. Computing Surveys, Vol. 17, No. 1, March 1985

Butcher, J.C.. (1999). Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*. **125**:1-29

Chapra, S. C. and Canale, R. P. (2010). *Numerical Methods for Engineers*, 6th ed. McGraw-Hill Co., Inc., New York, pp.751-777

Clement, A., Amy, Clement, G., Greg, F., Jeff, F., and Jayson, L. (2008). Comparing Numerical Integration Methods. *SIAM*. **2**:1–9.

Faires, J.D. and Burden, R.L.(2002). *Numerical Methods*, 3rd ed. Brooks Cole, pp. 277-280.

Gerardus, P. (2013). Comparing Accuracy of Differential Equation Results between Runge-Kutta-Fehlberg and Adams-Moulton Methods. *Applied Mathematical Sciences*. **7**: 116-127

Jorba, A. and Zou, M. (2004). A software package for the numerical integration of ODEs by means of high-order Taylor methods. *ACM Trans. Math. Software*. **8**:114-144

Hornberger, G.M. and Wiberg, P.L. (2005). *Numerical Methods in the Hydrological Sciences.* Johns Hopkins Press, Baltimore, pp.302.

Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E. (1972). Comparing numerical methods for ordinary differential equations. *SIAM J. NumeR. AnaL* **9**: 603-637.

Jain, M.K., Iyengar, S.R.K., and Jain, R.K. (1984). *Numerical Methods for Scientfic and Engeneering Computation*. New Age International (P) Limited Publishers, New Delhi, pp.471-539

Mathews, J.H., Kurtis D. (2004). *Numerical Methods Using MATLAB*, 4th ed. Pearson Education, Inc, pp.450-455.

Muhammad, A. And Arshad, H. (2013). Comparing Numerical Methods for Multi-component Gas Separation by Single Permeation Unit. *Chiang Mai J. Sci*. **41**: 184-199.

Volino, P. and Magnenat-Thalmann, N . (2000). Comparing Efficiency of Integration Methods for Cloth Simulation. *VSMM*. **5**: 109-118.

Yang, W.Y., Cao, W., Chung, T. and Morris, J. (2005*). Applied Numerical Methods Using MATLAB*. John Wiley & Sons, Inc., Hoboken, New Jersey, pp.274-277

Yatim, S. A., Ibrahim, Z. B., Othman, K. I., and. Suleiman, M. B. (2011). A Numerical Algorithm for Solving Stiff Ordinary Differential Equations. *ISSN*. **3**(2):55

APPENDICES

Appendix 1: Code for RK4

```
function R=rk4(f,a,b,ya,M)
%Input     - f is a function entered as a string 'f'
%          - a and b are left and right end points
%          - ya is the initial condition Y(a)
%          - M is the number of steps
%          - h is the step length
%Output    - R=[X',Y'] where X is the vector of abscissas
%            and Y is the vector of relative errors
f = input('Enter the function');
a = input('Enter the left end point a');
b = input('Enter the right end point b');
M = input('Enter the number of steps M');
ya = input('Enter the initial condition');
format long;
h=(b-a)/M;
X=zeros(1,M+1);
Y=zeros(1,M+1);
X=a:h:b;
Y(1)=ya;
for j=1:M
    xj=X(j);
    yj=Y(j);
    k1=h*feval(f,xj,yj);
    k2=h*feval(f,xj+h/2,yj+k1/2);
    k3=h*feval(f,xj+h/2,yj+k2/2);
    k4=h*feval(f,xj+h,yj+k3);
    Y(j+1)=Y(j)+(k1+2*k2+2*k3+k4)/6;
end
R=[X',Y'];
```

Appendix 2: Code for ABM

```
function A=abm(f,a,b,ya,M)
%Input     - f is a function entered as a string 'f'
%          - a and b are left and right end points
%          - ya is the initial condition Y(a)
%          - M is the number of steps
%          - h is the step length
%Output    - R=[X',Y'] where X is the vector of abscissas
%            and Y is the vector of relative errors
f = input('Enter the function');
a = input('Enter the left end point a');
b = input('Enter the right end point b');
M = input('Enter the number of steps M');
ya = input('Enter the initial condition');
format long;
h=(b-a)/M;
X=zeros(1,M+1);
Y=zeros(1,M+1);
Z=zeros(1,M+1);
X(1)=a;
Y(1)=ya;
for j=1:M
    xj=X(j);
    yj=Y(j);
    k1=h*feval(f,xj,yj);
    k2=h*feval(f,xj+h/2,yj+k1/2);
    k3=h*feval(f,xj+h/2,yj+k2/2);
    k4=h*feval(f,xj+h,yj+k3);
    Y(j+1)=Y(j)+(k1+2*k2+2*k3+k4)/6;
    X(j+1)=a+h*j;
    n=length(X);
    if n<5,break,end;
    F=zeros(1,4);
    F=feval(f,X(1:4),Y(1:4));
    h=X(2)-X(1);
end
for j=4:n-1
    % predictor
    p=Y(j)+(h/24)*(F*[-9 37 -59 55]');
    X(j+1)=X(1)+h*j;
    F=[F(2) F(3) F(4) feval(f,X(j+1),p)];
    % corrector
    Y(j+1) = Y(j)+(h/24)*(F*[1 -5 19 9]');
    F(4)=feval(f,X(j+1),Y(j+1));
end
A=[X',Y'];
```

Appendix 3: Code for MABM

```
function A=mabm(f,a,b,ya,M)
%Input     - f is a function entered as a string 'f'
%          - a and b are left and right end points
%          - ya is the initial condition Y(a)
%          - M is the number of steps
%          - h is the step length
%Output    - R=[X',Y'] where X is the vector of abscissas
%            and Y is the vector of relative errors
f = input('Enter the function');
a = input('Enter the left end point a');
b = input('Enter the right end point b');
M = input('Enter the number of steps M');
ya = input('Enter the initial condition');
format long;
h=(b-a)/M;
X=zeros(1,M+1);
Y=zeros(1,M+1);
Z=zeros(1,M+1);
X(1)=a;
Y(1)=ya;
for j=1:M
    xj=X(j);
    yj=Y(j);
    k1=h*feval(f,xj,yj);
    k2=h*feval(f,xj+h/2,yj+k1/2);
    k3=h*feval(f,xj+h/2,yj+k2/2);
    k4=h*feval(f,xj+h,yj+k3);
    Y(j+1)=Y(j)+(k1+2*k2+2*k3+k4)/6;
    X(j+1)=a+h*j;
    n=length(X);
    if n<5,break,end;
    F=zeros(1,4);
    F=feval(f,X(1:4),Y(1:4));
    h=X(2)-X(1);
    pold=0;
    cold=0;
end
for j=4:n-1
   % predictor
    pnew=Y(j)+(h/24)*(F*[-9 37 -59 55]');
    X(j+1)=X(1)+h*j;
    F=[F(2) F(3) F(4) feval(f,X(j+1),pnew)];
    % modifier
    pmod = pnew+(251/270)*(cold-pold);
    X(j+1)=X(1)+h*j;
    F=[F(2) F(3) F(4) feval(f,X(j+1),pmod)];
     % corrector
    cnew = Y(j)+(h/24)*(F*[1 -5 19 9]');
```

Cont'd…

```
    Y(j+1)=cnew-(19/270)*(cnew-pnew);
    pold=pnew;
    cold=cnew;
    F(4)=feval(f,X(j+1),Y(j+1));
    X(j+1)=a+h*j;
end
A=[X',Y'];
```

Appendix 4: Computation times for the three methods for the problem $y' = -y,\ y(0) = 1$

# of steps	RK4	ABM	MABM
10	0.002157	0.000867	0.001196
20	0.003954	0.001702	0.002832
30	0.006053	0.002778	0.004440
40	0.007980	0.003794	0.005625
50	0.010012	0.004736	0.007336
60	0.011991	0.005373	0.008616
70	0.014095	0.006580	0.010022
80	0.015922	0.007515	0.011385
90	0.018376	0.008117	0.013163
100	0.019617	0.008964	0.014769

Appendix 5: Computation times for the three methods for the problem $y' = -y^3/2,\ y(O) = 1$

# of steps	RK4	ABM	MABM
10	0.002286	0.000791	0.001187
20	0.004206	0.001955	0.002913
30	0.006355	0.003397	0.004434
40	0.008310	0.004751	0.005631
50	0.010225	0.004583	0.007492
60	0.012260	0.006092	0.008603
70	0.014597	0.006982	0.010806
80	0.016313	0.007723	0.011770
90	0.019082	0.008517	0.013583
100	0.021476	0.009534	0.017086

Appendix 6: Computation times for the three methods for the problem $y' = y\cos x,\ y(0) = 1$

# of steps	RK4	ABM	MABM
10	0.002317	0.000855	0.001176
20	0.004145	0.001681	0.002779
30	0.006215	0.003181	0.004198
40	0.008045	0.003760	0.005789
50	0.010244	0.004480	0.007253
60	0.012287	0.005458	0.008977
70	0.014049	0.007481	0.010519
80	0.015911	0.007787	0.011899
90	0.018361	0.009544	0.013723
100	0.019822	0.009599	0.014900

Appendix 7: Computation times for the three methods for the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$

# of steps	RK4	ABM	MABM
10	0.002380	0.000808	0.001250
20	0.004499	0.001828	0.002794
30	0.006655	0.002877	0.004381
40	0.009061	0.003659	0.006145
50	0.010248	0.004702	0.007603
60	0.012726	0.005647	0.009319
70	0.016454	0.006643	0.010669
80	0.017513	0.007885	0.012432
90	0.019616	0.008445	0.013587
100	0.021492	0.009709	0.015462

Appendix 8: Computation times for the three methods for the problem $y' = \frac{y-x}{y+x},\ y(0) = 4.$

# of steps	RK4	ABM	MABM
10	0.002316	0.000769	0.001359
20	0.004218	0.001892	0.002816
30	0.006252	0.002915	0.004206
40	0.008425	0.003785	0.005784
50	0.010333	0.004760	0.007079
60	0.012321	0.006307	0.008533
70	0.014163	0.006636	0.010199
80	0.016570	0.007443	0.011734
90	0.017915	0.008726	0.013410
100	0.019707	0.009912	0.015288

Appendix 9: Relative errors for the problem$y' = -y,\ y(0) = 1$ using the three methods

	Relative Errors		
X	**RK4**	**ABM**	**MABM**
0	0	0	0
0.025	0.024690087890625	0.024690087890625	0.024690087890625
0.050	0.048770575341203	0.048770575341203	0.048770575341203
0.075	0.072256513440178	0.072256513440178	0.072256513440178
0.100	0.095162581663294	0.095162582009953	0.096388481489407
0.125	0.117503097048754	0.117503097723587	0.117529984850853
0.150	0.139292023145825	0.139292024132696	0.138856072372295
0.175	0.160542978742516	0.160542980025634	0.159824575966977
0.200	0.181269246377766	0.181269247941900	0.180297007955011
0.225	0.201483780643456	0.201483782473948	0.200272802189024
0.250	0.221199216281459	0.221199218364202	0.219762044500648
0.275	0.240427876080758	0.240427878402179	0.238776370040683
0.300	0.259181778579592	0.259181781126638	0.257327322298864
0.325	0.277472645577439	0.277472648337551	0.275426190016955
0.350	0.295311909461512	0.295311912422618	0.293083989987828
0.375	0.312710720352384	0.312710723502875	0.310311470935830
0.400	0.329679953073168	0.329679956401888	0.327119119690347
0.425	0.346230213946640	0.346230217442866	0.343517167524069
0.450	0.362371847424532	0.362371851077965	0.359515596375270
0.475	0.378114942553157	0.378114946353902	0.375124144923372
0.500	0.393469339279386	0.393469343217942	0.390352314517046
0.525	0.408444634600937	0.408444638668181	0.405209374957934
0.550	0.423050188564811	0.423050192751988	0.419704370143429
0.575	0.437295130117625	0.437295134416334	0.433846123571940
0.600	0.451188362811504	0.451188367213683	0.447643243713996
0.625	0.464738570369085	0.464738574867007	0.461104129252458
0.650	0.477954222111134	0.477954226697386	0.474236974195023
0.675	0.490843578250140	0.490843582917618	0.487049772862143
0.700	0.503414695053220	0.503414699795118	0.499550324753384
0.725	0.515675429877549	0.515675434687346	0.511746239295193
0.750	0.527633446081462	0.527633450952915	0.523644940472965
0.775	0.539296217814302	0.539296222741434	0.535253671350225
0.800	0.550671034688010	0.550671039665103	0.546579498477676
0.825	0.561765006333367	0.561765011354949	0.557629316194803
0.850	0.572585066843743	0.572585071904584	0.568409850826644
0.875	0.583137979109137	0.583137984204237	0.578927664778289
0.900	0.593430339043196	0.593430344167778	0.589189160529592
0.925	0.603468579705881	0.603468584855383	0.599200584532537
0.950	0.613258975324337	0.613258980494405	0.608968031013623
0.975	0.622807645214490	0.622807650400969	0.618497445683585
1.000	0.632120557605816	0.632120562804744	0.627794629356713

Appendix 10: Relative errors for the problem $y' = -y^3/2,\ y(0) = 1$ using the three methods

	Relative Errors		
X	RK4	ABM	MABM
0	0	0	0
0.025	0.012270403335064	0.012270403335064	0.012270403335064
0.050	0.024099927023193	0.024099927023193	0.024099927023193
0.075	0.035514355620019	0.035514355620019	0.035514355620019
0.100	0.046537410706102	0.046537417026142	0.047139462939076
0.125	0.057190958361956	0.057190969981390	0.057208446964000
0.150	0.067495191697286	0.067495207793281	0.067224913213931
0.175	0.077468791903366	0.077468811771855	0.077026971238643
0.200	0.087129070752541	0.087129093793465	0.086539979696351
0.225	0.096492097018917	0.096492122720595	0.095769915018566
0.250	0.105572808921275	0.105572836846499	0.104728902906974
0.275	0.114385114378696	0.114385144153210	0.113429653782886
0.300	0.122941980609857	0.122942011912777	0.121884223695245
0.325	0.131255514389269	0.131255546945077	0.130103924996378
0.350	0.139337034090425	0.139337067662267	0.138099362672474
0.375	0.147197134491057	0.147197168875111	0.145880487080963
0.400	0.154845745184456	0.154845780205184	0.153456644362327
0.425	0.162292183329289	0.162292218835423	0.160836622557718
0.450	0.169545201375189	0.169545237236331	0.168028693560167
0.475	0.176613030320051	0.176613066423782	0.175040651295272
0.500	0.183503418985171	0.183503455234589	0.181879846518467
0.525	0.190223669734335	0.190223706045952	0.188553218575707
0.550	0.196780671011214	0.196780707313155	0.195067324434793
0.575	0.203180927024671	0.203180963255131	0.201428365259447
0.600	0.209430584872804	0.209430620978719	0.207642210767480
0.625	0.215535459362860	0.215535495298767	0.213714421587578
0.650	0.221501055754826	0.221501091481871	0.219650269805635
0.675	0.227332590630891	0.227332626115982	0.225454757870929
0.700	0.233035011070613	0.233035046285675	0.231132636014259
0.725	0.238613012291970	0.238613047213309	0.236688418314149
0.750	0.244071053901288	0.244071088509033	0.242126397533084
0.775	0.249413374879853	0.249413409157469	0.247450658833272
0.800	0.254644007421651	0.254644041355521	0.252665092470342
0.825	0.259766789724913	0.259766823303965	0.257773405553559
0.850	0.264785377829654	0.264785411045043	0.262779132952451
0.875	0.269703256584189	0.269703289429014	0.267685647421921
0.900	0.274523749815349	0.274523782284405	0.272496169011057
0.925	0.279250029769863	0.279250061859428	0.277213773814610
0.950	0.283885125887824	0.283885157595468	0.281841402120636
0.975	0.288431932963379	0.288431964287797	0.286381866002821
1.000	0.292893218742588	0.292893249683455	0.290837856401572

Appendix 11: Relative errors for the problem $y' = ycosx,\ y(0) = 1$ using the three methods.

Relative Errors			
X	**RK4**	**ABM**	**MABM**
0	0	0	0
0.025	0.000000000077310	0.000000000077310	0.000000000077310
0.050	0.000000000154375	0.000000000154375	0.000000000154375
0.075	0.000000000230952	0.000000000230952	0.000000000230952
0.100	0.000000000306804	0.000000001878810	0.000076704572598
0.125	0.000000000381696	0.000000001933898	0.000701330479974
0.150	0.000000000455404	0.000000001940428	0.000899613367122
0.175	0.000000000527712	0.000000001948098	0.001141975868875
0.200	0.000000000598415	0.000000001945052	0.001374908839367
0.225	0.000000000667320	0.000000001932938	0.001597030149591
0.250	0.000000000734251	0.000000001911757	0.001808494499284
0.275	0.000000000799044	0.000000001881505	0.002008992320315
0.300	0.000000000861552	0.000000001842267	0.002198238376714
0.325	0.000000000921647	0.000000001794192	0.002375964414861
0.350	0.000000000979215	0.000000001737494	0.002541916605329
0.375	0.000000001034165	0.000000001672449	0.002695856520861
0.400	0.000000001086421	0.000000001599400	0.002837561879569
0.425	0.000000001135927	0.000000001518755	0.002966827241083
0.450	0.000000001182646	0.000000001430982	0.003083464673198
0.475	0.000000001226559	0.000000001336612	0.003187304384807
0.500	0.000000001267665	0.000000001236232	0.003278195323311
0.525	0.000000001305980	0.000000001130489	0.003356005734799
0.550	0.000000001341536	0.000000001020078	0.003420623685382
0.575	0.000000001374383	0.000000000905743	0.003471957542136
0.600	0.000000001404581	0.000000000788274	0.003509936412210
0.625	0.000000001432208	0.000000000668497	0.003534510538762
0.650	0.000000001457350	0.000000000547274	0.003545651652468
0.675	0.000000001480106	0.000000000425494	0.003543353277485
0.700	0.000000001500584	0.000000000304067	0.003527630990833
0.725	0.000000001518899	0.000000000183920	0.003498522634310
0.750	0.000000001535171	0.000000000065989	0.003456088478129
0.775	0.000000001549527	0.000000000048788	0.003400411335635
0.800	0.000000001562097	0.000000000159478	0.003331596628555
0.825	0.000000001573012	0.000000000265157	0.003249772402364
0.850	0.000000001582402	0.000000000364923	0.003155089291492
0.875	0.000000001590399	0.000000000457897	0.003047720434204
0.900	0.000000001597131	0.000000000543231	0.002927861337150
0.925	0.000000001602724	0.000000000620119	0.002795729689666
0.950	0.000000001607298	0.000000000687798	0.002651565128089
0.975	0.000000001610971	0.000000000745558	0.002495628950428
1.000	0.000000001613852	0.000000000792744	0.002328203781908

Appendix 12: Relative errors for the problem $y' = \frac{y}{4}\left(1 - \frac{y}{20}\right),\ y(0) = 1$ using the three methods

	Relative Errors		
X	**RK4**	**ABM**	**MABM**
0	0	0	0
0.02125	0.000000000000026	0.000000000000026	0.000000000000026
0.04250	0.000000000000051	0.000000000000051	0.000000000000051
0.06375	0.000000000000077	0.000000000000077	0.000000000000077
0.08500	0.000000000000103	0.000000000000094	0.000062056908031
0.10625	0.000000000000128	0.000000000000112	0.000032176309877
0.12750	0.000000000000154	0.000000000000129	0.000041547789309
0.14875	0.000000000000180	0.000000000000148	0.000053389621574
0.17000	0.000000000000205	0.000000000000167	0.000065331980973
0.19125	0.000000000000230	0.000000000000186	0.000077255664318
0.21250	0.000000000000255	0.000000000000205	0.000089165607583
0.23375	0.000000000000281	0.000000000000226	0.000101061879711
0.25500	0.000000000000306	0.000000000000246	0.000112944394727
0.27625	0.000000000000331	0.000000000000267	0.000124813075748
0.29750	0.000000000000357	0.000000000000288	0.000136667845733
0.31875	0.000000000000382	0.000000000000310	0.000148508627311
0.34000	0.000000000000407	0.000000000000332	0.000160335342791
0.36125	0.000000000000432	0.000000000000355	0.000172147914167
0.38250	0.000000000000457	0.000000000000378	0.000183946263116
0.40375	0.000000000000482	0.000000000000401	0.000195730310998
0.42500	0.000000000000507	0.000000000000425	0.000207499978856
0.44625	0.000000000000531	0.000000000000449	0.000219255187413
0.46750	0.000000000000556	0.000000000000474	0.000230995857079
0.48875	0.000000000000581	0.000000000000499	0.000242721907943
0.51000	0.000000000000605	0.000000000000525	0.000254433259777
0.53125	0.000000000000630	0.000000000000551	0.000266129832035
0.55250	0.000000000000655	0.000000000000577	0.000277811543855
0.57375	0.000000000000679	0.000000000000604	0.000289478314054
0.59500	0.000000000000704	0.000000000000632	0.000301130061133
0.61625	0.000000000000728	0.000000000000659	0.000312766703276
0.63750	0.000000000000752	0.000000000000687	0.000324388158346
0.65875	0.000000000000776	0.000000000000716	0.000335994343890
0.68000	0.000000000000801	0.000000000000745	0.000347585177137
0.70125	0.000000000000825	0.000000000000774	0.000359160574998
0.72250	0.000000000000849	0.000000000000804	0.000370720454066
0.74375	0.000000000000873	0.000000000000834	0.000382264730617
0.76500	0.000000000000897	0.000000000000865	0.000393793320608
0.78625	0.000000000000921	0.000000000000896	0.000405306139679
0.80750	0.000000000000945	0.000000000000928	0.000416803103153
0.82875	0.000000000000969	0.000000000000960	0.000428284126037
0.85000	0.000000000000992	0.000000000000992	0.000439749123018

Appendix 13: Relative errors for the problem $y' = \frac{y-x}{y+x}$, $y(0) = 4$ using the three methods

Relative Errors			
X	**RK4**	**ABM**	**MABM**
0	0	0	0
0.025	0.006211259257811	0.006211259257811	0.006211259257811
0.050	0.012346294716601	0.012346294716601	0.012346294716601
0.075	0.018406929425530	0.018406929425530	0.018406929425530
0.100	0.024394906032813	0.024394906159741	0.024515538469876
0.125	0.030311891738192	0.030311891982142	0.029116958020628
0.150	0.036159482857307	0.036159483209319	0.033681485068910
0.175	0.041939209034616	0.041939209486577	0.038028040708785
0.200	0.047652537137489	0.047652537682042	0.042195954247987
0.225	0.053300874860602	0.053300875491060	0.046199158803690
0.250	0.058885574066603	0.058885574776882	0.050049497840650
0.275	0.064407933886373	0.064407934670929	0.053757321630280
0.300	0.069869203599745	0.069869204453517	0.057331733972149
0.325	0.075270585315470	0.075270586233831	0.060780789733294
0.350	0.080613236467329	0.080613237446046	0.064111654747364
0.375	0.085898272141627	0.085898273176818	0.067330736293125
0.400	0.091126767249848	0.091126768337951	0.070443790387442
0.425	0.096299758558912	0.096299759696655	0.073456010649779
0.450	0.101418246590338	0.101418247774710	0.076372102401032
0.475	0.106483197398533	0.106483198626760	0.079196344841702
0.500	0.111495544237536	0.111495545507061	0.081932643537555
0.525	0.116456189124670	0.116456190433131	0.084584574971483
0.550	0.121366004308828	0.121366005654043	0.087155424559983
0.575	0.126225833650440	0.126225835030391	0.089648219254021
0.600	0.131036493919542	0.131036495332359	0.092065755626808
0.625	0.135798776017853	0.135798777461804	0.094410624180531
0.650	0.140513446130234	0.140513447603711	0.096685230469300
0.675	0.145181246810487	0.145181248311997	0.098891813528350
0.700	0.149802898006023	0.149802899534178	0.101032462013760
0.725	0.154379098025582	0.154379099579090	0.103109128387830
0.750	0.158910524453832	0.158910526031492	0.105123641429340
0.775	0.163397835016397	0.163397836617087	0.107077717302396
0.800	0.167841668398556	0.167841670021233	0.108972969380302
0.825	0.172242645020646	0.172242646664334	0.110810916990323
0.850	0.176601367772928	0.176601369436715	0.112592993219873
0.875	0.180918422712496	0.180918424395532	0.114320551903748
0.900	0.185194379724616	0.185194381426103	0.115994873894496
0.925	0.189429793150686	0.189429794869878	0.117617172703461
0.950	0.193625202384872	0.193625204121072	0.119188599587732
0.975	0.197781132441323	0.197781134193875	0.120710248147913
1.000	0.201898094493706	0.201898096261997	0.122183158492860