COMPUTER SIMULATION OF INITIAL VALUE PROBLEMS

IKOKWU N. IBEBUIKE

PGD/MCS/232/96/97

DEPARTMENT OF MATHEMATICS/ COMPUTER SCIENCE,

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE.

SEPTEMBER 2001.

APPROVAL PAGE

THIS IS TO CERTIFY THAT THIS PROJECT HAS BEEN READ AND APPROVED BY MEETING THE REQUIREMENT FOR THE AWARD OF POST GRADUATE DIPLOMA IN COMPUTER SCIENCE IN THE DEPARTMENT OF MATHEMATICS / COMPUTER SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY MINNA, NIGER STATE.

DR. YOMI AIYESIMI PROJECT SUPERVISOR	**DATE**
DR. S. A. REJU HEAD OF DEPARTMENT	**DATE**
EXTERNAL EXAMINER	**DATE**

ACKNOWLEDGEMENTS

I would like to express my gratitude to Almighty God for preserving my life throughout my study at the F.U.T. Minna.

I am much indebted to my project supervisor, **Dr. Yomi Aiyesinmi,** for the innumerable sessions we had during which he made illuminating suggestions, which were fully utilized for the completion of this work. Special thanks go to the head of department, **Dr. S. A. Reju and Other Lecturer's** in the department, who have contributed in one way or the other for the realization of this project

Finally, I wish to thank my wife **Mrs. Agnes, Ibebuike** for her encouragement and assistance.

DEDICATION

This work is dedicated to my wife Agnes Ibebuike and my daughter Ogechi P. A. Ibebuike.

ABSTRACT

The purpose of this study was aimed at verifying and introducing a numerical computation of the solution of ordinary differential equation under initial value conditions. The numerical methods for the solution of the differential equation (dy/dx = f (x,y) y(xo) = yo) are the algorithm which will produce a table of approximation values of y(x)at certain equally spaced points called grid, nodal, net or mesh point along the x coordinate. Each grid point in terms of the previous point is given by the relationship.

$X_{n+1} = X_n + h$, n = 0, 1, 2, . . . , N-1

Where h is called the step size. The program used a single step procedure based on the Runge – Kutta method of order N = 4 (RK4).

TABLE OF CONTENT

CHAPTER TWO

CHAPTER THREE

CHAPTER ONE

1.0 NUMERICAL METHODS AND COMPUTER SIMULATION

1.1 INTRODUCTION

Humans have been calculating for thousands of years. The Pythagorean formula, an early landmark of mathematics, is a computational formula. In ancient Greece, Archimedes and others approximated II. Hundred of years ago mathematical tables were used in warfare and navigation. And yet the field of Numerical Analysis only came into being about fifty years ago, just after World War II. How did the human race avert computational disaster for all these centuries?

Even though Numerical Analysis as a separate topic is relatively new, the underlying ideas and goals are not. It is only with the invention of the electronic computer in the 1940's that large-scale automated calculations become an important tool for science and technology. This invention has two implications for us.

i) Computer arithmetic is not the same as "pencil and paper" arithmetic. In hand calculations, it is possible to monitor the intermediate results and adjust the accuracy of the

calculation as required. With computer arithmetic, each number has a fixed number of digits, which in some cases may be inadequate for a calculation.

ii) A hand calculation will usually be short, whereas a computer calculation can involve millions of steps. Tiny errors that would be negligible in a short calculation can be devastating when accumulated over a long calculation. Also, methods that are perfectly adequate for a small problem may be hopelessly inefficient when scaled to a large problem.

1.2 STATEMENT OF PROBLEM

During the past decades, giant needs forever more sophisticated mathematical models and increasingly complex and extensive computer simulations have arisen. In this fashion, two in-dissociable activities, namely mathematical modeling and computer simulation, have gained a major status in all aspect of sciences, technology, and industry.

In order that these two sciences be established on the safest possible grounds, numerical method is indispensable. For this reason,

two companion sciences: numerical methods and scientific software have emerged as essential steps for validating the mathematical models and the computer simulations that are based on them.

1.3 THE NEEDS FOR NUMERICAL METHODS

The Means by which physical situations and processes are described, analyzed, designed and simulated is through mathematics. Natural laws are stated in terms of mathematical equations, and the behaviour of systems that obey these laws is described by their solutions.

Unfortunately, the mathematics of many of the processes we would like to study quickly becomes intractable when approached by conventional means. The best that can be done, in the traditional sense, is to attempt a series expansion of the solution.

Today, there is another approach: the problem statement and variable of interest can be approximated numerically. Analysis and problem solution can then be performed through numerical computation with the aid of high-speed digital computers. In this

way, numerical computing serves as a bridge between scientific theory and practical knowledge.

Similar uses of numerical computing arise in most branches of science and technology. The design of earthquake-resistant structures, the prediction of ground water, the understanding of the inner mechanisms of the atom, the design of electron devices, the development of petroleum recovering technologies, and the interpretation of medical CAT scans are just a few of the many tasks that exploit high – speed computers to solve complicated mathematical problems numerically. In some cases, numerical computing actually aids in the exploration of new scientific principles.

1.4 **NUMERICAL COMPUTATION**

Numerical analysis involves development and evaluation of methods for computing required numerical results from given numerical data. This makes it a part of the modern subject of information processing. The given data are the input information, the required results are the output information, and the method of computation is known as the algorithm. These essential ingredients

of a numerical analysis problem may be summarized in a flow chart below:-

1.5 CHARACTERISTICS OF NUMERICAL METHODS

Numerical methods describe schemes that are used on computers, and its objective consists in obtaining a clear, precise, and faithful representation of all the "information" contained in a mathematical model.

They frequently yield only an approximation to the exact solution of the problem. However, this approximation can be refined if we are prepared to expend more computational effort in order to obtain a better accuracy.

They are conceptually and farely simple, not in volving an elaborate knowledge of mathematics and can be expressed concisely in algorithmic form.

They are readily adaptable to implementation to a digital computer.

1.6 **ALGORITHMS**

The concept of an algorithm is basic to any computational scheme, numerical or non-numerical.

An algorithm can be defined as a finite set of rules, which gives a sequence of operations for solving a specific type of problem. It has the following important features:

i. **Finiteness:** Algorithm should always terminate after a finite number of steps.

ii. **Definiteness:** Each step of the algorithm is precisely defined. This means that the rules should be consistent (contradiction-free)

iii. **Completeness:** The rules must be complete so that the algorithm can solve all problems of a particular type for which the algorithm is designed.

iv. **Input – Output:** An algorithm has certain inputs, and certain outputs that are in specific relation to the inputs.

In formulating an algorithm, we are concerned with the efficiency which is a function of:

i) Speed of solution (Economy of Operation).

ii) Stability of solution (for small errors in input, large errors in output do not occur).

iii) Accuracy of the result.

These three aspects serve as a measure for comparing two or more algorithms for solving a particular type of problem.

1.7 **FLOWCHART LANGUAGE**

One of the most convenient languages, which is effective for the communication, and description of an algorithm is the language of flowcharting.

A flowchart consists basically of a diagram of characteristically - shaped boxes connected by directed line segments. Each characteristically – shaped box usually represents a particular type of activity. The boxes represent groups of elementary steps of the algorithm. The statements in the boxes are simply the elementary steps of the algorithm. The directed lines show us the flow of the algorithm.

1.8 ITERATIVE LOOPS

A basic component of an algorithm is iteration. This word is synonymous with repetition and it indicates the repeated execution some of the elementary steps of an algorithm.

A loop usually starts after the initialization of certain quantities, and consists of mainly three types of step:

i. Computation of Elementary Steps,

ii. Test for termination;

iii. Updating or modification of Repetition

It is possible for such an iterative loop to occur with some other iterative loop.

1.9 LOGIC OF COMPUTER PROGRAMMING

A mathematical algorithm as defined in section (1.6), describes a finite sequence of operations which must be performed to arrive at the solution of a problem. This algorithm has to be converted into a list of statements or sentences which a computer can analyze and execute. Such a finite sequence of statements, which completely and unambiguously defines the sequence of operations a computer must

carry out in a calculation, is called a program. Most of these statements specify the actions to be performed by the computer, and they are known as executable statements. The remaining statements in the sequence describe the elements that appear in the executable statements, and they are called declaration statements.

The executable statements are very closely related to the algorithm. A class of executable statements that is often unnecessary in mathematical algorithms is that of input and output statements. Associated with these statements are declarations that describe the exact form of the inputs and outputs. Thus, a program is organized as follows:

Declaration of variables Declaration of input/output	(declarations)
Input statements Computations Output statements	(executable statements)

The executable statements contain both imperative and interrogative sentences, and these respectively correspond to the function box and the decision box of the flowchart. The functions and

decisions are, of course, to be chosen from the following fundamental set of operations, which can be performed by a digital computer.

i. Transferring data from external devices to the fast memory or vice versa, and from one storage to another.

ii. Executing certain basic arithmetic and logical operations.

iii. Testing whether a logical statement is true or false or a numerical quantity is positive, negative, or zero and accordingly branching out into one or more alternative paths in the program.

These operations have specific codes in a numeric form for each computer. This is called the machine language. Since the machine language is very inconvenient to use, a language called the assembly language, which uses mnemonic word (a word intended to aid the memory) rather than a numeric code, is developed. The assembler is a program that translates this assembly language into the machine language.

In order to make the task of programming easier, a class of higher-level languages has been developed. Such languages are

translated into the machine language by what is known as the compiler. The algorithmic programming languages, for example, Fortran, Pascal, Algol, and PL/1, belong to this class. It is safe to say that there is no mathematical problem that cannot be solved by using one of these algorithmic languages.

1.10 **STEPS FOR SOLVING PROBLEM ON DIGITAL COMPUTER**

The steps involved in solving a problem on a digital computer can be summarized from the foregoing discussion as follows:

i. Converting the problem into a mathematical and/or logical model if it is not already in such a model.

ii. Selecting or devising an algorithm suitable for a digital computer.

iii. Draw a flowchart for the algorithm.

iv. Based on the flowchart (or the algorithm), writing an ordered sequence of instructions, called the program, in a language the computer will recognize and accept. This process of writing computer instructions is called coding.

v. Punching the program and the data (input to the program), if any, on punched cards, and hence preparing the job deck.

vi. Making a test run on the computer. If the machine indicates coding errors or yields incorrect answers to a (test) problem or operates in an unplanned manner (such as a permanent loop), the coding should be checked and then corrected. The process of checking and correcting the codes is called debugging.

vii. Using the debugged coding for a production run.

1.11 **THE PRESENCE OF ERROR**

Several algorithms are available for producing the required output information, and we must choose between them. There are various reasons for preferring one algorithm over another, but two obvious criteria are speed and accuracy. Speed is clearly an advantage. The issue of accuracy will expose the subject, the presence of error. Rarely will input information be exact, since it ordinarily comes from measurement devices of some sort. And usually the computing algorithm introduces further error. The output information therefore

contains error from both these sources, as suggested in a second flow-chart below:

An algorithm, which minimizes error growth, clearly rates serious consideration.

1.12 **PURPOSE OF STUDY**

The aim of this study is to carry out a Computer Code based on Pascal, to simulate the solution of an initial value problem in Ordinary Differential Equations (ODEs) over the interval (a, b). The program shall use a single – step procedure, based on the Runge-Kutta method (RK 4 5).

1.13 **AREA OF STUDY**

This study is to design and carry out a numerical experiment to verify numerically the solution of the function.

$Y' = 1 + Y^2$, $Y^{(0)} = 0$ (0,1.4),

by solving an appropriate initial value problem,

using digital computer

True solution: $Y(X_k) = \tan(X)$

1.14 DEFINITION OF TERMS

ALGORITHMS: A numerical algorithm is a precise, step-by-step description of the implementation of a numerical method.

COMPUTER: A Device which is capable of accepting information, performing arithmetic and logical operations upon this information and producing the results of such operations.

CALL: A reference in a program to another program or subprogram which then assumes control of processing until all of its instructions have been executed, at which point it returns control to the original program.

DATA: Any information, numeric or non-numeric coded or literal. By input data for a problem, we mean numbers that are required to be provided in order for the solution process to start or continue. Output data are numbers generated by the solution process.

ERROR: Difference between the approximation to a quantity and its true value. An error bound is a positive number that is known to be larger than the magnitude of an error.

EXIT: The return of control to the supervisory routines after completion of a job.

FORMAT: The general arrangement of data and identification for input and output purposes.

INPUT: Data or other information in a form which may be directly transmitted to the computer.

INPUT-UNIT: A part of computer used to feed program and data.

INSTRUCTION: A command given to a computer. Is normally consists of a code to indicate the operation to be performed and address or addresses of a memory where the operand would be found.

DEBUGGING – PROCESS: The process of tracking down and correcting execution in a program.

ITERATION: Repeated process, where the input to each cycle is determined from the output of preceding cycles. The input to at least the first cycle is given in order to start the process. Criteria for stopping the iteration must be provided.

NUMERICAL ANAYSIS: The study of the solution of mathematical problems through the implementation of numbers.

PROGRAM: A series of sequential instructions in a computer language for the performance of some task.

READING: The process by which information is transmitted from a peripheral device to the Central Processing Unit (CPU).

ROUTINE: A program or program segment designed to carry out a cohesive operation.

STABILITY: A numerical is unstable if, when it is applied to a well – conditioned problem, a small change in the data results in a

large change in the numerical solution. It is stable if the change in the solution remains small.

WRITING: The process of transmitting output to a peripheral device.

SIMULATION: The representation of the behaviour of a physical system on some other system intended to imitate that behaviour.

CHAPTER TWO

2.0 ORDINARY DIFFERENTIAL EQUATION UNDER INITIAL CONDITION

2.1 INTRODUCTION TO DIFFERENTIAL EQUATIONS

Ordinary differential equations (ODE) are the principal form of the mathematical models encountered in the sciences and engineering, and consequently their numerical solution is a very large area of study. Consider the equation

$y' = dy/dx = 1-\exp(-x)$ ---------- 2.1

It is a differential equation because it involves the derivative dy/dx of the "unknown function" y=y(x). Only the independent variable x appears on the right hand side of equation (2.1); hence a solution is an anti derivative of 1-exp (-x). The rules for integration can be used to find

$y^{(x)} = x + \exp(-x)$, ---------- 2.2

Where C is the constant of integration. All the functions in eqn. (2.2) are solutions of eqn. (2.1) because they satisfy the requirement that y' (x)=1-exp (-x). They form the family of curves in fig (2,1).

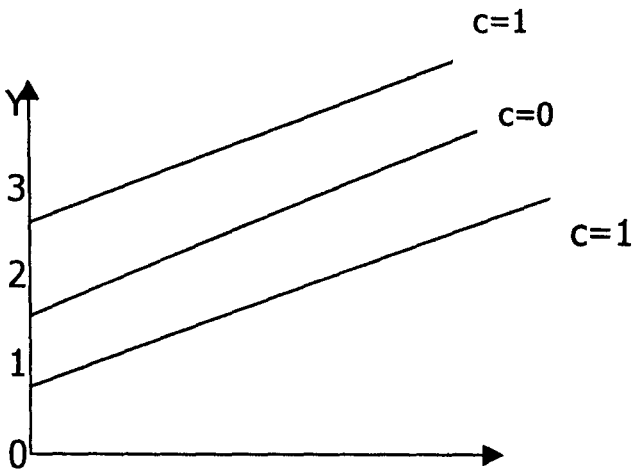

fig 2.1 The solution curves x

$$y(x) = x+\exp(-x)+c$$

Integration was the technique used to find the explicit formula for the function in eqn. (2.2), and Fig. (2.1) emphasize that there is one degree of freedom involved in the solution, namely the constant of integration C. By varying the value of C one "moves the solution curve" up or down and a particular curve can be found that will pass through any desired point.

One usually measures how a change in one variable affects another variable. When this is translated into mathematical model, the result is an equation involving the rate of change independent and/or dependent variable.

Consider the temperature y(t) of a cooling object. It might be conjectured that the rate of change of the temperature of the object is related to the temperature difference between the temperature of the object and that of the surrounding medium. Experimental evidence verifies this conjecture. Newton's law of cooling asserts that the rate of change is directly proportional to the difference in these temperatures. If A is the temperature of the surrounding medium and Y(t) is the temperature of the body at time t, then

$dy/dt = - k(y-A),$ ---------- 2.3

Where K is a positive constant. The negative sign is required because dy/dx will be negative when the temperature of the body is greater than the temperature of the medium.

If the temperature of the object is known at time t=o, we call this an initial condition and include the information in the statement of the problem. Usually we are asked to solve

$dy/dt = - k(y-A)$ with $y(o) = y_0$ ---------- 2.4

The technique of separation of variables can be used to find the solution:

$y = A + (y_0-A) \exp(-kt)$ ---------- 2.5

For each choice of y_0, the solution curve will be different, and there is no simple way to move one curve around to get another one. The initial value is a point where the desired solution is "nailed down". Several solution curves are shown in Fig. (2.2) and it can be observed that as t gets larger, the temperature of the object approaches room temperature. If y<A, then the body is warning instead of cooling

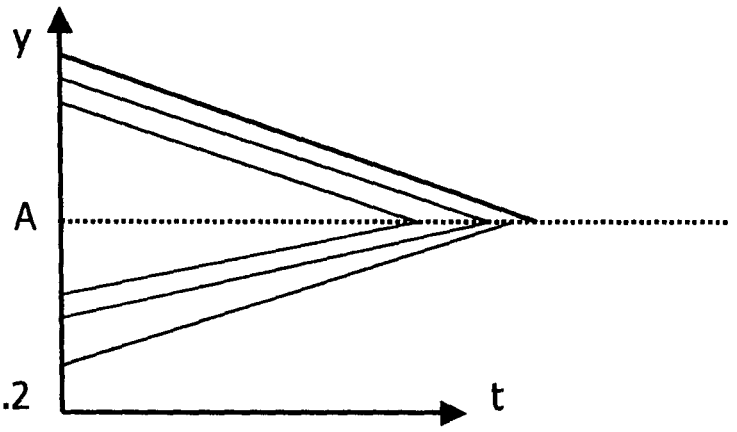

Fig 2.2

The solution I^n curves $y=A+(y_0-)\exp(-kt)$ for Newton's law of cooling (and warming)

2.2 THE GEOMETRIC INTERPRETATION

At each point (x,y) in the rectangular region R: $a \leq x \leq b$, $c \leq y \leq d$, the slope of a solution curve y=y(x) can be found by using the implicit formula $f(x,y(x))$. Hence the values $mi,j = f(xi, yi)$ can be

computed throughout the rectangle, and each value mi,j represents the slope of the line tangent to a solution curve that passes through the point (Xi,yi).

A slope field or direction field is a graph that indicates the slopes (mi,j) over the region. It can be used to visualize how a solution curves "fits" the slope constraint. To move along a solution curve one must start at the initial point and check the slope field to determine which direction to move. Then take a small step from x_0 to $x_0 + h$ horizontally and move the appropriate vertical distance hf (X_0, y_0) so that the resulting displacement has the required slope. The next point on the solution curve is (X_1, y_1). Repeat the process to continue your journey along the curve. Since a finite number of steps will be used, the method will produce an approximation to the solution.

The methods of numerical solution can be derived by various means, including the finite difference formulae and the truncated Taylor series. These derivations show that in each computation an approximation is made, and this introduces an error.

2.3 EULER'S METHOD

L. Euler was the first to devise an approximate method to solve ODEs and this method serves to illustrate the concepts involved in the advanced methods. It has limited usage because of the larger error that is accumulated as the process proceeds.

Consider a first order ODE in the symbolic form:

$\frac{dy}{dx} = f(x,y)$. IC: $y(x_0)=y_0$ ---------- 2.6

Integrating Equation (2.6), we obtain y as a function of xi thus,

$Y = \varnothing(x)$ ---------- 2.7

The graph of Equation (2.7) is a curve in the xy-plane. Since a smooth curve can be considered linear over a small length, we can write the approximate relation.

$$\frac{\Delta Y}{\Delta X} = \left(\frac{dy}{dx}\right) X = X_0 \quad \text{---------- 2.8}$$

Thus

$$\Delta Y = \Delta X \left(\frac{dy}{dx}\right) X = X_0 \quad \text{---------- 2.9}$$

so that

$$y, = y_0 + \Delta y = y_0 + \Delta X\left(\frac{dy}{dx}\right)X = X_0 \quad \text{---------- 2.10}$$

Then, the values y corresponding to $X_2 = (X_1 + h_1)$, $X_3 = (x_2 + h_2)$, ... are

$$\begin{aligned} Y_2 &= y_1 + h_1 \left(\frac{dy}{dx}\right) X = X_2 \\ y_3 &= y_2 + h_1 \left(\frac{dy}{dx}\right) X = X_2 \end{aligned} \quad \text{---------- 2.11}$$

o
o
o

Thus, we can integrate Equation (2.11) and obtain the results as a set of the corresponding values of x and y; h_1, h_2, ..., however, have to be suitably chosen.

2.4 TAYLOR'S SERIES METHOD OR METHOD OF SUCESSIVE DIFFERENTIATION.

The IVP is

$$\left[\frac{dy}{dx}\right] = f(X,y). \qquad \text{IC: } y=y_0 \text{ at } x=x_0$$

Obtain $y(x_0 + h)$

By the Taylor series, we write

$$y(x_0+h) = y(x_0) + hy'(x_0) + \left[\frac{h}{2!}\right] Y''(x_0) + \ldots' \qquad \text{---------- 2.12}$$

We have

$$\left.\begin{aligned} y' &= f(x,y),\ y'(x_0) = f(x,y)\,|\,x_0,y_0 \\ y'' &= g(x,y),\ y''(x_0) = g(x,y)\,|\,x_0,y_0 \end{aligned}\right\} \qquad \text{---------- 2.13}$$

o
o
o

Substituting the values of y′, y″, ..., in equation (2.12), we obtain $y(x_0+h)$. Again this method is not very practical. However, we can make use of it to get a rough idea of the solution.

2.5 RUNGE – KUTTA METHODS

The methods named after Carl Runge and Wilhelm Kutta are designed to initiate the Taylor series method without requiring analytic differentiation of the original differential equation. Recall that in using the Taylor series method on initial value problem

$$\left.\begin{aligned} y' &= (x,y), \\ y(0) &= y_0, \end{aligned}\right\} \qquad \text{---------- 2.14}$$

We need to obtain y″, y‴, ... by differentiating the function, f. This requirement can be a serious obstacle to using the method. The user of this method must do some preliminary analytical work before writing a computer program. Ideally, a method for solving equation (2.15) should involve nothing more than writing a subprogram to evaluate the function $f(x, y)$

RUNGE-KUTTA METHOD OF ORDER 1

The IVP is

$$\frac{dy}{dx} = f(X,y). \qquad \text{IC: } y=y_0 \text{ at } x=0$$

Find $y(x_0+h)$.

From the Taylor series, we have $y(x_0+h)$

$$= y(x_0) + hy'(x_0) + \frac{h^2}{2!} y''(x_0) + \dots .$$

An evaluation of the derivatives y′, y″, ... is avoided in the Runge – Kutta method by defining certain functions:

$$y' = \frac{dy}{dx} = f(x,y),$$

$$y'' = \frac{d}{dx}(f(x,y,)) = f_x + ff_y = \left(\frac{\partial}{\partial x} + f\frac{\partial}{\partial y}\right) f$$

$$= D,f,$$

$$y''' = f_{xx} + f_{xy}\, y' + f(f_{xy} + f_{yy}.y') + f_y\,(f_x + f_y.f)$$

$$= f_{xx} + 2ff_{xy} + f^2 f_{yy} + f_y\,(f_x + f_y f) = D_1^2 f + f_y\, D,f)$$

if $f_{xy} = f_{yx}$

Where

$$D_1^2 = \frac{\partial^2}{\partial x} + 2f\,\frac{\partial^2}{\partial x \partial y} + f^2\frac{\partial^2}{\partial y^2}$$

$$y^{(4)} = \frac{\partial}{\partial x}(f_{xx} 2ff_{xy} + f^2 f_{yy} + f_y\,(f_x + f_y f))$$

$= f_{xxx} + f_{xxy} f + 2(f_x + f f_y) f_{yy} + 2f (f_{yy} z + f_{xyy} f) + \ldots.$

It should be noted that

$f_x = \frac{\partial f}{\partial x}$, $f_{xx} = \frac{\partial^2 f}{\partial x^2}$, $f_{xxx} = \frac{\partial^3}{\partial x^3}$

$f_{xy} = \frac{\partial^2 f}{\partial x \partial y}, \ldots,$

$$y(x_0 + h) = y(x_0) + \frac{h}{2!} y'(x_0) + h^2 y''(x_0) + \frac{h^3}{3!} y'''(x_0) + \ldots$$

$$= y(x_0) + h f_0 + \frac{h^2}{2!} (f_x + f f_y)_0 + \frac{h^3}{3!} (f_{yy} + 2 f f_y + f_y (f_x + f f_y))_0 + \ldots. \quad \text{----------} \quad 2.15$$

SECOND ORDER RUNGE-KUTTA METHOD

Define

$K_1 = h f(x_0, y_0)$, $K_2 = h f(x_0 + h_0, y_0 + \beta K_1)$

Then, let

$K = \omega_1 K_1 + \omega_2 K_2$

$Y(x_0 + h) = y_0 + k \quad \text{----------} \quad 2.16$

Where α_1 β_1 ω_1 and ω_2 are constants selected in such a way that equations 2.11 and 2.12 agree up to and including the terms in h^2:

$K_2 = f(x_0 + \alpha h, y_0 + \beta k1) = f_0 + (\alpha h \, \partial/\partial x + \beta k1) f_0) + \ldots$

$\overline{h}$

$= f_0 + ch.(fx)_0 + \beta h f_0 (fy)_0 + \ldots,$

$K = W_1K_1 + W_2K_2$

Therefore,

$K = W_1 h f_0 + W_2 \{h f_0 + \alpha h^2 (fx)_0 + \beta h^2 f_0 (fy)_0 + \ldots\}$ --------- 2.17

Comparing Equations 3.10 and 3.17, we get

$$\left.\begin{array}{l} 1 = W_1 + W_2 \\ \frac{1}{2} = W_2 \alpha \\ \frac{1}{2} = W_2 \beta \end{array}\right\} \longrightarrow \begin{array}{l} \alpha = \beta \\ W_1 + W_2 = 1 \\ W_2 \alpha = \frac{1}{2} \end{array}$$

Scheme 1: we write $W_1 = 0$. Then, $W_2 = 1$,

$\alpha = \frac{1}{2}, \beta = \frac{1}{2}$. Thus

$K_1 = hf(x_0, , y_0)$

$K_2 = hf(x_0 + h, y_0 + k_1)$

$K = \frac{1}{2} k_1 + \frac{1}{2} k_2,$

$Y(x_0+h) = y_0 + k$

THIRD ORDER RUNGE KUTTA METHOD

Define

$K_1 = hf(x_0, y_0)$

$K_2 = hf(x_0 + \alpha h, y_0 + \beta k_1)$

$K_3 = hf(x_0 + \alpha h, y_0 + \beta_1 k_1 + \wp_1 k_2),$

$K = w_1 k_1 + w_2 K_2 + w_3 K_3,$ ---------- 2.18

Where α, β, α, β, $\wp$, W_1, W_2, W_3, are chosen such that Equations 2.11 and (2.12) agree up to and including the terms in h^3. Hence, the third order Runge-Kutta schemes can be written as follows:

Scheme 1

$K_1 = hf(x_0, y_0)$

$K_2 = hf(x_0 + h/2, y_0 + k½)$,

$K_3 = hf(x_0 + h/2, y_0 + k_1 + 2\,k_2)$,

$K = 1/6\,(k_1 + 4\,K_2 + K_3)$,

$Y\,(x_0 + h) = y_0 + k$

Scheme 2

$K_1 = hf(x_0, y_0)$

$K_2 = hf(x_0 + h/3, y_0 + k_1/3)$,

$K_3 = hf(x_0 + h/3, y_0 + 2\,k_2/3)$, ---------- 2.19

$K = ¼\,(k_1 + 3\,K_3)$,

$Y\,(x_0 + h) = y_0 + k$

FOURTH ORDER RUNGLE-KUTTA METHOD

The fourth order Runge-Kutta scheme is derived in the same way as the third order scheme.

Define

$K_1 = hf(x_0, y_0)$

$K_2 = hf(x_0 + h/2, y_0 + k½)$,

$K_3 = hf(x_0 + h/2, y_0 + 2\ k_2/2)$,

$K_4 = hf(x_0, + h, y_0 + k\ k_3)$,

$K = 1/6\ (k_1 + 2\ k_2 + 2\ k_3 + k_4)$,

$Y\ (x_0 + h) = y_0 + k$

As can be seen, the solution at y $(x_0 + h)$ is obtained at the expense of evaluating the function f four times. The final formula agrees with the Taylor's expansion up to and including the term h^4. The error therefore, is $0(h^4)$.

RUNGE-KUTTA-FEHLBERG METHOD (RKF45)

A more sophistated method for automatically adjusting the step size in algorithms for the initial value problem was developed by E. Fehlberg (1969). The Fehlberg method of order 4 is of Runge-Kutta type and uses these formulas: From the slopes:

$K_1 = hf(x_0, y_0)$

$K_2 = hf(x_0 + ¼h, y_0 + ¼\ k_1)$,

$K_3 = hf(x_0 + 3/8\ h, y_0 + 3/32\ k_1 + 9/32\ k_2)$,

$K_4 = hf(x_0, + 12/13h, y_0\ 1932/2197\ k_1\ - 7200\ k_2\ f7296/2197k_3),$

$K_5 = hf(x_0 + h, y_0 + 439/216k_1 - 8k_2 + 3680/513k_3 - 845/4104k_4),$

$K_6 = hf(x_0 + h, y_0 - 8/27k_1 + 2k_2 - 3544/2565k_3 + 1859/4104k_4), -11/40k_5$

We obtain two approximate solutions $k_n + 1, 4$ and $k_n + 1, 5,$

$k_n + 1, 4 = (y_n + 25/216\ k_1 + 1408/2565k_3 + 2197/4104k_4 - 1/5k_5)$

$k_n + 1, 5 = (y_n + 16/216\ k_1 + 6656/12825k_3 + 28561/56430k_4 -$

$9/50k_5 + 2/55K_6\ )$

Respectively or orders p=4 and p=5.

The error $E_n = 1 = |k_n + 1,5 - k_n + 1,4\ |/h$

$= |1/360K_1 - 128/4275K_3 - 2197/75240K_4 + 1/50K_5 + 2/55K_6\ |\ /6$

The optimal step size 5h can be determined by multiplying the scalar S times the current step six h. The scalar S is

$$S = \left(\frac{T\,O\,L}{2|K_n + 1,5 - K_n + 1, 4} \right)^{1/4}$$

$$\left(\frac{0.84\ T\ O\ L\ h}{K_n + 1,5 - K_n + 1, 4\ |} \right)^{1/4}$$

Where T O L is the specific error control tolerance.

2.6 **GILL's MOTHOD**

Gill (1951) developed a step-by-step integration procedure base on the Runge-Kutta method. This procedure has the following advantages over the other available methods:

i. It needs less storage registers

ii. It controls the growth of rounding errors and is usually stable.

iii. It is computationally economical.

let the IVP be

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \text{---} y_n)$$

$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \ldots y_n)$$

o

o

o

$$\frac{dy}{dx} = f_n(x, y_1, y_2, \text{---} y_n)$$

I . c. : at $x = x_0, y_{10}, y_2 = y_{20}, y_n = y_{no}$.

Integrate step-by-step from $x = x$ to $x_0 = x_2$ at an interval of h, i.e, obtain $y_1, y_2, \ldots y_n$ at $x = x_0$ (h) x_2.

The gill algorithm can be given as follows:

Step 1 set $x = x_0, y_1 = y_{10}, y_2 = y_{20}, \ldots, y_n = y_{no}$

Step 2 set $j = 1$

Step 3 set $x = x + h/2$

Step 4 compute $k_1 = f_i(x, y_1, y_2, \ldots, y_n)$, i = | (|)n.

Step 5 compute $y_; + h\ (a_i (k_i - bjq_i))$, $q_i = q_i + 3\ (a_i\ k_i - bjq_i))$

$- cj\ K_i$, I = | (|)n

Step 6 set $j = j + 1$

Step 7 | f j = 5, go to step 9; otherwise go to step 8

Step 8 | f J = 3, go to step 3, otherwise go to step 4

Step 9 | f $x \geq x_2$, stop; otherwise go to step 2.

2.7 **MERSON'S METHOD**

Let the I V P be

$Y'(x) = f(x, y)$, i.c: $y(x_0) = y_0$

The formulae proposed by R. H. Merson for step-by-step integration for this problem are

$Y + 1 = y_n + \frac{1}{2} (k_1 + 4k_4 + k_5) + 0(h^5)$,

$X_n + 1 = X_n + h \qquad n = 0, 1, \ldots$

Where

$K_1 = 1/3\ hf(x_n, y_n)$,

$K_2 = 1/3\ hf(x_n + 1/3h, y_n + k_1)$,

$K_3 = 1/3\ hf(x_n + 1/3\ h, y_n + ½\ K_1 + ½\ k_2)$,

$K_4 = 1/3\ h(x_n + ½h), y_n + 3/8K_1 + 9/8k_3)$,

$K_5 = 1/3\ hf(x_n + h), y_n + 3/2K_1 - 9/2k_3 + 6K_4)$.

The advantage of this method is that an estimate of the truncation error ε is given by

$5\varepsilon = k_1 - 9/2k_2 + 4K_4 - 1/2\ k_5$.

Interval changing criterion. If the right hand side of equation above is greater than five times the preassigned accuracy, then the interval h is halved and the computation of the step is begun again; but if the right-hand side is less than 5/32 of the preassigned accuracy, the interval may be doubled and the calculation for the step is repeated.

CHAPTER THREE

3.0 RESEARCH DESIGN AND METHODOLOGY

3.1 INTRODUCTION

The design stage is probably the most important stage and it outlines or defines the set of roles required for the solution of the problem. It involves the listing and ordering of successive steps and activities to be undertaken to achieve the desired goal. The tool mostly used in this stage is pseudocodes, which is used for algorithm representation.

3.2 PURPOSE OF STUDY

This research project was on attempt to verify and introduce a computer simulation of initial value to problems.

3.3 INITIAL VALUE PROBLEM OF INTEREST IN THIS STUDY

We would compute and print a table of the function

$Y^1 (X) = 1 + y^2$, $y(0) = 0$ to cover the interval (0, 1. 4)

3.4 **ALGORITHM**

The pseudocode below would be used to approximate the solution of the initial value problem

$F(x) = 1 + y^2$

Over the interval (0, 1.4) by using the formula:

$Y_k + 1 = Y_k + h/6\ (k_1 + 2k_2 + 2k_3 + k_4)$

INPUT (A, B, Y(0)) {ENDPOINT AND INITIAL VALUE}

INPUT M {NUMBER OF STEPS}

H = (B – A) / M {COMPUTE THE STEP SIZE}

X (0) = A {INITIALIZE THE VALUE}

FOR J = 0 TO M-1 DO

X = X(J) and Y(J) {Local Variables}

K_1 = N X F (X,Y) {Function value at X_1}

$K_2 = H * F(X + h/2, Y + 5*K_1)$ {Function value at $X_1 + ½$}

$K_3 = H * F(X + h/2, Y + 5*K_2)$ {Function value at X + ½ }

$K_4 = H * F(X + H, Y + K_3)$ {Function value at X + 1}

$Y (J + 1) = Y + (K_1 + 2k_2 + 2k_3 + K_4)/6$ {integrate F (X, Y)}

X (J + 1) = A + H * (J + 1) {General the mesh point}

FOR J = O TO M DO

PRINT X (J), Y (J) {Output}

CHAPTER FOUR

4.0 DATA ANALYSIS AND DISCUSSION

TABLE: RK4 SOLUTION TO $Y^1 = 1 + Y^2$, $Y(0) = 0$

K	XK	RK4 APPROXIMATION $F(X_K)$	TRUE SOLUTION $Y(X_K) = \tan(X_K)$	ERROR $Y(X_K) - F(X_K)$
0	0.00000	0.00000	0.00000	0.00000
1	0.10000	0.10033	0.10033	0.00000
2	0.20000	0.20271	0.20271	0.00000
3	0.30000	0.30934	0.30934	0.00000
4	0.40000	0.42279	0.42279	0.00000
5	0.50000	0.54630	0.54630	0.00000
6	0.60000	0.68414	0.68414	0.00000
7	0.70000	0.84229	0.84229	0.00000
8	0.80000	1.02964	1.02964	0.00000
9	0.90000	1.26016	1.26016	0.00000
10	0.00000	1.55741	1.55741	0.00000
11	0.10000	1.96475	1.96476	0.00001
12	0.20000	2.57207	2.57215	0.00008
13	0.30000	3.60156	3.60210	0.00054
14	0.40000	5.79197	5.79788	0.00091

The Runge-Kutta method (RK4) just outlined was quite stable and easily programmed, but rather wasteful of computing time.

In realistic situations involving the numerical solution of initial problems, there is always a need to estimate the precision attained numerical solution must not deviate from the true solution beyond this tolerance. Once a method is selected, the error tolerance needed on one portion of the solution curve, where as a larger one may suffice else where.

Unfortunately, this was lacking in Runge-Kutta method of order 4. The algorithm for the numerical process was given in pseudocode, and this was easy for students to translate into BASIC, C, FORTRAN OR PASCAL.

CHAPTER FIVE

5.0 SUMMARY

The research project was an attempt to verify and introduce the computer simulation of initial value problems.

In chapter one emphasis was placed on understanding why numerical methods work and the processes in computer simulations.

In chapter two, ordinary differential equation (ODE) under initial conditions were explained and various methods were explained and derived.

The desired processes were given in chapter three. In chapter four, the data obtained was compared and contrasted with the true solution (y (x).

Chapter five was a summary, conclusion and recommendation.

5.1 CONCLUSION

The Runge-Kutta method of order 4 was easily programmed andself-starting in the sense that it did not require the help of another method to start computation of y (x) at X = at X = $X_1 = X_0 +$ h, X = $X_2 = X_1 +$ h, or equivalently, it did not require the knowledge

of Y (X_1), Y(X_2), ..., Y (X_p) before computing the derivative of the function F(X,Y)

5.2 **RECOMMENDATION**

For automatically adjusting the step size in algorithms for the initial value problem, a more sophisticated method developed by E. Fehberg was recommended for more accuracy, speed and stability.

Also recommended is Gill's method of step-by-step integration procedure based on the Runge-Kutta method. This procedure has the following advantages over the other available methods:

(1) It needs less storage registers

(2) It controls the growth of rounding errors and is usually stable.

(3) It is computationally economical.

REFERENCE

1 John H. Mathews (1987)

Numerical Method for Mathematic Science and Engineering

Prentice. Hall, Engle Wood Cliffs

N. J.

2. Shampine, Lawrence F, and M. K. Gordon (1973)

Numerical Solution of Ordinary Differential Equations:

The initial. W. H Freeman and Company Publishers.

3. Scraton R. E. R.E. 984). **Numerical Methods. An**

Introduction to a Mic Com ng.

Edward Arnold Battimor

4. Rice, John Rischard (198

Numerical Methods,

Software and Analysi MSL R ence Edition.

M C Graw – Hill New York.

Sewell, Granvilla (1988)

The Numerical Solution to Ordinary Partial Differential Equations; Harcourt Brace Jovanorich, San Diego Clif.

Pearson Carl E. (1986).

Numerical Methods Engineering and Science.

Van Nostrand Reinhol ew York.

7. Gear C. William (19

Numerical Initia Problems in Ordinary

Differential Equ rentice Hall, Englewood Cliff N. J.

8. Caroll John M.

Simulation onal Computers.

Prentice Ha Cliff N. J.

ncaid.

9. Ward Ch

tics and Computing.

Nume

Brooks/Cole Publishing Company (1991)

10. M. K. Jain (1987)

Numerical Solution of Differential Equations.

Wiley Eastern Limited.

11. E. V. Krishnamurthy; S. K. Sen (1986)

Numerical Algorithms:

Computations in Science and Engineering.

Affiliated East – West Press

Private Limited.

12. Simeon Ola-Fatunla (1993)

Fundamentals of Fortran Programming.

ADA + JANE Press Nigeria Ltd.

13. **Encyclopedia of Physical Science and Technology.**

Second Edition.

Volume 11 – None – Orl

```
Program Runge(Input,Output,Fileout);
Uses graph,crt;
Var
  m,i,j:Integer;
  flag,choice:Integer;
  h,k1,k2,k3,k4:real;
  yyrad,xx,yy,lower,higher,inival:real;
  x,y:array[1..200] of real;
  ans:char;
  Fileout:text;

Function Convert_to_Rad(r : real):real;
Begin
  Convert_to_Rad := r * pi/180;
end;

Function f(a,b:real):real;
Var g : real;
Begin
  f := 1 + (b*b);
  {f := sqrt(1 - (0.25*sin(b)*sin(b)));}
end;

Procedure Clear;
Begin
   Clrscr;
   Writeln(' ':20,'================================');
   Writeln(' ':20,'     Computer Simulation of     ');
   Writeln(' ':20,'     Runge Kutta Problems       ');
   Writeln(' ':20,'================================');
end;

Procedure EnterData;
Begin
  flag := 1;
  clear;
    Writeln('Enter Lower Endpoint  : '); Readln(Lower);
    Writeln('Enter Upper Endpoint  : '); Readln(Higher);
    Writeln('Enter Initial Value   : '); Readln(Inival);
    Writeln('Enter Number of Steps : '); Readln(m);
End;

Procedure Computen;
Begin
    h := (Higher - Lower)/m;
    x[1] := Lower;
    y[1] := Inival;
    for j := 1 to m do
     begin
       xx := x[j];
       yy := y[j];
       k1 := h * f(xx,yy);
       k2 := h * f(xx+h/2,yy+0.5*k1);
       k3 := h * f(xx+h/2,yy+0.5*k2);
       k4 := h * f(xx+h,yy+k3);
       y[j+1] := yy + ((K1+2*K2+2*k3+k4)/6);
       x[j+1] := lower + h * j;
     end;
end;
```

```
Procedure Outputter;
Begin
    clear;
    Writeln;
    for j := 1 to m+1 do
     begin
       Writeln(x[j]:10:5,y[j]:10:5);
     end;
     readln;
     Rewrite(Fileout);
    Writeln(Fileout);
    Writeln(Fileout,'   x            f(x) ');
    for j := 1 to m+1 do
     begin
       Writeln(Fileout,x[j]:10:5,'   ',y[j]:10:5);
     end;
     close(Fileout);
End;


Procedure Computev;
Begin
  if flag = 0 then
     Begin
       clear;
       gotoxy(20,10);Write('Empty Data !, Select options 1 before 2 ');
       Delay(3500);
     end
  else
     Begin
        clear;
        Computen;
        Outputter;
       Write('Press Enter to Continue ');readln;
     end;
End;

Begin
 flag := 0;
 assign(Fileout,'out.out');
repeat
 repeat
  clear;
  gotoxy(25,5);Writeln('M A I N   M E N U');
  gotoxy(25,6);Writeln('******************');
  gotoxy(23,8) ;Writeln('1. Enter Variables ');
  gotoxy(23,12);Writeln('2. Computation     ');
  gotoxy(23,16);Writeln('3. Quit Program    ');
  gotoxy(25,19);Write('Enter Choice (1-3) ');readln(choice);
 until (choice > 0) and (choice <= 3);
  Case Choice of
     1:EnterData;
     2:Computev;
  end;
until choice = 3;
end.
```

x	f(x)
0.00000	0.00000
0.10000	0.10033
0.20000	0.20271
0.30000	0.30934
0.40000	0.42279
0.50000	0.54630
0.60000	0.68414
0.70000	0.84229
0.80000	1.02964
0.90000	1.26016
1.00000	1.55741
1.10000	1.96475
1.20000	2.57207
1.30000	3.60156
1.40000	5.79197