ISSN 0252-1075
Research Report No. RR-068

Contributions from
Indian Institute of Tropical Meteorology

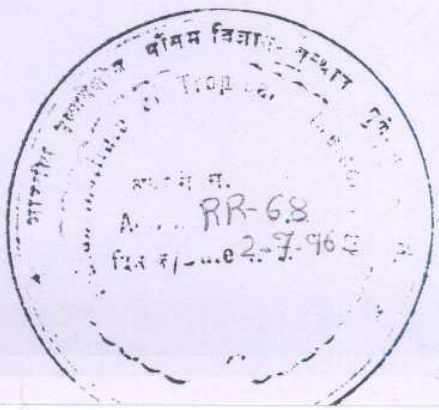
SOME NUMERICAL EXPERIMENTS ON ROUNDOFF-ERROR GROWTH IN FINITE PRECISION NUMERICAL COMPUTATION

by

SUVARNA FADNAVIS

PUNE - 411 008
INDIA

MAY 1996



Some Numerical Experiments on Roundoff-Error Growth in Finite Precision Numerical Computation

Suvarna Fadnavis

Indian Institute of Tropical Meteorology, Pune 411 008, India

Abstract

Finite precision computations using digital computers involve the following inherent errors: (1) Roundoff-error of finite precision computations (2) Binary computer arithmetic precludes exact number representation of traditional decimal system used in data input stage to the computer.

Though the roundoff-error is as small as 10^{-7} (single precision) in the beginning, it can enter the mainstream computation within 50 iterations in iterative computations, such as that used in numerical integration schemes, for example, the commonly used fourth order Runge-Kutta method.Growth of roundoff-error in recursive vis-a-vis iterative computing is described in the text.

In this paper several computational experiments are presented to demonstrate the rapid growth of roundoff-error in iterative computations.

1. Introduction

Taxonomy of computer errors

There are two types of errors in computer

i) Software errors

ii) Hardware errors

Since this study is software based, only software error are given below.

a) Error due to finite representation of numbers

e.g. 1/3 is not exact representable using finite number of digits[11].

b) Error due to arithmetic operation using normalized floating point numbers such error is called rounding error[3].

e.g. 2.407
.005 Number to half adjust
2.41[2 digit to be dropped

c) Error due to conversion of real number from decimal to binary number system
e.g. 32 bit computer stores 0.1 as 0.0999999

Among all types of errors introduced in numerical computations, errors of roundingoff numbers due to finite precision is important. It is difficult to estimate this roundoff-error.

Roundoff Error[9] is defined as the error due to arithmetic operation using normalized floating point number, such error is also called rounding error. The technique of half adjust is used to round to the least significant digit to be kept. 5 is added to the digit to be dropped, if digit to be dropped is equal to or greater than 5 rounding or carry of 1 is forced on the next digit to the left.

e.g. 2.407
.005 Number to half adjust
2.41[2 digit to be dropped

Truncation error[11] is defined as the error due to finite representation of an inherently infinite process.

The use of finite number of terms in infinite series

$$(1+x)^n = 1+nx+\frac{n(n-1)}{2!}x^2+\frac{n(n-1)(n-2)}{3!}x^3+\ldots\ldots \quad (1)$$

Use of 30 terms of the above series gives truncation error.

Computers convert data, traditionally decimal numbers to binary numbers, as coded numeric data. Any integer number is represented in binary form as

$$I = \sum_{j=1}^{L} b_j 2^{L-j} \quad (2)$$

where b_j = j^{th} most significant bit in representation of I where each bit is either '0' or '1', depending on position of switch either **on** or **off**.

Internal storage I can be represented schematically as

$$b_1 b_2 b_3 b_4 \ldots . b_L$$

The number L of binary elements required to code 'I' is called length of representation.

Any real number can be represented by a binary number X given as

$$X = \pm[(b_n 2^n + b_{n-1} 2^{n-1} + \ldots . b_1 2^1 + b_0) + (b_{-1} 2^{-1} + b_{-2} 2^{-2} + \ldots . b_{-k} 2^{-k})] \quad (3)$$

b_n= n^{th} significant bit

n and k are any integer numbers[2].

All real numbers cannot be stored exactly on k bit as mantissa. A real number **X** is represented as

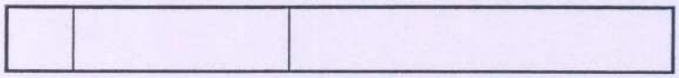
$$X = \pm f.2^e \quad (4)$$

where e = exponent, f = fraction

in which (**X**)error is imbedded, that is a conversion error due to finite word length of computer may be introduced, particularly if there is no exact binary equivalent. For example 0.1 in decimal system is stored as 0.09999999404. Error is introduced during decimal to binary conversion. Also there is hardware limitations for storage of digits. 23 bit mantissa device stores numbers to seven accurate significant digits. Hence roundoff- error is introduced during decimal to binary conversion and due to limited machine storage capacity.

Word-length of computer and computer precision are dependent terms, computer precision is fixed for particular word length of computer

e.g. for 32 bit computer, precision is calculated as follows



1 bit sign 8 bit exponent 23 bit mantissa

$$2^{-23} = 10^{-N} \quad (5)$$

$$23 \log 2 = N \log 10$$

$$7 = N \quad \text{for single precision}$$

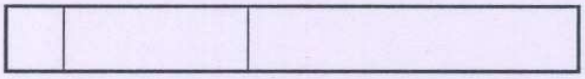
In double precision one word is added to prevision word. As sign bit and exponent is same, the only change is in mantissa i.e. of 55 bits. Therefore number of bits can be calculated as

$$2^{-55} = 10^{-N} \quad (6)$$

$$55 \log 2 = N \log 10$$

$$17 = N$$

For NEC system 1000 computer[9] word length is 36 bits, number of bits for single precision can be calculated as



1 sign bit 8 bit exponent 27 bit mantissa

$$2^{-27} = 10^{-N} \quad (7)$$

$$27 \log 2 = N \log 10$$

$$8 = N$$

for double precision mantissa will be of 63 bits

$$2^{-63} = 10^{-N} \quad (8)$$

$$63 \log 2 = N \log 10$$

$$19 = N$$

more number of bits are available in NEC system 1000 computer for single and double precision to represent a number therefore less roundoff error as compared to 32 bit computer.

Roundoff-error increases as number of terms increases in the algorithm which is being computed. Truncation error is due to truncation of terms in the algorithm, and it decreases as number of terms increases. The nature of roundoff-error and truncation error versus number of terms and the optimum number of terms for minimization of errors has been discussed[1,5].

The nature of floating point numbers makes it desirable to control, bound, or estimate, the rounding error of floating point algorithm. The most influential technique for studying rounding error is floating point error analysis. The four basic floating point operations are addition, subtraction, multiplication, division. Result of operations is stored in a machine-register which has greater precision than ordinary floating point number. Therefore inaccuracies are only in the rounding of results[2,3].

In so far as roundingoff-error is concerned in any case the answer in the register is in error by at the most one in that place i.e 2^{-d+1}. Rounding the result for storage may produce a relative error of 2^{-t} in single precision value (t=places, d= digit)[2].

Arithmetic operations of addition, substraction, multiplication and division are done with respective highest exponent. Hence smaller the value larger is the error.

The roundoff-error results from combined effects of inherent roundoff and propagated roundoff. Propagated roundoff due to addition of small quantities (negligible addition) results in error magnification, loss of significance[6].

All the computations of this study are carried out on HP 9000/735 workstation.

Word length of computer used is 32 bits.

2. Experiments to test Roundoff-errors

Propagation of roundoff-error from initial value in numerical computations were studied by performing following four experiments

a) Bernoulli shift

b) Iterative addition

c) Runge-Kutta integration schemes

d) calculation of square-root two ($\sqrt{2}$)

2.1 Bernoulli shift[10,12,13]

$$X_{n+1} = \text{Mod}\,(2 * X_n) \qquad (9)$$

n = number of iterations

X = any number

As the process is iterative, error magnifies as number of iteraions increases and within 50-60 iterations error becomes larger than the true value. Experiments were done for X_1 = 0.1, 0.01, .001, .0001) X_1 is the initial value of X.

Fig.1 shows computed value departs from true value from 50th iteration itself. Computed value becomes zero within 60-65 iterations.

Error grows slowly and enters the mainstream within 50 iterations and results become totally incorrect after 60-65 iterations.

Mary Selvam[8] has shown that roundoff error approximately doubles on an average for each iteration. Therefore for single precision (10^{-7} accuracy) computation, the iteration number (N) at which roundoff error enters the main stream (unit place) can be computed as follows

$$2^N .10^{-7} = 1 \qquad (10)$$

$$N \log 2 = 7 \log 10$$

$$N = 7 / \log 2$$

$$N = 23$$

For double precision (10^{-14} accuracy) computation error enters in the main stream is computed as follows

$$2^N .10^{-14} = 1 \qquad (12)$$

$$N \log 2 = 14 \log 10$$

$$N = 14 / \log 2$$

$$N = 46$$

Therefore results become incorrect after 23 iterations, as error enters in main stream in single precision and after 46 iterations in double precision.

If we relate error to noise, error becomes quite significant, it is as good as saying that noise will be of same scale as signal and shall be difficult to detect the signal. Even though word length is increased, error enters in the main stream after larger number of iterations. The only way to minimise it is to keep noise much smaller than signal. Either large iterative computations should be avoided or an alternate method should be used.

2.2 Iterative addition (successive addition)

$$X_{n+1} = X_n + \Delta X \qquad (13)$$

ΔX = small increments in X

n = integer number

X = any number

Propagation of roundoff-error by successive addition of the small increments as 0.1, $1x10^{-2}$, 1×10^{-3}, 10^{-7} one million times (10^6) were studied and listed in Table 1.

Table 1 : Propagation of roundoff error by successive addition

X_1	ΔX	Number of iterations	Computed X_{n+1}	Actual X_{n+1}	Error
0.1	0.1	10^6	$.9851239 \times 10^{+5}$	10^5	1487.61
0.01	0.01	10^6	$.9815246 \times 10^{+4}$	10^4	184.754
0.001	0.001	10^6	$.9838521 \times 10^{+3}$	10^3	17.1476
10^{-7}	10^{-7}	10^6	.1006866	0.1	-6.866×10^{-6}

X_1= initial value

Error decreases as X_1 decreases , but it is high for x = 0.01, 0.1 i.e for larger values of X_1

2.3 Runge-Kutta integration schemes

Runge-Kutta methods are self starting. Given an initial point next point in the solution curve may be quickly interpolated.

A variable Y which is an unknown function of X is computed in terms of rate of change $(\frac{dY}{dX})$ which is given as function of Y and X.

Based on Newtonian continuum dynamics it is assumed that $(\frac{dY}{dX})$ is continuous for infintesimally small increments dX. Starting from a given an initial coordinates (X_1 and Y_1) the incremental change ΔY in Y corresponding to incremental change ΔX in X is computed as

$$\Delta Y = (\frac{\Delta Y}{\Delta X})_1 . \Delta X \qquad (14)$$

where $(\frac{\Delta Y}{\Delta X})_1$ is slope 'S' equal to $\frac{dY}{dX}$ at the intitial location (X_1, Y_1). Numerical computations impose finite magnitudes for ΔX and ΔY values. Therefore Newtonian continuum dynamics is no more valid.

The value Y_2 of **Y** corresponding to $(X+\Delta X)$ is now given as

$$Y_2 = Y_1 + \Delta Y \tag{15}$$

The new coordinates are designed as Y_2 and X_2

$$Y_2 = Y_1 + \Delta Y \tag{16}$$

$$X_2 = X_1 + \Delta X \tag{17}$$

The successive value of Y corresponding to successive incremental changes ΔX in X are therefore computed iteratively i.e.

$$Y_{n+1} = F(Y_n) \tag{18}$$

where Y_{n+1} the $(n+1)^{th}$ value of **Y** is a function F of Y_n. Such iterative evaluation of **Y** is known as numerical integration. To obtain an accurate estimate of the evolution of **Y** with respect to **X** the fourth order Runge-Kutta integration scheme is commonly used.

A brief description of fourth order Runge-Kutta scheme[4,11].

In this method, slope S, is first evaluated at the intitial location (X_1, Y_1) the next location (X_2, Y_2) is then determined where $X_2 = X_1 + h$, $h = \Delta X$

The slope S_2 at location $(X_1+h/2, Y_1+S_1h/2)$ is then used to compute slope S_3 at location $(X_1+h/2, Y_1+S_2h/2)$ similarly slope S_4 at the fourth loation (X_1+h, Y_1+S_3h) is computed. The weighted average S_{av} of four slopes is then used to compute the next location (X_2, Y_2).

$$S_{av} = \frac{h}{6}(S_1+2S_2+2S_3+S_4) \quad (19)$$

$$Y_2 = Y_1+\frac{h}{6}(S_1+2S_2+2S_3+S_4) \quad (20)$$

$S_1 = f(X_1, Y_1)$ $S_2 = f(X_1+h_2, Y_1+S_1h/2)$

$S_3 = f(X_1+h/2, Y_1+S_2h/2)$ $S_1 = f(X_1+h, Y+S_3h)$

Description of second order scheme[11].

Numerical integrations can be done using second order Runge-Kutta method. Where slopes $S_1 = f(X_1, Y_1)$ and $S_2 = f(X_1+h, Y_1+S_1h)$ alone determine the evolution of Y with X. Second order is less accurate than fourth order method. Therefore second order is not in general use.

In order to estimate the error in Runge-Kutta integration methods the following four different algebric relationship between Y and X were studied.

$$Y = \frac{4}{4-2X-X^2} \quad (21)^7$$

$$Y = 2X^2 \quad (22)$$

$$Y = 2X^4 \quad (23)$$

$$Y = 2X^2+2X^4 \quad (24)$$

The percentage error of each step of integration was computed as $(\frac{Yexact-Y}{Yexact})x100$, where *Yexact* is the true solution as obtained from equations (i-iv) and Y is corresponding computed solution.

Integrations were performed in order to estimate the accuracy as follows.

(i) The relative accuracies of second and fourth order Runge-Kutta methods.

The percentage error for second and fourth order Runge-Kutta methods for the above four equations were computed for step size 0.1 and shown in Fig.2.

The fourth order method is more accurate than the second order method. For complex equations error becomes very large (>100%) within 35 iterations even in fourth order Runge-Kutta iterations. In simple algebraic equations error growth is rapid upto 5-6 iterations then percentage error remains constant and is less than 100%.

(ii) The influence of step size in relation to initial conditions for the fourth order Runge-Kutta methods was investigated by integrating the four equations for two different step sizes. Results are shown in Fig.3.

In fourth order Runge-Kutta method when step size is same as initial condition i.e. X=0.1 h=0.1 percentage error is small but increases to 15% within 4-5 iterations for simple algebraic equations. In complex equations percentage error is negligible upto 10-30 iterations but it grows rapidly (>100%) within 40 iterations.

When step size is 10 times the initial condition i.e. X =0.01 h=0.1 percentage error is very large showing the same nature of curve for simple and complex equations.

Fourth order Runge-Kutta method is sensitive to step size and shows very large error when step size is larger in magnitude than the initial value of X.

2.4 Square-root two ($\sqrt{2}$) computation

The value of square-root two is computed by three different techniques

(a) Continued fraction

(b) Binomial series

(c) Manual calculation

The description of the three methods is as follows

(a) Continued fraction

From the equation $(X+1)^2=2$ (25)

$\sqrt{2} = (X+1)$

The evaluation of X is as follows:

$(X+1)^2 = 2$

$X^2+2X+1 = 2$

$X^2+2X = 1$

$X(X+2) = 1$

$$X = \frac{1}{2+X}$$ (substituting for X)

$$X = \cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+.....}}}}$$

The value of $\sqrt{2}$ remains constant after 20th term and is listed in Table 2.

(b) Binomial series[7]

$$(1+X)^n = 1+nX+\frac{n(n-1)}{2!}X^2+\frac{n(n-1)(n-2)}{3!}X^3+..... \quad (26)$$

To calculate square-root two n=1/2 X=1

To get accurate value number of terms of series should be large so that the truncation error is small. The value of $\sqrt{2}$ was obtained using 15000 terms and is given in Table 2.

(c) Manual calculation

Arithmetical calculations of square-root is used

	1.4142
1	2
	1
24	100
	96
281	400
	281
2824	11900
	11296
	60400
	....

√2 value is computed upto 14 decimals and is given in Table 2.

Table 2 : Computaion of √2 value

Method of computation	Number of Terms	Results(√2) double precision
	Computer-Arithmetic	
Continued fraction	20	1.4142135623731
Binomial series	15000	1.41421361170650
Manually calculated	Manual-Arithmetic	
	14 decimals	1.4142135623731

The continued fraction method gives exact value as manually calculated one because of exact decimal to binary conversion of 1/2 without roundoff-error. But binomial series gives error due to roundoff and truncation. The number of terms is very large and therefore truncation error is small, major error is due to rounding of numbers.It was also found that growth of roundoff-error is more in recursive computing than in iterative computing.

We all know that like many other electronic devices, a computer also has some inherent problems, viz. errors. In spite of such inherent problems, computers are considered to be indispensable in any field of modern world. Domain of usefullness of a computer is an ever increasing one.Since this is a computer based study I wish to conclude this study with the following widely acclaimed statement*.

"IN SPITE OF THE ASSOCIATION OF DIFFERENT TYPES OF ERRORS WITH IT, A COMPUTER CONTINUES TO EXERT A LINGERING FASCINATION FOR ENTIRE HUMAN CIVILIZATION".*

3.0 Conclusion

From figures (1-3) and Tables (1-2) it is seen that increase in roundoff-error of finite precision computation increases exponentially in iterative computations and enters the mainstream computation within 50-60 iterations. Long-term numerical integration schemes which incorporate iterative computations will therefore give erroneous results.

Acknowledgement:

The author is grateful to Dr. A.M. Selvam and Shri.M.K.Tandon for valuable guidance and encouragement for the study of these numerical experiments on growth of roundoff-error.

* This widely acclaimed statement was brought to my notice by Shri. M.K.Tandon

References

[1] Alt F.L, *Electronic digital computers.* Academic Press Inc., New York (1958), pp 330.

[2] Chambers J.M., . *Computational methods for data analysis.*John Wiley and Sons Inc., Canada (1977), pp 227.

[3] Clark F.J., *Mathematics for data processing.* Reston Publishing Company Inc., Virginia, (1974), pp 298.

[4] Cartwright J.E., Piroo, The dynamics of Runge-Kutta methods, International Journal of Bifarcatian and Chaos (1992), pp 427-449.

[5] Krishnamurthy E.V. and Sen K.V., *Computer based numerical algorithms.* affiliated East-West Press Private Limited, New Delhi, India (1976), pp 330

[6] Krishnamurthy E.V. and Sen K.S., *Numerical algorithms.* Affiliated East-West Press Private Limited, New Delhi, India (1989), pp 445.

[7] McCormick J.M. and Salvadori, *Numerical methods in Fortran.* Prentice-Hall of India Private Limited. New Delhi, India, (1968), pp 315.

[8] Mary Selvam, A., 'Universal spectrum for interannual variability of monsoon rainfall over India', Adv. Atmos. Sci., **10(2)**, (1993a), 221-226.

[9] National Informatics Centre (NIC) NEC system 1000 Hardware and software overview, Govt. of India, New Delhi, pp. 17.

[10] Ralston A. and Wilf, H.S., *Mathematical methods for digital computers*, John wiley and Sons Inc., New York (1960), pp 240.

[11] Rajaraman V., *Computer oriented numerical methods.* Prentice-Hall of India Private Limited. New Delhi, India (1980), pp 160.

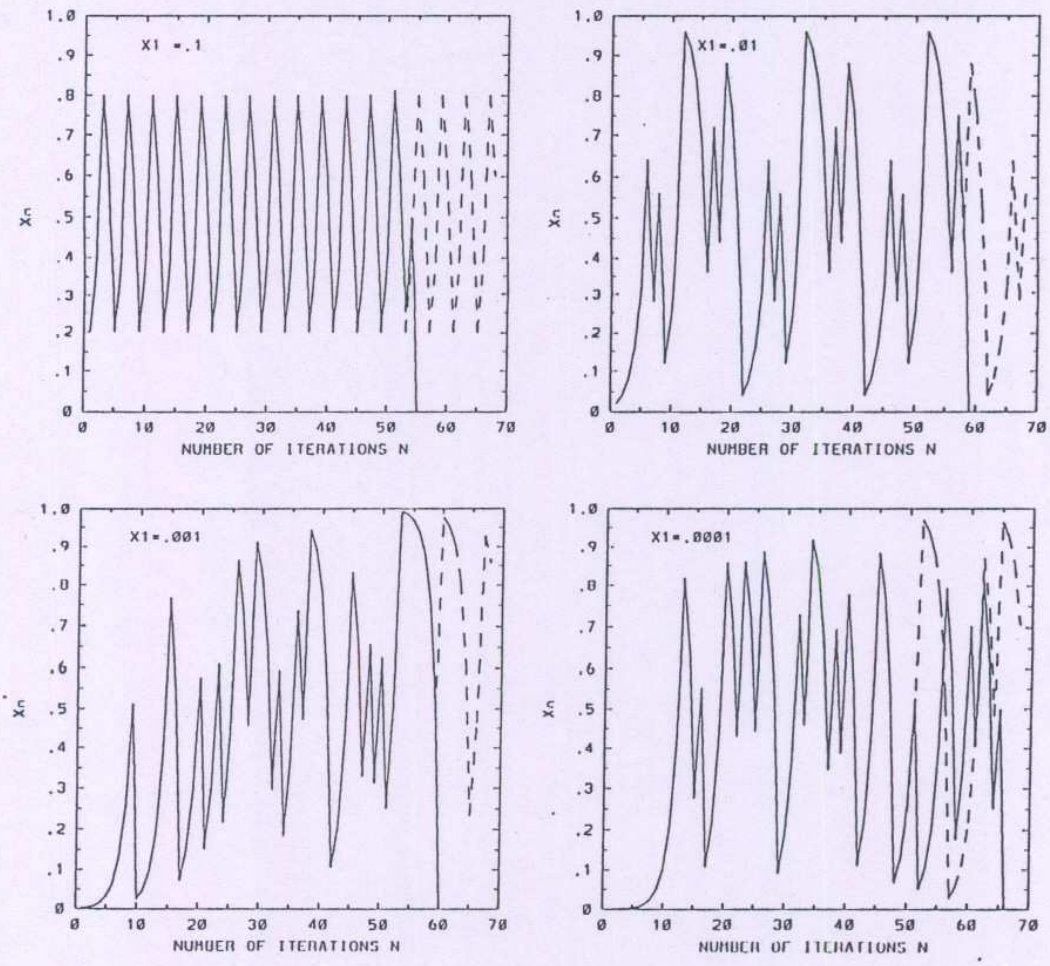
[12] Spiegel M.R., *Finite differences and difference equations.* Schaum's out-line series McGraw-Hill Book Company, New York (1971) pp 235.

[13] Spiegel M.R., Mathematical Hand Book, McGrow-Hill Book Company Inc., New York (1968) pp 120.

BERNOULLI SHIFT

$X_{n+1} = MOD(2*X_n)$

Single precision



Continous Lines–Computed value Dashed Lines–Actual value

Figure 1

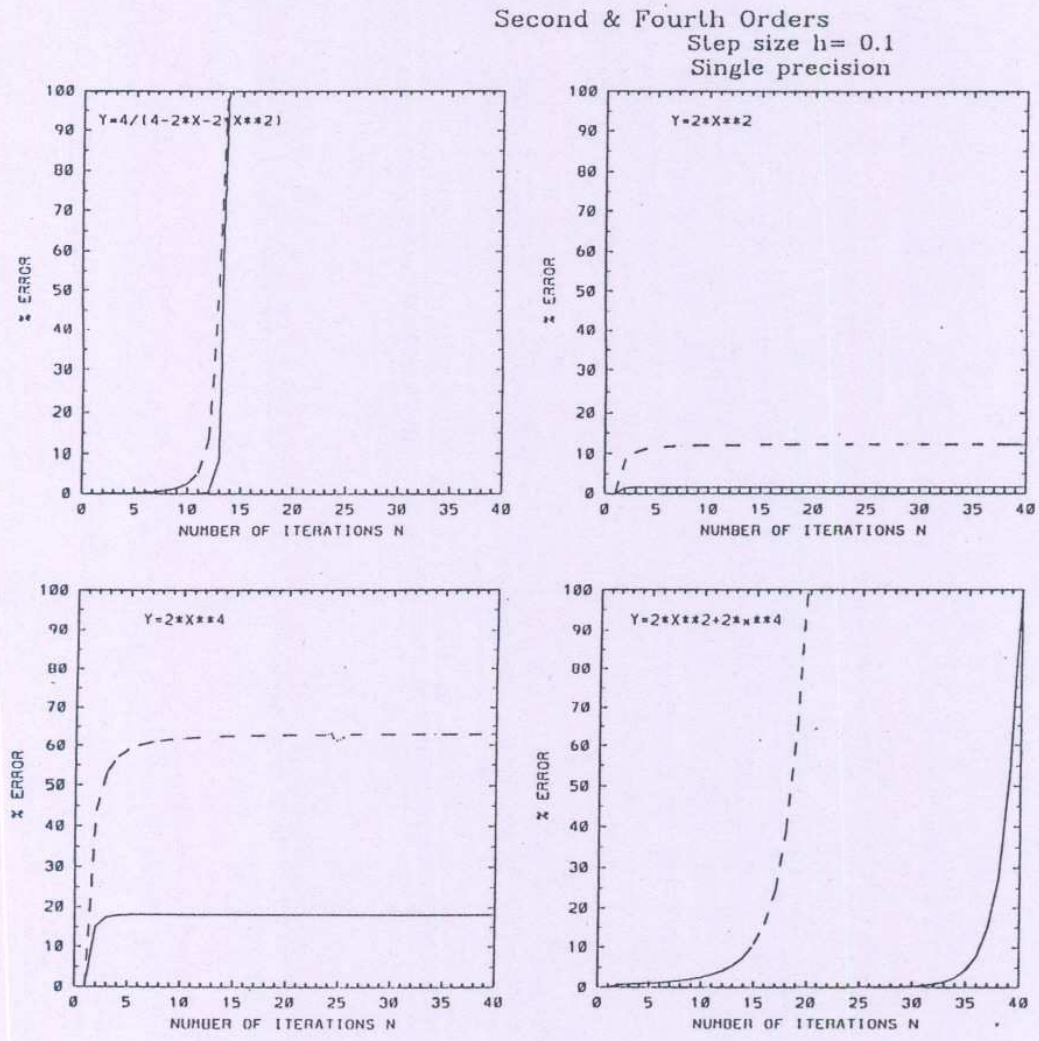
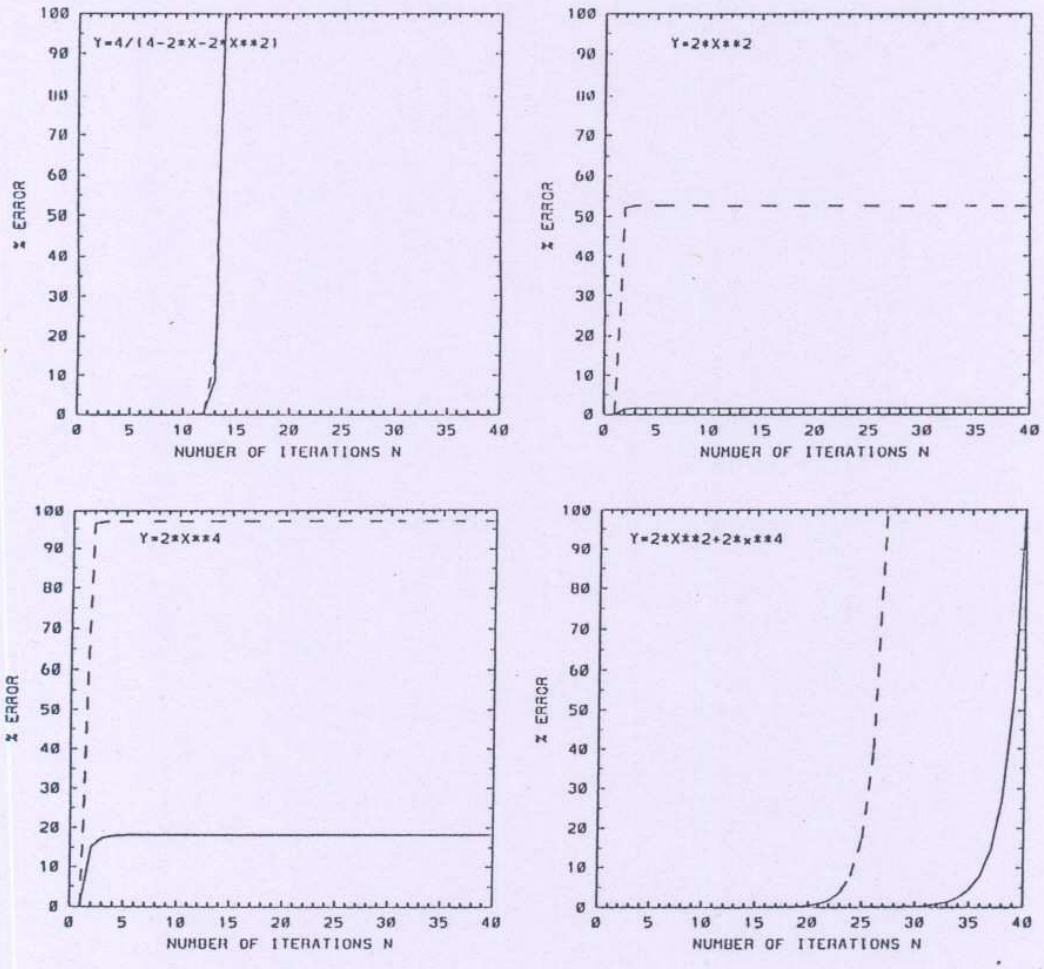

RUNGE KUTTA INTEGRATIONS
Second & Fourth Orders
Step size h= 0.1
Single precision
Y=4/(4-2*X-2*X**2)
Y=2*X**2
Y=2*X**4
Y=2*X**2+2*X**4
% ERROR
NUMBER OF ITERATIONS N
Continous line - 4th order,
Dashed line - 2nd order,

Figure 2

RUNGE KUTTA INTEGRATIONS

Fourth Order
Single precision h=.1



Dashed line X=.01,

Continous line X =.1

Figure 3