Department of Computer Science

Gujarat University

Certificate

Roll No: _____**30**_____ Seat No: _____

This is to certify that Mr./Ms. ***Rathod Ajinkya*** student of MCA Semester – II has duly completed his/her term work for the semester ending in June 2020, in the subject of ***Computer Oriented Numerical Methods*** towards partial fulfilment of his/her Degree of Masters in Computer Applications.

Date of Submission:**01/07/2020** Internal Faculty

Head of Department

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – II

Subject: - COMPUTER ORIENTED NUMERICAL METHODS

Name:- Ajinkya Rathod

Roll No.: - 30 **Exam Seat No.: -_____**

Sr. No.	Contents	Date	Signature
1	Errors	10/02/20	
2	Root finding methods	10/02/20	

Department Of Computer Science
Rollwala Computer Centre
Gujarat University

MCA – II

Subject: - COMPUTER ORIENTED NUMERICAL METHODS

Name:- Ajinkya Rathod

Roll No.: - 30 **Exam Seat No.: -_____**

Sr. No.	Contents	Date	Signature
1	Numerical Integration-I	01/07/20	
2	Open type Integration formulas & Gauss Quadrature Formulas	01/07/20	
3	Solution of Ordinary Numerical Methods	01/07/20	
4	Ordinary Differential Equations - II	01/07/20	
5	Numerical Differentiation	01/07/20	

E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y

M.C.A. – II

R O L L NO : 30
N A M E : Ajinkya Rathod
S U B J E C T : COMPUTER ORIENTED NUMERICAL METHODS

NO.	TITLE	PAGE NO.	DATE	SIGN
	Assignment-1	**96**		
1	**Apply Bisection method to solve the algebraic equation f(x) = x^3 - x - 1 = 0 correct to 6 decimal places.**		**01/07/2020**	
2	**Compute the number of iterations needed to find a root of f(x) = x*e^x = 1 in the interval (0,1) correct upto 6 decimal places. Apply Bisection method to solve the equation.**		**01/07/2020**	
3	**Apply False Position method to solve the algebraic equation f(x) = x *log10 (x) = 1.2 correct to 6 decimal places.**		**01/07/2020**	
4	**Find the root of f(x) = 3x -cosx- 1 using Secant method.**		**01/07/2020**	
5	**Set up a Newton Raphson iteration for computing the square root of a given positive number. Using the same find the square root of 2 exact to six decimal places.**		**01/07/2020**	
6	**Find the root of f(x) = 3x - cosx - 1 using Fixed Point method.**		**01/07/2020**	
	Assignment-2	**128**		
1	**Evaluate Integral of (e^x^2)*sin x dx from 0 to 1 using Trapezoidal rule correct to 3 decimal places**		**10/06/2020**	
2	**Evaluate the integral:integral of dx/(1+x) from 0 to 1 Using(i) Simpson's 1/3 Rule correct to six decimal places (ii) Simpson's 3/8 rule correct to six decimal places**		**10/06/2020**	
3	**A car laps a race track in 84 seconds. The speed of the car at each 6-second interval is determined by using a radar gun and is given from the beginning of the lap, in feet/second by the entries in the following table. Time 0 6 12 18 24 30 36 42 48 54 60 66 72 78 84 Speed 124 134 148 156 147 133 121 109 99 85 78 89 104 116 123 How long is the track? Use (i) Trapezoidal Rule (ii) Simpson's 1/3 rule (iii) Simpson's3/8 rule**		**10/06/2020**	
4	**Write a program to solve the differential equation dy/dx=(y-x)/(y+x), where y(0) = 1, using (i) Euler's method (ii) Runge - Kutta second order method in the interval 0 to 1 using step-size 0.1 Tabulate your results**		**10/06/2020**	
5	**Find the solution of differential equation, for the range 0 <= t <= 1 dy/dt = t + (y)^(1/2) with y(0) = 1, taking step size h =**		**10/06/2020**	

E P A R T M E N T O F C O M P U T E R S C I E N C E
R O L L W A L A C O M P U T E R C E N T R E
G U J A R A T U N I V E R S I T Y

M.C.A. – II

R O L L NO : 30
N A M E : Ajinkya Rathod
S U B J E C T : COMPUTER ORIENTED NUMERICAL METHODS

	0.2 using Runge-Kutta method of order 4			
6	**Find the solution of differential equation dy/dt = 1/2 (t+y), for y (2.0) given y(0) = 2 ,y(0.5) = 2.636,y(1.0) = 3.595 and y(1.5) = 4.968 , use h = 0.5 using (i) Milne-Simpson's predictor corrector method (ii) Adam-Bashforth-Moulton's predictor-corrector method**		**10/06/2020**	
7	**Use Adam-Bashforth-Moulton's predictor-corrector method to obtain the solution of the equation dy/dx= 1-xy/x^2 at x = 1.4, where y(1) = 1. Compute y(1.1), y(1.2) and y(1.3) using Runge-Kutta second order method. Tabulate the results obtained thus.**		**10/06/2020**	
8	**Use Milne Simpson predictor corrector method to obtain the solution of the equation dy/dx= 1-xy/x^2 at x = 1.4, where y(1) = 1. Compute y(1.1), y(1.2) and y(1.3) using Runge-Kutta fourth order method. Tabulate the results obtained thus.**		**10/06/2020**	
9	**From the following table estimate y'(1.1) and y'(1.2) using 3 point formulas and 5 point formulas** **x 1.0 1.05 1.10 1.15 1.20 1.25 1.30** **y 1.0 1.0247 1.0488 1.0724 1.0954 1.1180 1.1402**		**10/06/2020**	

Name : Rathod
Ajinkya

Class : MCA I

Roll : 30

Sub : CONM

Assigment 2:
Numerical Integration

What is need of numerical integration? Explain various situation where such need arises

Ans whenever anti derivative is known, in finite integral is known. The difficulty arises in computing definate integral in many situation by different reasons

Situation :

Evaluation $\int_a^b f(x)\,dx$ is anti-derivative

i) $\int_0^1 e^{x^2}$ & e) $\int_1^2 \frac{dx}{\log x}$

Many such examples are seen in statistical and engineering problems.

$$H(x) = \frac{4ll}{l^2 - x^2} \int_0^{\pi/2} \left(1 - \left(\frac{x}{l}\right)^2 \sin^2\theta\right)^{\frac{1}{2}} d\theta$$

If a, b and l given, the integral occuring in equation is called integral and can't be expressed in terms of standard form

Situation 2:

Rather than evaluate an definite a typical situation would be full expression is in the form.

Situation 3

Expression of function to be integrated is not available

Example, speed of an object may have been measured at different time intervals and distance needs to be computed.

Q2 Give geometrical interpretation with $f(x) \geq 0$ and $a \leq x \leq b$

The definite integral of non-negative integrand of closed interval is area under graph of f

Q] (c) Differentiate:

(a) Newton cotes integration formula and Guass quadrature formula

(b) Close type and open type formulas

(a) x_i are called absissas and w_i are weights

Fitting function are $(x_i, f(x_i))$ and integrating resulting of fitted function.

(b) x_i and w_i's both are assumed as unknown

Integration Formula

x_i fixed w_i unknown Newton cotes integration formula	x_i's unknown w_i's unknown Gauss Quadrature formula

If $a < x_0 < x_1 < x_2 \cdots < x_{n-1} < x_n < b$
then integration is open type integration

Close type

Open type

Q4 Explain trapezodial rule for estimating

Ans $\int_a^b f(x)\,dx = \frac{\Delta x}{2}\left[f(a) + f(b)\right]$

→ This rule is trapezodial rule as it gives area of trapezium formed from joining points by straight line and vertical lines $x=a$ $x=b$

Area $= \frac{1}{2} \times h \times (a+b)$

addition of parallel sides

So if we join points at $f(a)$ and $b(f(b))$ it forms a trapezium

Q5. Explain basic principle in deriving Newton Gate integration formula.

Ans Basic principle in obtaining Newton Gates integration formula is as follows

Between and integrand $f(x)$ is approximated by interpolating polynomial.

$f(x) \Rightarrow$ approximated by interpolating polynomial in (x) of degree

$$\int_a^b f(x)\,dx \cong \int_a^b (P_n(x)\,dx)$$

$$\int_a^b P_n(x)\,dx = \int_a^b \left(\sum_{i=0}^{n} l_i(x)\, y_i \right) dx$$

$$= \sum_{i=0}^{n} \left(\int_a^b l_i(x)\,dx \right) y_i$$

$$= \sum_{i=0}^{n} \left(a_i y_i \right.$$; giving

$$\left(a_i = \int_a^b l_i(x)\,dx \right)$$

Q. Write composite form of trapezoidal rule.

Ans

$$\int_a^b f(x)\,dx \simeq \frac{h}{2} \left[f(x_0) + 2 \sum_{i=1}^{N} f(x_i) + f(x_N) \right]$$

In following figure, intervals are shown reduces truncation error.

y
y = f(x)
Error
x
x0
x1
x2

y
y = f(x)
Error
x
x0
x1
x2
x3

$$= \frac{0.1667}{2}\left[1 + 2\left(\cancel{.792}\ .9726 + .8948 + .7788 + 0.6412 + .4954 + .3679\right)\right.$$

$$= 0.8334\left[\cancel{0.89}\ 8.9415\right]$$

$$= \underline{\underline{0.7452}}$$

Q8 Calculate integral of $\int f(x)\,dx$ when $f(x) = x^2$ using (2) Trapezoidal rule

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + f(b)\right]$$

$$h = \frac{b-a}{n} = \frac{2-0}{1} = 2$$

$$\int_0^2 x^2\,dx = \frac{2}{2}\left[f(0) + f(2)\right]$$

$$= 1[0+4] = 4.$$

(ii) Simpson rule

$n = 2 \quad \frac{2-0}{2} = 1$

$= \frac{1}{3}\left[f(0) + 4\left(f(0+1) + f(0+1)\right)\right]$

$= \frac{1}{3}\left[0 + 4 + 4\right]$

$= \frac{1}{3}(8) = 2.667$

(iii) Simpson 3/8 rule.

$h = \frac{2-0}{3} = \frac{2}{3}$

$\int_0^2 x^2\,dx = \frac{3}{8} \times \frac{2}{3}\left[f(0) + 3 \times \frac{2}{3} + 3f\left(0 + 2 \cdot \frac{2}{3}\right) + f\left(0 + 3 \cdot \frac{2}{3}\right)\right]$

$= \frac{1}{4}\left[0 + \frac{4}{3} + 3 \times \frac{16}{9} + 4\right]$

$= \frac{1}{4}\left[\frac{1}{3} + \frac{4}{3} + 1\right]$

$= 2.667$

Q7 Estimate $\int_0^1 e^{-x^2} dx$ using trapezodial rule.

Ans 2

$N = 2$

$$h = \frac{b-a}{N} = \frac{1-0}{2} = 0.5$$

$x_0 = 0 \Rightarrow 1$

$x_1 = 0.5 \Rightarrow 0.7788$

$x_2 = 1 \Rightarrow 0.3679$

$$\int_0^1 e^{-x^2} dx = \frac{h}{2}\left[1 + 2(0.7788 + \cdots 0.367\right.$$

$$= \underline{\underline{0.7313}}$$

$1/(x+1)$

(1) Trapezodial $n=1$ $n=2$

$$\int_0^2 1/(x+1)\,dx = \frac{2}{2}\left(f(0)+f(2)\right)$$

$$= 1\left[1+0.33\right]$$

$$= 1.33$$

(2) Simpson 1/3 rule: $n=2/3$ $n=3$

$$\frac{3h}{8}\left[f(0)+3f(0+2/3)+3f(0+2\times 2/3)+f(0+3\times 2/3)\right]$$

$$= 1/4\left(1+3\times 3/5+3\times(3/2)+1/3\right)$$

$$= 1.1048$$

$$f(4/3) = \frac{1}{4/3+2} = 3/7$$

$$f(\tfrac{6}{3}) = f(2) = 1/3 = \underline{\underline{0.33}}$$

$\underline{\underline{\sin x}}$

(1) Trapezoidal $n=1$ $h=2$

$$\int_0^2 \sin x \, dx = \frac{h}{2}\left[f(0) + f(2)\right]$$

$$f(0) = \sin 0 = \underline{0}, \quad f(2) = \sin 2 = 0.9093$$

$$= 1\left[0 + 0.9093\right]$$

$$= \underline{\underline{0.9093}}$$

(2) Simpson 1/3 Rule

$$\int_0^2 \sin x \, dx = \frac{h}{3}\left[f(0) + 4(f(0.8)) + f(0.2)\right]$$

$f \quad e^x$

(i) Trapezoidal rule

$n = 1 \qquad h = 2$

$$\int_0^2 e^x \, dx = \frac{h}{2}\left[f(0) + f(2)\right]$$

$f(0) = e^0 = 1$

$f(2) = e^2 = 7.3891$

$$\int_0^2 e^x \, dx = 1\left[1 + 7.3891\right] = 8.3891$$

(ii) Simpson 1/3 rule

$$\int_0^2 e^x \, dx = \frac{h}{3}\left[f(0) + 4f(x_1) + f(x_2)\right]$$

$$= \frac{1}{3}\left[1 + (4 \times 2.7183) + 7.3891\right]$$

$$= \frac{1}{3}\left[19.2623\right]$$

$$= 6.4208$$

$$\int_0^2 \sin x \, dx =$$

$$\frac{1}{3}\left[0 + 4 \times 0.8415 + 0.7093\right]$$

$$= \frac{1}{3}\left[4.2793\right]$$

$$= \underline{\underline{1.425}}$$

(iii) Simpson 3/8 rule

$n = 3 \quad h = 2/3$

$f(2/3) = \sin(2/3) = 0.6184$

$f(4/3) = \sin(4/3) \quad 0.9720$

$$\int_0^2 \sin x \, dx = \frac{1}{4}\left[0 + 3 \times 0.6184 + 3 \times 0.9720 + 0.9091\right]$$

$$= \frac{1}{4}\left[5.6805\right]$$

$$= \underline{\underline{1.420}}$$

NEW INDIA
PAGE NO.: DATE

(ii) Simpson 3/8 rule

$\int^{2} e^{x} dx$

$3h/8 \left[f(0) + 3f\left(0+\frac{2}{3}\right) + 3f\left(0+\frac{2\cdot 2}{3}\right) + f(0+3\cdot\frac{2}{3}) \right]$

$\int_0^2 e^{x} dx = 1/4 \left[1 + 3 \times 1.9477 + 3 \times 3.7932 + 7.3891 \right]$

$= 1/4 (25.6133)$

$= 6.4033$

Q2 Given function f at following value

x	1.8	2.0	2.2	2.4
$f(x)$	3.1204	4.42569	6.05241	8.03014

(a) Trapezoidal Rule.

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

$$= \frac{0.2}{2}\left[3.1204 + 2(4.42569 + 6.05241 + 8.03014) + 10.4617\right]$$

$$= 0.1\,[50.58337]$$

$$= \underline{\underline{5.05834}}$$

(B) Simpson 1/3 rule

$$\int_a^b f(x)\,d(x) = \frac{h}{3}\left[f(x_0) + 4[f(x_1) + f(x_3)] + 2[f(x_2) + f(x_4)]\right]$$

$$\int_{1.8}^{2.6} f(x) = \frac{0.2}{3}\left[3.1204 + 4[4.42569 + 8.03014] + 2(6.05241 + 10.46673)\right]$$

$$= \underline{\underline{5.11274}}$$

Q10 Given form of closed type Newton Cote's integration formula

$$\int_a^b f(x)\,dx = nh \sum_{i=0}^{n} C_i^n f(x_i) \text{ with}$$

$$h = \frac{b-a}{n} \quad x_i = x_0$$

State a properties of it.

Ans C_i^n has 2 properties

① $\sum_{i=0}^{n} C_i^n = 1$

② $C_i^n = C_{n-i}^n$

NEW INDIA
PAGE NO.: DATE

Q11 Derive

Trapezoidal Rule

Ans $\int_a^b f(x)\,dx \simeq nh \sum_{i=0}^{n} C_i^n f(x_i)$

with $h = \frac{b-a}{n}$, $x_i = x_0 + ih$, $i = 0, 1, 2, \ldots n$

a

let $n = 1$

$\int_a^b f(x)\,dx \simeq h \sum_{i=0}^{1} C_i^n f(x_i)$

$= h C_0^1 f(x_0) + h C_1^1 f(x_1)$

By two properties

$C_0^1 = C_1^1$ and $C_0^1 + C_1^1 = 1$

gives

$C_0^1 = C_1^1 = 1/2$

$C_0' = C_1'$ and $C_0' + C_1'$ gives $C_0' = C_1' = 1/2$

Simpson 1/3 Rule

$$\Rightarrow \int_a^b f(x)\,dx \simeq nh \sum_{i=0}^{n} c_i^n f(x_i)$$

with $h = \frac{b-a}{2}$ $x_i = x_0 + ih$

$i = 0, 1, 2, 2$

Let $n = 2$

$$\int_a^b f(x)\,dx \simeq 2h \sum_{i=0}^{2} c_i^2 f(x_i) \quad \cdots (1)$$

Here $c_0^2 = c_2^2$ and $c_0^2 + c_1^2 + c_2^2 = 1$

By evaluating c_i^2

$$c_i^n = \frac{(-1)^{n-i}}{n!(n-i)!} \cdot \frac{t(t-1)\cdots(t-n)}{(t-i)}$$

NEW INDIA
PAGE NO.: DATE

$$C_\alpha^2 = \frac{(-1)^{t-2}}{\alpha-2!\,(2-2)!} \int_0^2 \frac{t(t-1)(t-2)}{(t-2)}\,dt$$

$$= \frac{1}{4} \int_0^2 t(t-1)\,dt$$

$$= \frac{1}{4} \int_0^2 (t^2 - t)\,dt = \frac{1}{4}\left[\frac{t^3}{3} - \frac{t^2}{2}\right]$$

$$= \frac{1}{4}\left[\frac{8}{3} - \frac{4}{2}\right] = \frac{1}{4}\left(\frac{8}{3} - 2\right)$$

$$= \frac{1}{4}\left(\frac{8-6}{3}\right)$$

$$= \frac{2}{4\times 3} = \frac{1}{6}$$

$$C_\alpha^2 = \frac{1}{6} \Rightarrow C_0^2 = C_2^2 = \frac{1}{6}$$

$$\int_a^b f(x)\,dx \cong ah\,c_0^2 f(x_0) + ah\,c_1^2 f(x_1) + ah\,c_2^2 f(x_2)$$

$$\int_a^b f(x)\,dx \cong \frac{h}{3}\left[f(x_0) + 4f(x_1)\right.$$

$$= \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right.$$

it can also be expressed as

$$\int_a^b f(x)\,dx \cong \frac{h}{3}\left[f(a) + 4f(a+h) + f(a+2h)\right]$$

This is Simpson 1/3.

~~This~~ is

Q2

Estimate $\int_0^4 e^x \, dx$ by

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 4f(a+h) + f(a+2h)\right]$$

Here $a = 0$, $h = \frac{4-0}{2} = 2$

$$\int_0^4 e^x \, dx = \frac{2}{3}\left[f(0) + 4f(0+2) + f(0+2.2)\right]$$

$$= \frac{2}{3}\left[1 + 4[7.38906] + 54.59815\right]$$

$$= \frac{2}{3}[85.154]$$

$$= 56.769$$

$x_0 = 0 \Rightarrow f(x_0) = e^0 = 1$

$x_1 = 1 \Rightarrow e^1 = 2.38906$

$x_2 = 2 \Rightarrow 54.59815$

$= \frac{h}{3}[f(x_0) + 4(f(x_1) + f(x_2)]$

$= \frac{2}{3}[1 + 4(2.38906]$

$= \frac{h}{3} f(x_0) + 4(f(x_1) + f(x_2)]$

$= \frac{2}{3}[1 + 4(2.38906) + 54.59815]$

$= 56.76959$

$n = 4$

$$h = \frac{4-0}{4} = 1$$

$x_0 = 0 \quad = 1$
$x_1 = 1 \quad = 2.71828$
$x_2 = 2 \quad = 7.38936$
$x_3 = 3 \quad = 20.08554$
$x_4 = 4 \quad = 54.59815$

$$\int_0^4 e^x \, dx = \frac{h}{3}\left[f(x_0) + 4(f(x_1) + f(x_3)) + 2(f(x_2)) + f(x_4)\right]$$

$$= \frac{1}{3}(161.59135)$$

$$= 53.86785$$

Error in simpons 1/3 rule

error = True − approx.

$= 53.59815 - 56.76959$

$= -3.171143$

Composite simpson rule with N=2

53.59815 − 96.76959

= −3.17155

(c) Composite simpson rule N=4

53.59815 − 53.86785

= −0.2657

Q13 Write pseudocode for following integration formulas to obtain result

~~step 1:= input a, b : f(x) M~~

~~M = max iteration~~

NEW INDIA
PAGE NO.: DATE

~~Step 1: Input a, b ; f(x)~~
~~m = max no. of~~

Step 1: Input a, b f(x), m = max iteration

2: $h = (b-a)/2+1$
sum = 0, sum 1 = f(a+h)
K = 0 N = 1 old sum = 0

3. DO
FOR i = 1 to N-1
x(a+i) = a+h+2+i;
x(a+i+1)
a+h+2+i+1)
sum = sum + f(x)a+i)
sum 2 = sum 1 + f(x 2xi+1))

End do

4. Sum = 2 + sum + 4 * sum 1 + f(a) + f(b)

5. ~~Input: sum = h + sum/1;~~ Print sum

6. if (sum - old sum) < ε; Output: Ans = sum.

6. if |sum - oldsum| < ε

7. Else

k = k+1 x = a*x old sum = sum

8. Else print not give desired accuracy in M iterations

Q14 Car laps a race track in 84 secs.

The speed of car at each 6 second interval is determined using radar gun from beginning of lap in feet/second by intervals

Time	0	6	12	18	24	30	36	42	48
speed	104	134	148	156	147	133	121	107	99

54	60	66	72	78	84
85	78	84	107	116	127

How long is track?

7. $K = KH$, $N = 2 * N$ add sum = sum;
if $K <= M$, goto step 3.

8. Else print does not give desired accuracy in iterations

(b) <u>Trapezoidal</u>

1. INPUT N,

2. sum = 0 K = 0 N = 1

3. DO
   i = 1 to N - 1
   x = a + h * i
   sum += 2 * f(x)
   End DO

4. sum += f(a) + f(b)

5. sum = (h/2) * sum;

6. print sum

$x_0 = 0 \Rightarrow 0$

$x_1 = 0.05 = 0.00377$

$x_2 = 0.50 = 0.05574$

$x_3 = 0.75 = 0.05104$

$x_4 = 1 = 0.69315$

$x_5 = 1.05 = 1.47029$

$x_6 = 1.50 = 2.65197$

$x_7 = 1.75 = 4.29701$

$x_8 = 2 = 6.43775$

$$= \frac{0.05}{2}\left[0 + 0(0.00377 + 0.05577 + 0.05104 + 0.69315 + 1.47029 + 0.65197 + 4.29701) + 6.43775\right]$$

$$= 0.025\left[25.275\right]$$

$$= \underline{\underline{3.1595}}$$

$$\int_0^{84} f(x)\,dx = \frac{h}{3}\Big[f(x_0) + 4(f(x_1) + f(x_3) + f(x_5) + f(x_7) + f(x_9) + f(x_{11}) + f(x_{13})) + 2(x_2 + x_4) + f(x_6) + f(x_8) + f(x_{10}) + f(x_{12}) + f(x_{14}))\Big]$$

$$= \frac{6}{3}\Big[124 + 4(134 + 137 + 104 + 85 + 84 + 16 + 2(148 + 147 + 101 + 99 + 78 + 104 + 101)$$

$$= 2[4929]$$

$$= \underline{9858 \text{ feet}}$$

NEW INDIA
PAGE NO.: DATE

Q5 Approx: $\int_0^2 x^2 \ln|x^2+1|\,dx$

using $h=0.05$ use.

(i) Composite trapezoidal rule

Ans $h=0.05$

$$N=\frac{b-a}{0.25}=\frac{2}{0.25}=8$$

$$\int_0^2 x^2 \ln(x+1)\,dx = \frac{h}{2}\left[f(x_0)+2(f(x_1)+f(x_2)+f(x_3)+f(x_4)+f(x_5)+f(x_6)+f(x_7)+f(x_8)\right]$$

Approximate

$$\int_0^2 x^2 e^{-x^2} \quad \text{using } h=0.5$$

1) Composite Trapezoidal rule

$h = 0.05 \quad N = 8$

$x_0 = 0 \quad \Rightarrow 0$

$x_1 = 0.05 \quad \Rightarrow 0.05871$

$x_2 = 0.50 \quad \Rightarrow 0.1947$

$x_3 = 0.75 \quad \Rightarrow 0.3205$

$x_4 = 1 \quad \Rightarrow 0.36788$

$x_5 = 1.25 \quad \Rightarrow 0.3275$

$x_6 = 1.50 \quad \Rightarrow 0.23715$

$x_7 = 1.75 \quad \Rightarrow 0.14324$

$x_8 = 2 \quad \Rightarrow 0.07326$

$$= \frac{0.05}{2}\Big[0 + 2(0.05871 + 0.1947 + 0.3205 + 0.36788 + 0.3275 + 0.23715 + 0.14324) + 0.07326$$

$$= 0.92158$$

NEW INDIA
PAGE NO.: DATE

(5) Composite mid point Rule

$= 0.05 f(0.125)$

$= \underline{\underline{0.00385}}$

$$\int_{0.25}^{0.50} x^2 e^{-x^2} dx = 0.25 \times f(0.375)$$

0.25×0.12213

$= \underline{0.3054}$

$$\int_{0.50}^{0.75} x^2 e^{-x^2} dx = 0.05 \times f(0.875)$$

$= 0.05 \times 0.35605$

$= \underline{0.08901}$

$$\int_{0.75}^{1} x^2 e^{-x^2} dx = 0.05 \times f(1.125)$$

$= 0.05 \times 0.35694$

$= \underline{\underline{0.01785}}$

$$\int_{1}^{1.25} x^2 e^{-x^2} dx = 0.05 \times f(1.375)$$

$= 0.05 \times 0.28544$

$= \underline{\underline{0.07136}}$

$$\int_{1.25}^{1.50} x^2 e^{-x^2} dx = 0.05 \times f(1.375)$$

$= 0.25 \times 0.2744$

$= 0.07136$

(5) Composite simpson 1/3 rule

$$= \frac{h}{3}\left[f(x_0) + 4\left(f(x_1) + f(x_3) + f(x_5) + f(x_7) \right) + 2\left(f(x_2) + f(x_4) + f(x_6) \right) + f(x_8) \right]$$

$$= \frac{0.25}{3}\left[0 + 4(0.05871) + 0.3205 + 0.3275 + 0.14324 + 0.1947 + 0.36788 + 0.23715 + 0.07326 \right]$$

$$= 0.083\left[5.07252 \right]$$

$$= 0.42102$$

$$\int_{1.50}^{1.75} x^2 e^{-x^2} dx = 0.05 \times f(1.625) = 0.05 \times 0.18932 = \underline{0.09708}$$

$$\int_{1.75}^{2} x^2 e^{-x^2} dx = 0.05 \times f(1.875) = 0.05 \times 0.10452 = \underline{0.02613}$$

$$\int_{0}^{2} x^2 e^{-x^2} dx = 0.00385 + 0.03054 + 0.06608 + 0.08901 + 0.08905 + 0.07136 + 0.09709 + 0.02613$$

$$= \underline{0.4233}$$

NEW INDIA
PAGE NO.: DATE

CONM

Name: Ajin Kya Rathod

Roll no. 30

Solution of Ordinary Numerical Methods.

Assign - 3

Define a differential equation. What is meant by solution of differential equation? Verify.

1. $y = c_1 e^{x} + c_2 e^{x} x$ is a solution of differential. $(y'' - 2y' + y = 0)$

(ii) $y = a\sqrt{c-x}$ is solution of diff. equation $y + 1/y = 0$

Define a ordinary eq.

Ans A differential equation is an equation indept variable, depen variable & its one or more derivative.

eg. $y'' + ay = 0$

Solution of differential equation

e.g. $y'' - 2y' + y = 0$

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

$= C_1 e^x + C_2 \frac{d}{dx}[x e^x]$

$= C_1 e^x + C_2 (x \frac{d}{dx} e^x + e^x \frac{d}{dx} x)$

$= C_1 e^x + C_2 (x e^x + e^x)$

$y'(x) = C_1 e^x + C_2 e^x + C_2 e^x$

Substitution in (2),

$0 = C_1 e^x + 2C_1 e^x + C_2 x e^x + C_1 e^x + C_2 e^x$

$= 0$

Here diff eq. $y' + 1/y = 0$

$$\frac{-1}{\sqrt{c-x}} + \frac{1}{\sqrt{c-x}} \neq 0$$

Q2 How is ordinary differential equation diffrt from partial diffrential equation?

A3 When independent variable is function only one variable then all derivatives involved on equation are ordinary & equation is called ordinary diff. equation.

$$y'' + gy = 0$$

$$y' + 1/y = 0$$

Q3 Differential equations Initial value problem & boundary value problem. Classify following differential equations in initial value problem & boundary value problem.

Ans If coordinates are specified at single point. These conditions are called initial conditions.

→ If conditions are specified at more than one point. These are boundary conditions.

i) $y' = t - y$ $y(0) = 1$

→ Initial value problem

(ii) $y'' + 9y = e^x \sin x$, $y(0) = 1$
$y(1) = 2$

→ Boundary value problem

(iii) $y'' + y = 0$, $y(0) = 0$ $y'(0) = 1$

→ Boundary value problem

(iv) $y' = t^2 + y^2$ $y(0) = 1$

→ Initial value problem

Q4 Determine order & degree of following differential

i) $y'' + 4y = e^x$

→ Order = 2, degree = 1

(ii) $y'' + 4(y')^3 + y^2 = x^3 + x^2$

Order = 2 Degree = 1

(iii) $(y'')^2 + (y')^3 + 3y = 5x$

Order = 2 Degree = 2

(iv) $y' + ay^2 = x^2$

Order = 1 Degree = 1

Q5 What are characteristics of single step numerical methods to find solutions of first order.

→ It is direct

→ It is non-iterative

→ Practically, error can not be estimated at prior step.

first order equation

→ Practically, error can not be estimated.

→ ~~It can also be used in~~

a A a

NEW INDIA
PAGE NO.: DATE

ob

Single step	Multistep
• Direct	• More than one prev. step
• Not iterative	• Iterative
• Based on Taylor series method	• Predictor corr. formulas
• Self starting	• Not self-starting
• Probably accuracy and use restricted	• Practically, accuracy can be attained

Q7 What is Euler Formula to solve IVP?

Ans Euler Formula is Taylor series of order 1. when $N=1$ second and higher powers of h are ignored.

→ The Taylor formula for say

→ The Taylor Formula for say are IVP $\frac{dy}{dx}$, $f(x, y(x))$

$$y_{i+1} = y_i + hy_i + O(h^2)$$

NEW INDIA
PAGE NO.: DATE

Q8 Apply Euler method to approximate the solution of given initial value problem the indicated interval using given stepsize h.

i) $x' = x - t \quad (0 \le t \le 1)$,

$x(0) = 2, \; h = 0.25$

Euler Formula is

$$x_{i+1} \simeq x_i + h f(x_i, t_i)$$

$$x_{i+1} \simeq x_i + h(x_i - t_i)$$

$$x_2 = 1.25(x_1) - 0.25(t_1)$$

$$x_{1.25}(1.5) - 0.25(0.25)$$

$$= 3.125 - 0.0625$$

$$x_2 = 3.0625$$

(ii) $y' = xy^3 - y \quad (0 \le x \le 1)$

$y(0) = 1, \quad h = 0.25$

$\Rightarrow$ Here, $f(x, y) = xy^3 - y$

$0 \le x \le 1, \quad y(0) = 1 \quad h = 0.05$

$x_0 = 0. \quad x_1 = 0.05$

$x_2 = 0.50$

$x_3 = 0.75$

$x_4 = 1$

By Euler Formula

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

$0.75 + 0.25(x_i y_i)$

$y = 0.1$

NEW INDIA
PAGE NO.: DATE

t1

$y_1 = y_0 + 0.2(\cos 2 + 0 - \sin 3 + 0)$

$y_1 = 1 + 0.2(1 - 0) - 1 + 2$

$= 1.2$

t2

$y_2 = y_1 + 0.2(\cos 2 + 1 + \sin 3 - 1)$

$= 1.2 + 0.2(0.2100) + 0.5648$

$y_2 = 1.47714$

t3

$y_2 + 0.2(\cos 2 + t_2 + \sin 3 + 2)$

$1.47714 + 0.2(\cos 0.8 + \sin 1.2)$

$y_3 = 1.82289$

NEW INDIA
PAGE NO.: DATE

y_4

$y_3 + 0.2(\cos 2 + 3 + \sin + 3)$

$= 1.82187 + 0.2(\cos 1.82 + \sin 1.4)$

$= 2.09013$

y_5

$y_5 = y_4 + 0.2(\cos^2 + 4 + \sin 3 + 4)$

$= 2.09013 + 0.2(\cos 1.6 + \sin 2.4)$

$= 2.21939$

Solution → $\boxed{2.21939}$

NEW INDIA
PAGE NO.: DATE

Name: Rathod Ajinkya

Roll: 30

COMM

Assignment 4

Assignment-4

A ball at 1200 k is allowed to cool down in air at an ambient temperature at 300k.

Temperature at given by $\frac{d\theta}{dt} = 0.0067\theta^4$

where θ is in k and t in seconds. Find temp t=420 secs

Assume h=40

Ans Range Kutta formula of order 2

$$\frac{dy}{dx} = f(x, y(x)) = y(x_0) = y_0$$

$$a \le x \le b$$

$$k_1 = hf(x_i + h, y_i + k_0)$$

for $i = 0, 1, 2, n$

NEW INDIA
PAGE NO.: DATE

$t_1 = 400$ sec

$\theta_2 = \theta_1 + \frac{1}{2}(k_0 + l_1)$

$k_0 = 240 \times 2.02067 \times 10^{-12}$

$= -93.2856$

$k_1 = hf(t_1 + h, \theta_1 + k_0)$

$= 240 + f(400, 561.8731)$

$= 240 \times -2.067 \times 10^{-12}$

$= -48.4948$

Q Use Taylor's method of order 2 to approx. inde solution of each of problems

(5) (8)

Ans $a + x_2 = 1$

$$y_2 = y_1 + hy_1' + \frac{h^2}{2!} y_1''$$

$$= 1.05 + 0.5(1.05 - 10.5)^2 + 1 + \frac{0.25}{2}(-2(0.5))$$

$$y_2 = 2.125$$

→ $a + x_3 = 1.50$

$$= 2.125 + 0.5(y_2 - x_2^2 + 1) + \frac{(0.5)^2}{2!}(-2x_2)$$

$$y_3 = 2.9375 \quad \text{--- (3)}$$

$a + xy = 1$

$xy = y_3 + h y_3' + \frac{h^2}{2} y_3''$

$= 2.9375 + 0.5\left(2.9375 - (1.5)^2 + 1\right)$
$+ \frac{0.5^2}{2}\left(-2 + 1.5\right)$

$= 3.40625$

(ii) $\frac{dy}{dt} = \cos 2t + \sin 3t$,

Here $\frac{dy}{dt} = y' = \cos 2t + \sin 3t$

$y_0 = 1$, $t_0 = 0$ $h = 0.5$

$y' = \cos 2t + \sin 3t$

$y'' = -2\sin 2t + 3\cos 3t$

$= 1 + 0.5(\cos 2t_0 + \sin 3t_0)$

$= y_1 = 1.34375$

(iii) $y' = \frac{1+t}{1+y}$

Here,

$y' = \frac{1+t}{1+y}$, $y'' =$

NEW INDIA
PAGE NO.: DATE

Using Taylor method

$x + t_1 = 1.5$

$$y_1 = y_0 + h y_0' + \frac{h^2}{2!} y_0''$$

$$= 2 + 0.5\left(\frac{1+2}{1+y_0}\right) + \frac{(0.5)^2}{2}\left(\frac{1}{1+y_0}\right)$$

$$= 2 + 0.5\left(\frac{1+1}{1+2}\right) + \frac{0.25}{2}\left(\frac{1}{3}\right)$$

$$= 2.325$$

(ab) $y' = 1 + \frac{y}{t}$, $y(1) = 1$

$h = 0.1$

$y' = 1 + \frac{y}{t}$ $\quad y'' = \frac{y'}{t} - \frac{y}{t^2}$

$y'' = \frac{y}{t}$

Order 2 Taylor Method:->

$y_1 = y_0 + h y_0' + \frac{h^2}{2!} y_0''$

$1 + (0.25)\left(\frac{1+y_0}{t_0}\right) + \frac{0.25^2}{2}\left(\frac{-y_0}{t_0^2}\right)$

$y_1 = 1.46875$

$t_2 = 1.50$

$y_2 = y_1 + h y_1' + \frac{h^2}{2!} y_1''$

$1.46875 + 0.25 \times \left(\frac{1+y_1}{t_1}\right) + \frac{0.0625}{2}\left(\frac{-y}{t_1^2}\right)$

$z \quad 1.9831 \quad 25$

Compare previous of Euler method, second Taylor pk of order 2, 3 & 4.

① Euler

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

at $x_0 = 0$, $y_0 = 1$

$$x_1 = 0.5$$

$$y_1 = y_0 + 0.5\left(\frac{x_0}{y_0}\right)$$

$$= 1 + 0.5\left(\frac{0}{1}\right) = 1$$

2nd Taylor

$d + x_1 = 0.5$

$y_2 = 1.125 + 0.5\left(\frac{0.3}{1.25}\right) + 0.125\left(\frac{1}{1.25}\right)$

$= 1.4583$

$\rightarrow x + x_2 = 1.5$

$y_3 = 1.4583 + 0.5\left(\frac{1}{1.4311}\right)$

$+ 0.125\left(\frac{1}{1.9583}\right) = 1.8869$

$x\ y = 2$

$y_4 = 1.8869 + 0.5\left(\frac{1.70}{1.8869}\right) + 0.125$

$\left(\frac{1}{1.8869}\right)$

$= 2.3505$

Second Taylor

$x + x_1 = 0.5$

$y_2 = 1.125 + 2.5\left(\frac{0.3}{1.25}\right) + 0.125\left(\frac{1}{1.25}\right)$

$= 1.4583$

$\Rightarrow x + x_2 = 1.5$

$y_3 = 1.4583 + 0.5\left(\frac{1}{1.4317}\right)$

$+ 0.125\left(\frac{1}{1.9583}\right) = 1.9869$

$x\ y = 2$

$y_4 = 1.9869 + 0.5\left(\frac{1.70}{1.9869}\right) + 0.125$

$\left(\frac{1}{1.9869}\right)$

$= 2.3506$

RK order 4

$k_0 = h f(x_i, y_i)$ $k_1 = h f(x_i + h/2)$

$k_2 = h f(x_i + h/2, y_i + k_1/2)$

$k_3 = h f(x_i + h, y_i + k_2)$

$a + b_2 = 1$

k_0 $0.5\left(\frac{0.5}{1+0.0^2}\right) = 0.2222$

k_1 $0.5 f(0.75, 1.2222) = 0.3054$

k_2 $0.5 f(0.75, 1.2528) = 0.3187$

k_3 $0.5 f(1, 1.3774) = 0.3157$

k_4 $1 + \frac{1}{6}(0.2222 + 0.3054 + 0.3187 + 0.3157)$

$y_2 = 1.3$

$a + z = 1.5$

$k_0 = 0.5\left(\frac{1}{1.3}\right) = 1.2672$

$k_1 = 0.5\left(\frac{1.25}{1.9346}\right) = 0.3231$

$k_2 = 0.5\left(\frac{1.25}{1.9155}\right) = 0.327$

$k_3 = 0.5\left(\frac{1.5}{1.7226}\right) = 0.4347$

$y_7 = 1.3 + \frac{1}{6}\left(1.2672 + 2(0.3231) + 2(0.327) + 0.4347\right)$

$y_7 = 1.8741$

Q4 Explain Taylor method of order n to approximate solution of differential equation

$y'(x) = f(x, y(x))$

Lix Advantages. Why used seldom in real life?

Ans

$y_{i+1} = y_i + hy_i' + \frac{h^2}{2!} y_i'' + \frac{h^3}{3!} y_i'''$

→ Taylor series is seldom used in practical for practical due to requiring derivative of expression and takes

→ It is back bone of single step formulas

Repeat Question with Range Kutta formula

$$y_{i+1} = y_i + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

$$k_0 = hf(x_i, y_i)$$

$$k_1 = hf(x_i, y_i)$$

$$k_1 = hf(x_i + h/2, y_i + k_0/2)$$

$$k_2 = hf(x_i + h/2, y_i + k_1/2)$$

$$k_3 = hf(x_i + h, y_i + k_2)$$

NEW INDIA
PAGE NO.: DATE

$f(0,t) = 0.5$

$k = 0.5 \times f(0, 0.02)$

$= 0.5(0.5 + 0^2 + 1)$

$= 0.75$

$0.5 f(0.25, 0.5 + 0.9021)$

$= 0.5 + f(0.25, 0.0525)$

$= 0.945$

$0.5 + \frac{1}{6}(0.75 + 2(0.9625) + 2(0.945) + 0.1075)$

$y = 1.425$

At $x_2 = 2$

$k_0 = 0.5 f(0.5, 1.5251) = 1.08755$

$k_1 = 0.5 f(0.75, 1.7687) = 1.0032$

$k_2 = 0.5 f(0.75, 2.0067) = 1.2321$

$k_3 = 0.5 f(1.2, 2.6[illegible]) = 1.3280$

$y_2 = 2.6376$

At point x_4

$k_0 = 0.5 f(1.5, 4.0668) = 1.3203$

$k_1 = 0.5 f(1.75, 4.696) = 1.7167$

$k_2 = 0.5 f(1.75, 4.6652) = 1.3014$

$k_3 = 0.5 f(2, 5.3082) = 1.5541$

$x_4 = 4.0068 + \frac{1}{6}(1.3787 + 2(1.31675) + 2(1.30014) + 1.1541)$

$= 5.3016$

NEW INDIA
PAGE NO.: DATE

At point $t_2 = 0.50$

$$k_0 = 0.25 f(0.25, 1.2992) = 0.3377$$

$$k_1 = 0.25 f(0.375, 1.5241) = 0.4085$$

$$k_2 = 0.25 f(0.375, 1.5335) = 0.4085$$

$$y_2 = 1.3992 + \frac{1}{6}(0.3377 + 2 \cdot 0.4085 + 2(0.4085) + 0.3377)$$

$$= 1.7106$$

NEW INDIA
PAGE NO.: DATE

(c) $c\ ry' + \frac{1+t}{1+y}$,

$\alpha = t_0 = 1$ $y_0 = 2$

$t_1 = 1.5$

$k_0 = 0.5 f(1, 2) = \cancel{0.333}\ 0.333$

$k_1 = 0.5 f(1.25, 2.1667) = 0.3550$

$k_2 = 0.5 f(1.5, 2.354) = 0.3727$

$y_1 = \alpha + \frac{1}{6}(0.333 + 2(0.355) + 2.0550 + 0.3727$

$y = 2.3541$

NEW INDIA
PAGE NO.: DATE

(d) $y' = 1/ + y/t \quad y(1) = 1$
$h = 0.5$

At $t_0 = 1$

$k_0 = 0.25 \; f(1, 1) = 0.5$
$k_1 = 0.25 \; f(1.125, 1.05) = 0.5278$
$k_2 = 0.5 \; f(1.5, 1.0739)$

$= 1 + \frac{1}{6}(0.5 + 2 \cdot 0.5278 + 2 \cdot 0.5739 + 0.5262)$

$y_1 = 0.5269$

$t_2 = 1.5$

$k = 0.25\, f(1.25, 1.3281) = 0.5558$

$k_1 = 0.25\, f(1.375, 1.2068) = 0.5785$

$k_2 = 0.25\, f(1.375, 1.3182) = 0.5806$

$k_3 = 0.25\, f(1.5, 2.1071) = 0.6016$

$y_2 = 1.5294 + \frac{1}{6}\left[0.5556 + 2(0.5785) + 2(0.5806) + 0.6016\right]$

$y_2 = 2.1081$

e) $y' = xe^{y+x-1}$, $y(0) = 1$

$0 \le x \le 1$

$h = 0.1$

$y' = y^{y+x-1}$

$y(0) = 1$

$0 \le x \le 1$

$h = 0.1$

NEW INDIA
PAGE NO.: DATE

$x_1 = 0.5$

$k_0 = 0.5 f(0, -1) = 0.5$

$k_1 = 0.5 f(0.25, -1.25) = 0.454$

$k_2 = 0.5 f(0.25, -1.220) = -0.45$

$k_3 = 0.5 f(0.5, 1.433) = -0.4036$

$y_1 = -1 + 1/6(-0.5(2(-0.454) + 2(-0.45) + 0.4036))$

$y_1 = -1.4529$

Q2 Give answer of following in one-2 sentences

a) What is advantage of RK method over Taylor series methods of same order?

Ans RK method and Taylor method of same order are equally efficient but RK methods have advantage of no requirement of doing the expression of derivatives

(b) What is order of local truncation error in RK methods?

Ans In RK of order 2 the order of local truncation error is $O(h^3)$ and order of global truncation error is $O(h^2)$.

e) What is drawback of multistep methods?

Ans Not self starting.

f) Advantage of predictor corrector methods over RK methods?

Ans In RK 4 function evaluation per step required.

Q8 Consider IVP

$y' = y - t^2 + 1$

taking $h = 0.2$

Find Milne Simpson predictor corrector formula to estimate at $y(0.8)$ & $y(0)$

Ans Exact soln $t + 1 = e^t$

$t_0 = 0$	$y_0 = 0.5$
$t_1 = 0.2$	$y_1 = 0.8273$
$t_2 = 0.4$	$y_2 = 1.2141$
$t_3 = 0.6$	$y_3 = 1.6487$
$t_4 = 0.8$	$y_4 = 2.1272$

$$y_4 = y_3 + \frac{h}{3}\left(f(t_3, y_3) + f(t_4, y_4)\right)$$

$$= 0.5 + 0.2667\left(0.1(0.5) - (0.5)^2 + 0.1(0.5)^2\right)$$

$$= 0.5 + 0.2667\left(0 \times 0.5 - 0.25 + 0.1 \times 0.25\right)$$

$$= 0.5 + 0.2667\left(-0.25 + 0.025\right)$$

$$= 0.5 + 0.2667\left(-0.225\right)$$

$$y_4 = 0.1275$$

$y_3 = 1.2141 + 0.6667(f(0.4, 1.1277)$
$+ 4 + (0.6, 1.6487)) + (0.8, 1.2, 15))$

$y_4 = 2.5277$

Corrector Formula

$y_4 = y_3 + \frac{0.2}{3}(f(0.4, y_3) + 4f(0.6, 1.6487)$
$+ f(1.2, 5877)) + 4(0.8, 2.1272) + f(0.6, 1.6487)$

Again,

$y_5 = 1.6487 + 0.0667 + (f(1.2, 5.377)$
$+ 4f(0.8, 2.172) + f(0.6, 1.6487))$

$y_5 = 2.6413$

Compare ~~results~~ results for exact solution

	Milne-Simpson	Exact Sol	Error
$y(0.1)$	0.1277	0.1272	0.0005
$y(0.2)$	0.6413	0.6407	0.00325

	Adams Moulton	Exact Soln	Error
$y(0.1)$	0.1272	0.1272	0.0
$y(0.2)$	0.6407	0.6407	0.0001

NEW INDIA
PAGE NO.: DATE

Name: Ajinkya
&
Rathod

Assign- 5
&
Sub: CONM

Numerical Differentiation

NEW INDIA
PAGE NO.: DATE

Q1. Values for $f(x) = xe^x$ are given

x	1.8	1.9	2.0
$f(x)$	10.889365	10.703199	14.778112

2.1	2.2
17.148957	19.855030

Ans Three point end-point

$$f'(x_0) = \frac{1}{2h}\left[-3f(x_0) + 4f(x_0+h) - f(x_0+2h)\right]$$

$h = 0.1$

So,

$$f'(2.0) = \frac{1}{2\times 0.1}\left[-3f(2.0) + 4f(2.1) - f(2.2)\right]$$

$$= 5\left[-3\times(14.778112) + 4\right) - 19.855030$$

$$f'(2.0) = 22.03216$$

Three point midpoint

$f'(x_0) = 8[-f(1.9) + f(2.1)]$

$8[10.703199 + 17.148754]$

$f'(20) = 22.22877$

→ 3 point end point

$\frac{1}{2} h[f[x_0 - 2h] - 4f[x_0 - h] + 3f(x_0)]$

$f'(2.0) = 22.055675$

Five Point midpoint

$f'(2.0) = \frac{1}{12 \times 0.1}[10.877365 - 8(12.703199)$

$+ 8(17.148737$

$f'(2.0) = 22.168992$

Q2 Graphical Representation 3 point midpoint formula.

Q3 How h is too small is not advantagous in numerical differentiation?

Ans In each of numerical differentiation, h occurs in denominator.
As h is made smaller and smaller to increase accuracy, division by small number causes round-off errors to increase.

As a result, beyond certain value h can't be reduced further.

And round-off errors start dominating accuracy not be improved further.

Q4

t	1.00	1.01	1.02	1.03	1.04
i	3.10	3.12	3.14	3.17	3.24

Ans Here,

$$E(t) = L\frac{di}{dt} + Ri \quad \text{where } L = 0.98$$

$$n = 0.01$$

$$R = 0.142$$

By 3 point end point :→

$$i'(1.00) = \frac{1}{2} \times 0.01\left[-3(3.10) + 4(3.12) - 3.14\right]$$

$$= 50 \times 0.04 = 2$$

$$E(1.00) = 0.98 \times 2 + 0.142 \times (3.10)$$

$$= 2.4002$$

Now at t .01

3-point midpoint

$f'(x_0) = \frac{1}{2}h[-f(x_0-h) + f(x_0+h)]$

$f(1.01) = 2$

$= 0.98 \times 2 + 0.142 \times (3.12)$

$= 2.40304$

By 5-point midpoint

$f'(x_0) = \frac{1}{12} \times h[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h))$

$f'(1.01) = \frac{1}{0.12}[3.10 - 8(3.12) + 8(3.16) - 3.04))$

$f'(1.01) = 2.93$

$= 0.97 \times 2.83 + 0.142 \times 3.14$

$= 3.21928$

Now $t = 1.03$

By three point midpoint:-

$$f'(1.03) = \frac{1}{2} \times 0.01 \left[-3.14 + 3.24 \right]$$

$$= 5$$

So, $E(1.03) = 0.98 \times 5 + 0.142 \times 3.18$

$$= \underline{\underline{5.35156}}$$

Now $t = 1.04$

By three point mid point

$$f'(x_0) = \frac{1}{24} \left(3.14 - 4(3.18) + 3(3.24) \right)$$

$$= 7$$

So, $E(1.04) = 0.98 \times 7 + 0.142 \times 3.24$

$$= \underline{\underline{7.32008}}$$

```
/*
==============================
==============================
 * File:      1-bisection.cpp
 * Copyright: 10-Feb-2020 by Ajinkya
Rathod(ajinzrathod)
 *
 * Content:   Bisection Method
 *
==============================
============================== */

#include <iostream>
#include <math.h>
#include <iomanip>
#include <stdbool.h>
// #include <conio.h>
#include <cassert>

#define EPSILON 0.000005
```

```
using namespace std;

double f(double x)
{
	return (x * x * x) - (x) - 1;
}

bool sign(double value)
{
	if(value < 0) {
		return 0; // value is less than 0
	}
	else if(value > 0) {
		return 1; // value is greater than 0
	}
	else {
		cout << endl << " *** Neither +ve nor -
ve. Returning value 0 *** " << endl;
		return 0;
	}
}
```

```
void displayData(double a, double b, double c)
{
    static int count = 0;
    cout << setw(03) << ++count;
    cout << setw(10) <<   a  << " |" <<
setw(10) <<   b  << " |" ;
    cout << setw(10) << setprecision(6) <<
f(a) << " |" << setw(10) << f(b) << " |" ;
    cout << setw(10) <<   c  << " |";
    cout << setw(10) << setprecision(6) <<
fixed << f(c) << endl;
}

void recursive(double a, double b)
{
    double c  = (a + b) / 2;
    // double c  = ((a * f(b)) - (b * f(a))) / (f(b)
- f(a));
    static int count = 0;

    displayData(a, b, c);
```

```
    // Accurate upto 6 decimal places

    float nearest = fabs(roundf(f(c) * 1000000) /
1000000);
    if (EPSILON > nearest) { return; }

    // Accurate upto 6 decimal places Ends



    if((f(c) * f(b)) < 0) { // if negative
        recursive(c, b);
    }
    else { // if positive
        recursive(a, c);
    }
}

int main()
{
    double a = 0.00, b = 1.00;
    bool sign_a, sign_b;
```

```
    while(1) {

        if(sign(f(a)) == sign(f(b))) {
            a += 1;
            b += 1;
        }
        else { break; }
    }
    cout << setw(04) << "No.";
    cout << setw(10) << "a"    << " |"  <<
setw(10) << "b"     << " |" ;
    cout << setw(18) << "f(a)" << " |"  <<
setw(15) << "f(b)"  << " |";
    cout << setw(20) << "c"    << " |"  <<
setw(20) << "f(c)"  ;

    cout << endl << endl;

    recursive(a, b);

    return 0;
}
```

/*
==========
==========

Output :

No.	a	b	f(a)	f(b)	c	f(c)
1	1	2	-1	5	1.5	0.875000
2	1.000000	1.500000	-1.000000	0.875000	1.250000	-0.296875
3	1.250000	1.500000	-0.296875	0.875000	1.375000	0.224609
4	1.250000	1.375000	-0.296875	0.224609	1.312500	-0.051514
5	1.312500	1.375000	-0.051514	0.224609	1.343750	0.082611
6	1.312500	1.343750	-0.051514	0.082611	1.328125	0.014576
7	1.312500	1.328125	-0.051514	0.014576	1.320312	-0.018711
8	1.320312	1.328125	-0.018711	0.014576	1.324219	-0.002128
9	1.324219	1.328125	-0.002128	0.014576	1.326172	0.006209
10	1.324219	1.326172	-0.002128	0.006209	1.325195	0.002037
11	1.324219	1.325195	-0.002128	0.002037	1.324707	-0.000047
12	1.324707	1.325195	-0.000047	0.002037	1.324951	0.000995
13	1.324707	1.324951	-0.000047	0.000995	1.324829	0.000474
14	1.324707	1.324829	-0.000047	0.000474	1.324768	0.000214
15	1.324707	1.324768	-0.000047	0.000214	1.324738	0.000084
16	1.324707	1.324738	-0.000047	0.000084	1.324722	0.000018
17	1.324707	1.324722	-0.000047	0.000018	1.324715	-0.000014
18	1.324715	1.324722	-0.000014	0.000018	1.324718	0.000002

*
==========
========== */

```
/*
==============================
==============================
 * File:      2-bisection.cpp
 * Copyright: 10-Feb-2020 by Ajinkya
Rathod(ajinzrathod)
 *
 * Content:   Bisection Method
 *
==============================
============================== */

#include <iostream>
#include <math.h>
#include <iomanip>
#include <stdbool.h>
// #include <conio.h>
#include <cassert>
```

```
#define EPSILON 0.000005

using namespace std;

double f(double x)
{
	return x * exp(x) - 1;
}

bool sign(double value)
{
	if(value < 0) {
		return 0; // value is less than 0
	}
	else if(value > 0) {
		return 1; // value is greater than 0
	}
	else {
		cout << endl << " *** Neither +ve nor -
ve. Returning value 0 *** " << endl;
		return 0;
	}
```

```
}

void displayData(double a, double b, double c)
{
    static int count = 0;
    cout << setw(03) << ++count;
    cout << setw(10) <<   a  << " |" <<
setw(10) <<   b  << " |" ;
    cout << setw(10) << setprecision(6) <<
f(a) << " |" << setw(10) << f(b) << " |" ;
    cout << setw(10) <<   c  << " |";
    cout << setw(10) << setprecision(6) <<
fixed << f(c) << endl;
}

void recursive(double a, double b)
{
    double c  = (a + b) / 2;
    // double c  = ((a * f(b)) - (b * f(a))) / (f(b)
- f(a));
    static int count = 0;

    displayData(a, b, c);
```

```
    // Accurate upto 6 decimal places

    float nearest = fabs(roundf(f(c) * 1000000) /
1000000);
    if (EPSILON > nearest) { return; }

    // Accurate upto 6 decimal places Ends



    if((f(c) * f(b)) < 0) { // if negative
        recursive(c, b);
    }
    else { // if positive
        recursive(a, c);
    }
}

int main()
{
    double a = 0.00, b = 1.00;
```

```
    bool sign_a, sign_b;

    while(1) {

        if(sign(f(a)) == sign(f(b))) {
            a += 1;
            b += 1;
        }
        else { break; }
    }
    cout << setw(04) << "No.";
    cout << setw(10) << "a"    << " |"  <<
setw(10) << "b"     << " |" ;
    cout << setw(18) << "f(a)" << " |"  <<
setw(15) << "f(b)"  << " |";
    cout << setw(20) << "c"    << " |"  <<
setw(20) << "f(c)"  ;

    cout << endl << endl;

    recursive(a, b);

    return 0;
```

```
}

/*
==========================================
==========================================
```

Output :

No.	a	b	f(a)	f(b)	c	f(c)
1	0	1	-1	1.71828	0.5	-0.175639
2	0.500000	1.000000	-0.175639	1.718282	0.750000	0.587750
3	0.500000	0.750000	-0.175639	0.587750	0.625000	0.167654
4	0.500000	0.625000	-0.175639	0.167654	0.562500	-0.012782
5	0.562500	0.625000	-0.012782	0.167654	0.593750	0.075142
6	0.562500	0.593750	-0.012782	0.075142	0.578125	0.030619
7	0.562500	0.578125	-0.012782	0.030619	0.570312	0.008780
8	0.562500	0.570312	-0.012782	0.008780	0.566406	-0.002035
9	0.566406	0.570312	-0.002035	0.008780	0.568359	0.003364
10	0.566406	0.568359	-0.002035	0.003364	0.567383	0.000662
11	0.566406	0.567383	-0.002035	0.000662	0.566895	-0.000687
12	0.566895	0.567383	-0.000687	0.000662	0.567139	-0.000013
13	0.567139	0.567383	-0.000013	0.000662	0.567261	0.000325
14	0.567139	0.567261	-0.000013	0.000325	0.567200	0.000156
15	0.567139	0.567200	-0.000013	0.000156	0.567169	0.000072
16	0.567139	0.567169	-0.000013	0.000072	0.567154	0.000029
17	0.567139	0.567154	-0.000013	0.000029	0.567146	0.000008
18	0.567139	0.567146	-0.000013	0.000008	0.567142	-0.000002

```
 *
==============================
============================== */




 /*
==============================
==============================
 * File:      3-false-position.cpp
 * Copyright: 10-Feb-2020 by Ajinkya
Rathod(ajinzrathod)
 *
 * Content:   False Position Method
 *
==============================
============================== */

#include <iostream>
#include <math.h>
#include <iomanip>
#include <stdbool.h>
#include <conio.h>
```

```
#define EPSILON 0.000000005

using namespace std;

double f(double x)
{
	return (x * (log10(x)) - 1.2);
}

bool sign(double value)
{
	if(value < 0) {
		return 0; // value is less than 0
	}
	else if(value > 0) {
		return 1; // value is greater than 0
	}

	else {
		cout << "Neither +ve nor -ve. Returning
value 0";
```

```
        getch();
        exit(1);
    }
}

void displayData(double a, double b, double c)
{
    cout << setw(10) <<   a  << " |" <<
setw(10) <<   b  << " |" ;
    cout << setw(10) << f(a) << " |" <<
setw(15) << f(b) << " |" ;
    cout << setw(10) <<   c  << " |" <<
setw(15) << f(c) << endl ;
}

void recursive(double a, double b)
{
    double c  = ((a * f(b)) - (b * f(a))) / (f(b) -
f(a));
    static int count = 0;

    displayData(a, b, c);
```

```
    // if(f(c) > 0.000005) { return; }
    if(count++ > 15) { return; }

    if(sign(f(c))) { // if sign_c is positive
        recursive(c, b);
    }
    else { // if sign_c is negative
        recursive(a, c);
    }

}

int main()
{
    double a = 2.00, b = 3.00;
    bool sign_a, sign_b;

    while(1) {

        if(sign(f(a)) == sign(f(b))) {
            a += 1;
            b += 1;
```

```
        }
        else { break; }
    }

    cout << setw(10) << "a"    << " |"  <<
setw(10) << "b"     << " |" ;
    cout << setw(10) << "f(a)" << " |"  <<
setw(10) << "f(b)"  << " |";
    cout << setw(10) << "c"    << " |"  <<
setw(10) << "f(c)"  ;

    cout << endl << endl;

    recursive(a, b);

    return 0;
}
```

```
/*
==============================
==============================
```

Output :

```
    a |        b |     f(a) |          f(b) |        c |          f(c)

    2 |        3 |  -0.59794 |      0.231364 |   2.72101 |     -0.0170911
    2 |   2.72101 |  -0.59794 |     -0.0170911 |   2.74223 |     0.00138143
 2.74223 |   2.72101 |0.00138143 |     -0.0170911 |   2.74064 |  -2.47288e-006
 2.74223 |   2.74064 |0.00138143 |  -2.47288e-006 |   2.74065 |  -3.55667e-010
 2.74223 |   2.74065 |0.00138143 |  -3.55667e-010 |   2.74065 |  -5.12287e-014
 2.74223 |   2.74065 |0.00138143 |  -5.12287e-014 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 2.74223 |   2.74065 |0.00138143 |  -1.55041e-016 |   2.74065 |  -1.55041e-016
 *
==============================================================================
==============================================================================
 */
```

```
 /*
==============================================================================
==============================================================================
```

```
 * Content:   Secant Method
 *
==============================
============================== */



#include<iostream>
#include<math.h>
#include <iomanip>

#define EPSILON 0.000000005

using namespace std;

double f(double x)
{
    return (3 * x) - cos(x) - 1;
}

bool sign(double value)
{
    if(f(value) < 0) {
```

```
        return 0; // value is less than 0
    }
    else if(f(value) > 0) {
        return 1; // value is greater than 0
    }
}

void displayData(double a, double b, double c)
{
    cout << setw(10) <<   a  << " |" <<
setw(10) <<   b  << " |" ;
    cout << setw(18) << f(a) << " |" <<
setw(15) << f(b) << " |" ;
    cout << setw(20) <<   c  << " |" <<
setw(20) << f(c) << endl ;
}

void recursive(double a, double b)
{
    double c;
    bool sign_c;
    static int count = 1;
```

```
    a = b;
    b = c;

    displayData(a, b, c);

    // if(f(c) > 0.000005) { return; }
    if(count++ > 15) { return; }

// these below lines might not come, coz secant
method does not care if sign is +ve or -ve
    // sign_c = sign(c);

    // if(sign_c) {
    //   recursive(c, b);
    // }
    // else {
    //   recursive(a, c);
    // }
}

int main()
{
```

```
    double a = 2.00;
    double b = 3.00;

    bool sign_a = true;
    bool sign_b = true;

    sign_a = sign(a);

    while(1) {
        sign_b = sign(b);

        if(sign_a == sign_b) {
            a += 1;
            b += 1;
        }
        else {
            break;
        }
    }

    cout << setw(10) << "a"    << " |"  <<
setw(10) << "b"     << " |" ;
```

```
    cout << setw(18) << "f(a)" << " |"  <<
setw(15) << "f(b)"  << " |";
    cout << setw(20) << "c"    << " |"  <<
setw(20) << "f(c)"  ;


    cout << endl << endl;


    recursive(a, b);


    return 0;
}


 /*
==================================
==================================
 Output :


      a            b        f(a)        f(b)           c
f(c)


2.000000 |    3.000000 |  -0.597940 |
0.231364 |   2.721014 |   -0.017091 |
2.721014 |    3.000000 |  -0.017091 |
0.231364 |   2.740206 |   -0.000384 |
```

```
2.740206 |    3.000000 |  -0.000384 |
0.231364 |   2.740636 |   -0.000009 |
2.740636 |    3.000000 |  -0.000009 |
0.231364 |   2.740646 |   -0.000000 |

 *
==============================
============================== */


 /*
==============================
==============================
 * Content:   Newton Rahpson Method
 *
==============================
============================== */


#include<stdio.h>
#include<math.h>
#define EPSILON 0.0000005
```

```
double function5(double x,double y)
{
    return (x * x) - y;
}


double dfunction5(double x)
{
    return 2 * x;
}


void newton_rafson(double x)
{
    double c, y;
    int i = 1, limit, flag = 0;

    printf("Enter number to find its square root:
");
    scanf("%lf",&y);
```

```
    printf("Enter x: ");
    scanf("%lf",&x);

    printf("Enter Max iter: ");
    scanf("%d",&limit);

    c = x - (function5(x, y) / dfunction5(x));

    printf("\n\nno\txn\t\tf(xn)\t\tf'(xn)\t\txn+1\t
\tf(xn+1)");

    printf("\n\n%d |\t%lf |\t%lf |\t%lf |\t%lf
|\t%lf",i,x,function5(x,y),dfunction5(x),c,function
5(c,y));

    while(fabs(function5(c,y)) >= EPSILON &&
i<limit) {
        i++;
        x = c;
        if(fabs(dfunction5(x)) < 0.00001) {
            printf("\nDivide by zero Error
encountered!!!");
```

```
            flag = 1;
            break;
        }

        c=x-(function5(x,y)/dfunction5(x));

        printf("\n%d |\t%lf |\t%lf |\t%lf |\t%lf
|\t%lf",i,x,function5(x,y),dfunction5(x),c,function
5(c,y));
    }
    if(flag != 1)
        printf("\n\nRoot =%lf",c);
}



int main()
{
    double x=0;

    newton_rafson(x);

    return 0;
```

```
}
```

/*
==========
==========

Output :

Enter number to find its square root: 2

Enter x: 5

Enter Max iter: 7

no	xn	f(xn)	f'(xn)	xn+1	f(xn+1)
1	5.000000	23.000000	10.000000	2.700000	5.290000
2	2.700000	5.290000	5.400000	1.720370	0.959674
3	1.720370	0.959674	3.440741	1.441455	0.077794
4	1.441455	0.077794	2.882911	1.414471	0.000728
5	1.414471	0.000728	2.828942	1.414214	0.000000

Root =1.414214

```
 *
==============================
============================== */
```

```
 /*
==============================
==============================
 * Content:   Fixed Point Method
 *
==============================
============================== */
```

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define EPSILON 0.0000005
```

```
double g(double x)
{
    return (3  * x) - cos(x)- 1;
}

void fixed_point(double x)
{

    int i = 1, limit;
    double c;

    printf("Enter value of x: ");
    scanf("%lf",&x);

    printf("Enter Maximum iterations: ");
    scanf("%d",&limit);

    printf("no \t \tx \t \t\t g(x)");

    while(fabs(x-g(x)) > EPSILON && i <= limit)
{
```

```
        printf("\n%d |\t\t%lf |\t\t%lf",i,x,g(x));
        x = g(x);
        i += 1;
    }
}


int main()
{
    double x = 0;

    fixed_point(x);

    return 0;
}


 /*
==============================
==============================
 Output :
```

```
Enter value of x: 0
Enter Maximum iterations: 15
no          x                   g(x)
1 |         0.000000 |          0.666667
2 |         0.666667 |          0.595296
3 |         0.595296 |          0.609328
4 |         0.609328 |          0.606678
5 |         0.606678 |          0.607182
6 |         0.607182 |          0.607086
7 |         0.607086 |          0.607105
8 |         0.607105 |          0.607101
9 |         0.607101 |          0.607102


 *
==============================
============================== */
```

==========
==========
==========
**

NAME : Ajinkya Rathod

ROLL NO : 30

CLASS : MCA-II

SUBJECT : COMPUTER ORIENTED NUMERICAL METHODS(CONM)

==========
==========
==========
**

==========
==========
==========
**

Q(1): Evaluate Integral of (e^x^2)*sin x dx from 0 to 1

using Trapezoidal rule correct to 3 decimal places

```
==============================
==============================
==============================
**
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define epsilon 0.0005

void trapezoidal(double,double,int);

double f(double x)
{
    return (exp(x*x)*sin(x));
}

void main()
{
    int N=2;
    double a,b;
    a=0;
    b=1;
    trapezoidal(a,b,N);
```

```
    getch();
}

void trapezoidal(double a,double b,int N)
{
    int i,limit=20,k=1;
    double sum=0,old_sum=0,h;
    printf("==========================
==============Trapezoidal
Rule==============================
========\n\n");
    printf("\nSr
No\t\t|\tN\t\t|\th\t\t\t|\tIntegral\n");
    printf("_____________________________________
_____________________________________________
_________________________");
    while(k<=limit)
    {
        sum=0;
        h=(b-a)/N;
        for(i=1;i<N;i++)
        {
            sum+=2*f(a+i*h);
```

```
        }
        sum+=(f(a)+f(b));
        sum *=h/2;

    printf("\n%d\t\t|\t%d\t\t|\t%lf\t\t|\t%lf",k,N
,h,sum);
        if(fabs(sum-old_sum)<epsilon)
        {
            printf("\n\n-->The Estimate of the
Integral is %lf",sum);
            break;
        }
        N*=2;
        k++;
        old_sum=sum;
    }
}
```

==========
==========
==========
**

==========
==========

==========
**

output:

==========
=======Trapezoidal Rule==========
========

Sr No	N	h	Integral
1	2	0.500000	0.879636
2	4	0.250000	0.804736
3	8	0.125000	0.785295
4	16	0.062500	0.780386
5	32	0.031250	0.779156
6	64	0.015625	0.778848

-->The Estimate of the Integral is 0.778848

==========
==========
==========
**

==========
==========
==========
**

Q(2): Evaluate the integral:

integral of dx/(1+x) from 0 to 1

Using

(i) Simpson’s 1/3 Rule correct to six decimal places

(ii) Simpson’s 3/8 rule correct to six decimal places

==========
==========

```
==============================
**
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define epsilon 0.0000005

void simpsons1_3(double,double,int);
void simpsons3_8(double,double,int);
double f(double x)
{
    return (1/(1+x));
}

void main()
{
    int N=2;
    double a,b;
    a=0;
    b=2;
    simpsons1_3(a,b,N);
    simpsons3_8(a,b,N);
```

```
    getch();
}



void simpsons1_3(double a,double b,int N)
{
    printf("==========================
===============Simpsons 1/3
Rule==============================
========\n\n");
    int i,limit=20,k=1;
    double sum=0,old_sum=0,h;
    printf("\nSr
No\t\t|\tN\t\t|\th\t\t\t|\tIntegral\n");
    printf("_____________________________________
_____________________________________________
_________________________");
    while(k<=limit)
    {
        sum=0;
        h=(b-a)/N;
        for(i=1;i<N;i++)
        {
```

```
            if(i%2==0)
                sum+=2*f(a+i*h);
            else
                sum+=4*f(a+i*h);
        }
        sum+=(f(a)+f(b));
        sum *=h/3;

    printf("\n%d\t\t|\t%d\t\t|\t%lf\t\t|\t%lf",k,N
,h,sum);
        if(fabs(sum-old_sum)<epsilon)
        {
            printf("\n\n-->The Estimate of the
Integral Using simpsons1/3 Rule is %lf",sum);
            break;
        }
        N*=2;
        k++;
        old_sum=sum;
    }
    printf("\n\n");
}
```

```
void simpsons3_8(double a,double b,int N)
{
    printf("==========================
===============Simpsons 3/8
Rule==========================
=========\n\n");
    int i,limit=20,k=1;
    double sum=0,old_sum=0,h;
    printf("\nSr
No\t\t|\tN\t\t|\th\t\t\t|\tIntegral\n");
    printf("_____________________________
_____________________________________________
_________________________");
    while(k<=limit)
    {
        sum=0;
        h=(b-a)/N;
        for(i=1;i<N;i++)
        {
            if(i%3==0)
                sum+=2*f(a+i*h);
            else
                sum+=3*f(a+i*h);
```

```
        }
        sum+=(f(a)+f(b));
        sum *=3*h/8;

    printf("\n%d\t\t|\t%d\t\t|\t%lf\t\t|\t%lf",k,N
,h,sum);
        if(fabs(sum-old_sum)<epsilon)
        {
            printf("\n\n-->The Estimate of the
Integral Using simpsons3/8 Rule is %lf",sum);
            break;
        }
        N*=2;
        k++;
        old_sum=sum;
    }
}
```

==========
==========
==========
**

==========
==========

==========
**

output:

==========
========Simpsons 1/3 Rule==========
========

Sr No	N	h	Integral
1	2	1.000000	1.111111
2	4	0.500000	1.100000
3	8	0.250000	1.098725
4	16	0.125000	1.098620
5	32	0.062500	1.098613
6	64	0.031250	1.098612

-->The Estimate of the Integral Using simpsons1/3 Rule is 1.098612

==========
========Simpsons 3/8 Rule==========
========

Sr No	N	h	Integral
1	2	1.000000	1.062500
2	4	0.500000	1.056250
3	8	0.250000	1.087541
4	16	0.125000	1.087999
5	32	0.062500	1.095955
6	64	0.031250	1.095995
7	128	0.015625	1.097958
8	256	0.007813	1.097960
9	512	0.003906	1.098449
10	1024	0.001953	1.098449

-->The Estimate of the Integral Using simpsons3/8 Rule is 1.098449

==========
==========
==========
**

==========
==========
==========
**

Q(3): A car laps a race track in 84 seconds. The speed of the car at each 6-second interval is

determined by using a radar gun and is given from the beginning of the lap, in

feet/second by the entries in the following table.

Time 0 6 12 18 24 30 36 42 48 54 60 66 72 78 84

Speed 124 134 148 156 147 133 130 109 99 85 78 89 104 116 123

How long is the track?

Use (i) Trapezoidal Rule (ii) Simpson’s 1/3 rule (iii) Simpson’s3/8 rule

==========
==========

```
==============================
**
#include<stdio.h>
#include<conio.h>
#include<math.h>

void simpsons1_3(double,double,int);
void simpsons3_8(double,double,int);
void trapezoidal(double,double,int);

double f(int x)
{
    switch(x)
    {
        case 0:return 124;
        case 6:return 134;
        case 12:return 148;
        case 18:return 156;
        case 24:return 147;
        case 30:return 133;
        case 36:return 130;
        case 42:return 109;
```

```
            case 48:return 99;
            case 54:return 85;
            case 60:return 78;
            case 66:return 89;
            case 72:return 104;
            case 78:return 116;
            case 84:return 123;
        }
}
void main()
{
        int N=14;
        double a,b;
        a=0;
        b=84;
        trapezoidal(a,b,N);
        simpsons1_3(a,b,N);
        simpsons3_8(a,b,N);
        getch();
}

void trapezoidal(double a,double b,int N)
```

```
{
    int i;
    double sum=0,h;
    sum=0;
    h=(b-a)/N;
    for(i=1;i<N;i++)
    {
        sum+=2*f(a+i*h);
    }
    sum+=(f(a)+f(b));
    sum *=h/2;
    printf("\n-->Length of track using
Trapezoidal Rule=%0.2lf Feet",sum);
}

void simpsons1_3(double a,double b,int N)
{
    int i;
    double sum=0,h;
        sum=0;
        h=(b-a)/N;
        for(i=1;i<N;i++)
```

```
        {
            if(i%2==0)
                sum+=2*f(a+i*h);
            else
                sum+=4*f(a+i*h);
        }
        sum+=(f(a)+f(b));
        sum *=h/3;
        printf("\n-->Length of track using
Simpsons 1/3 Rule=%0.2lf Feet",sum);
}

void simpsons3_8(double a,double b,int N)
{
    int i;
    double sum=0,h;
    sum=0;
    h=(b-a)/N;
    for(i=1;i<N;i++)
    {
        if(i%3==0)
            sum+=2*f(a+i*h);
```

```
        else
            sum+=3*f(a+i*h);
    }
    sum+=(f(a)+f(b));
    sum *=3*h/8;
    printf("\n-->Length of track using Simpsons
3/8 Rule=%0.2lf Feet",sum);
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

-->Length of track using Trapezoidal Rule=9855.00 Feet

-->Length of track using Simpsons 1/3 Rule=9858.00 Feet

-->Length of track using Simpsons 3/8 Rule=9760.50 Feet

==========
==========
==========
**

==========
==========
==========
**

Q(4): Write a program to solve the differential equation dy/dx=(y-x)/(y+x), where y(0) = 1, using

(i) Euler's method

(ii) Runge - Kutta second order method

in the interval 0 to 1 using step-size 0.1
Tabulate your results

==========
==========
==========
**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void euler(double,double,double,int);
void runge_kutta_2(double,double,double,int);

double f(double x,double y)
{
    return ((y-x)/(y+x));
}
void main()
```

```
{
	int limit;
	double xi,yi,h;
	xi=0;
	yi=1;
	h=0.1;
	limit=1;
	euler(xi,yi,h,limit);
	runge_kutta_2(xi,yi,h,limit);
	getch();
}

void euler(double xi,double yi,double h,int limit)
{
	double yi_1;
	yi_1=yi;
	printf("==============EULER
METHOD==============\n\n");
	printf("\nx\t\t|\tSolution\n");
	printf("_____________________________________
_____\n");
	while(xi<=limit)
```

```
    {
        yi=yi_1;
        printf("\n%0.2lf\t\t|\t%lf",xi,yi);
        yi_1=yi + h* f(xi,yi);
        xi+=h;
    }
    printf("\n\n-->Solution With Eulers method=
%lf\n\n",yi);
}


void runge_kutta_2(double xi,double yi,double
h,int limit)
{
    double yi_1,k0,k1;
    yi_1=yi;
    printf("========RUNGE-KUTTA SECOND
ORDER METHOD========\n\n");
    printf("\nx\t\t|\tSolution\n");
    printf("_____________________________________
_____");
    while(xi<=limit)
    {
        yi=yi_1;
```

```
        printf("\n%0.2lf\t\t|\t%lf",xi,yi);
        k0=h*f(xi,yi);
        k1=h*f(xi+h,yi+k0);
        yi_1=yi + (0.5)*(k0+k1);
        xi+=h;
    }
    printf("\n\n-->Solution With RUNGE-KUTTA
SECOND ORDER METHOD= %lf",yi);
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

==========EULER
METHOD==========

x | Solution

0.00	1.000000
0.10	1.100000
0.20	1.183333
0.30	1.254418
0.40	1.315818
0.50	1.369193
0.60	1.415694
0.70	1.456161
0.80	1.491231
0.90	1.530399
1.00	1.547062

-->Solution With Eulers method= 1.547062

=========RUNGE-KUTTA SECOND ORDER METHOD========

x	Solution

0.00		1.000000
0.10		1.091667
0.20		1.168728
0.30		1.234629
0.40		1.291489
0.50		1.340729
0.60		1.383361
0.70		1.420135
0.80		1.451627
0.90		1.478291
1.00		1.500491

-->Solution With RUNGE-KUTTA SECOND ORDER METHOD= 1.500491

==========
==========
==========
**

==========
==========
==========
**

Q(5): Find the solution of differential equation, for the range 0 <= t <= 1 dy/dt = t + (y)^(1/2)

with y(0) = 1, taking step size h = 0.2 using Runge-Kutta method of order 4

==========
==========
==========
**

```
#include<stdio.h>

#include<conio.h>

#include<math.h>


void runge_kutta_4(double,double,double,int);
```

```
double f(double t,double y)
{
	return (t+sqrt(y));
}
void main()
{
	int limit;
	double ti,yi,h;
	ti=0;
	yi=1;
	h=0.2;
	limit=1;
	runge_kutta_4(ti,yi,h,limit);
	getch();
}
void runge_kutta_4(double ti,double yi,double
h,int limit)
{
	double yi_1,k0,k1,k2,k3;
	yi_1=yi;
	printf("========RUNGE-KUTTA FORTH
ORDER METHOD========\n\n");
```

```
    printf("\nt\t\t|\tSolution\n");
    printf("_____________________________________
_____");
    while(ti<=limit)
    {
        yi=yi_1;
        printf("\n%0.2lf\t\t|\t%lf",ti,yi);
        k0=h*f(ti,yi);
        k1=h*f(ti+(h/2),yi+(k0/2));
        k2=h*f(ti+(h/2),yi+(k1/2));
        k3=h*f(ti+h,yi+k2);
        yi_1=yi + (k0+2*k1+2*k2+k3)/6;
        ti+=h;
    }
    printf("\n\n-->Solution With RUNGE-KUTTA
FORTH ORDER METHOD= %lf",yi);
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

=========RUNGE-KUTTA FORTH ORDER METHOD=========

t	Solution
0.00	1.000000
0.20	1.230632
0.40	1.524809
0.60	1.885413
0.80	2.314716
1.00	2.814506

-->Solution With RUNGE-KUTTA FORTH ORDER METHOD= 2.814506

==========
==========

==========
**

==========
==========
==========
**

Q(6): Find the solution of differential equation $dy/dt = 1/2\ (t+y)$, for $y\ (2.0)$ given

$y(0) = 2$

$y(0.5) = 2.636$

$y(1.0) = 3.595$

and $y(1.5) = 4.968$, use $h = 0.5$

using (i) Milne-Simpson's predictor corrector method

(ii) Adam-Bashforth-Moulton's predictor-corrector method

```
==========================================
==========================================
==========================================
**
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define epsilon 0.00005
void
milne_simpsone_predictor_corrector(double[],do
uble[],double);
void
adam_bashforth_moultons_predictor_corrector(d
ouble[],double[],double);


double f(double t,double y)
{
    return ((t+y)/2);
}
void main()
{
    double h,y[10],t[10];
    h=0.5;
```

```
    y[0]=2;
    y[1]=2.636;
    y[2]=3.595;
    y[3]=4.968;
    t[0]=0;
    t[1]=0.5;
    t[2]=1.0;
    t[3]=1.5;
    t[4]=2.0;
    milne_simpsone_predictor_corrector(y,t,h);
    adam_bashforth_moultons_predictor_correct
or(y,t,h);
    getch();
}
void milne_simpsone_predictor_corrector(double
y[],double t[],double h)
{
    double yi_old=0;
    int i;
    i=3;
    printf("========milne_simpsone_predicto
r_corrector METHOD========\n\n");
    //predictor Method
```

```
    y[i+1]=y[i-3]+(4*h)*(2*f(t[i],y[i])-f(t[i-
1],y[i-1])+2*f(t[i-2],y[i-2]))/3;
    printf("Using Predictor Formula y4
=%lf",y[i+1]);



    //Corrector formula
    while(fabs(yi_old-y[i+1])>epsilon)
    {
        yi_old=y[i+1];
        y[i+1]=y[i-1] + (h/3)
*(f(t[i+1],y[i+1])+ 4* f(t[i],y[i])+f(t[i-1],y[i-
1]));
        printf("\n-->Using Corrector Formula
y4=%lf",y[i+1]);
    }
    printf("\n\n---->Solution With
milne_simpsone_predictor_corrector METHOD=
%lf\n\n",y[i+1]);
}


void
adam_bashforth_moultons_predictor_corrector(d
ouble y[],double t[],double h)
```

```
{
    double yi_old=0;
    int i;
    i=3;
    printf("========adam_bashforth_moulto
ns_predictor_corrector
METHOD========\n\n");
    //predictor Method
    y[i+1]=y[i]+(h/24)*(55*f(t[i],y[i])-59*f(t[i-
1],y[i-1])+37*f(t[i-2],y[i-2])-9*f(t[i-3],y[i-3]));
    printf("Using Predictor Formula y4
=%lf",y[i+1]);



    //Corrector formula
    while(fabs(yi_old-y[i+1])>epsilon)
    {
        yi_old=y[i+1];
        y[i+1]=y[i] + (h/24)
*(9*f(t[i+1],y[i+1])+ 19 * f(t[i],y[i])-5*f(t[i-
1],y[i-1])+f(t[i-2],y[i-2]));
        printf("\n-->Using Corrector Formula
y4=%lf",y[i+1]);
    }
```

```
    printf("\n\n---->Solution With
adam_bashforth_moultons_predictor_corrector
METHOD= %lf",y[i+1]);
}


==================================
==================================
==================================
**
==================================
==================================
==================================
**
```

output:

```
=========milne_simpsone_predictor_corrector
METHOD=========


Using Predictor Formula y4 =6.871000
-->Using Corrector Formula y4=6.873167
-->Using Corrector Formula y4=6.873347
-->Using Corrector Formula y4=6.873362


---->Solution With
milne_simpsone_predictor_corrector METHOD=
6.873362
```

=========adam_bashforth_moultons_predictor_corrector METHOD========

Using Predictor Formula y4 =6.870781

-->Using Corrector Formula y4=6.873104

-->Using Corrector Formula y4=6.873322

-->Using Corrector Formula y4=6.873343

---->Solution With adam_bashforth_moultons_predictor_corrector METHOD= 6.873343

==========
==========
==========
**

==========
==========
==========
**

Q(7): Use Adam-Bashforth-Moulton's predictor-corrector method to obtain the solution of

the equation dy/dx= 1-xy/x^2 at x = 1.4, where y(1) = 1.

Compute y(1.1), y(1.2) and y(1.3) using Runge-Kutta second order method.

Tabulate the results obtained thus.

==========
==========
==========
**

```
#include<stdio.h>

#include<conio.h>

#include<math.h>
```

```
#define epsilon 0.00005


void
adam_bashforth_moultons_predictor_corrector(d
ouble[],double[],double);
double
runge_kutta_2(double,double,double,double);
double f(double x,double y)
{
    return ((1-x*y)/(x*x));
}
void main()
{
    double h,y[10],x[10];
    h=0.1;
    x[0]=1;
    x[1]=1.1;
    x[2]=1.2;
    x[3]=1.3;
    x[4]=1.4;

    y[0]=1;
    y[1]=runge_kutta_2(x[0],y[0],h,1.2);
```

```
    y[2]=runge_kutta_2(x[0],y[0],h,1.3);
    y[3]=runge_kutta_2(x[0],y[0],h,1.4);

    printf("\n=========================
===By Runge-Kutta second order method\n");
    printf("y(1.1)=%lf\ny(1.2)=%lf\ny(1.3)=%lf
\n\n",y[1],y[2],y[3]);
    adam_bashforth_moultons_predictor_correct
or(y,x,h);
    getch();
}

void
adam_bashforth_moultons_predictor_corrector(d
ouble y[],double x[],double h)
{
    double yi_old=0;
    int i;
    i=3;
    printf("========adam_bashforth_moulto
ns_predictor_corrector
METHOD========\n\n");
    //predictor Method
```

```
    y[i+1]=y[i]+(h/24)*(55*f(x[i],y[i])-59*f(x[i-
1],y[i-1])+37*f(x[i-2],y[i-2])-9*f(x[i-3],y[i-3]));
    printf("Using Predictor Formula y(1.4)
=%lf",y[i+1]);



    //Corrector formula
    while(fabs(yi_old-y[i+1])>epsilon)
    {
        yi_old=y[i+1];
        y[i+1]=y[i] + (h/24)
*(9*f(x[i+1],y[i+1])+ 19 * f(x[i],y[i])-5*f(x[i-
1],y[i-1])+f(x[i-2],y[i-2]));
        printf("\n-->Using Corrector Formula
y(1.4)=%lf",y[i+1]);
    }
    printf("\n\n---->Solution With
adam_bashforth_moultons_predictor_corrector
METHOD= %lf",y[i+1]);
}


double runge_kutta_2(double xi,double yi,double
h,double limit)
{
```

```
    double yi_1,k0,k1;
    yi_1=yi;
    while(xi<limit)
    {
        yi=yi_1;
        k0=h*f(xi,yi);
        k1=h*f(xi+h,yi+k0);
        yi_1=yi + (0.5)*(k0+k1);
        xi+=h;
    }
    return yi;
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

==========By Runge-Kutta second order method

y(1.1)=0.995868

y(1.2)=0.985480

y(1.3)=0.971311

=========adam_bashforth_moultons_predictor_corrector METHOD========

Using Predictor Formula y(1.4) =0.954695

-->Using Corrector Formula y(1.4)=0.954878

-->Using Corrector Formula y(1.4)=0.954873

---->Solution With adam_bashforth_moultons_predictor_corrector METHOD= 0.954873

==========
==========
==========
**

==========
==========
==========
**

Q(8): Use Milne Simpson predictor corrector method to obtain the solution of

the equation dy/dx= 1-xy/x^2 at x = 1.4, where y(1) = 1.

Compute y(1.1), y(1.2) and y(1.3) using Runge-Kutta fourth order method.

Tabulate the results obtained thus.

==========
==========
==========
**

```
#include<stdio.h>

#include<conio.h>
```

```
#include<math.h>
#define epsilon 0.00005

void
milne_simpsone_predictor_corrector(double[],do
uble[],double);
double
runge_kutta_4(double,double,double,double);
double f(double x,double y)
{
    return ((1-x*y)/(x*x));
}
void main()
{
    double h,y[10],x[10];
    h=0.1;
    x[0]=1;
    x[1]=1.1;
    x[2]=1.2;
    x[3]=1.3;
    x[4]=1.4;

    y[0]=1;
```

```
    y[1]=runge_kutta_4(x[0],y[0],h,1.2);
    y[2]=runge_kutta_4(x[0],y[0],h,1.3);
    y[3]=runge_kutta_4(x[0],y[0],h,1.4);


    printf("\n=========================
===By Runge-Kutta Forth order method\n");
    printf("y(1.1)=%lf\ny(1.2)=%lf\ny(1.3)=%lf
\n\n",y[1],y[2],y[3]);
    milne_simpsone_predictor_corrector(y,x,h);
    getch();
}
void milne_simpsone_predictor_corrector(double
y[],double x[],double h)
{
    double yi_old=0;
    int i;
    i=3;
    printf("========milne_simpsone_predicto
r_corrector METHOD========\n\n");
    //predictor Method
    y[i+1]=y[i-3]+(4*h)*(2*f(x[i],y[i])-f(x[i-
1],y[i-1])+2*f(x[i-2],y[i-2]))/3;
```

```
    printf("Using Predictor Formula
y(1.4)=%lf",y[i+1]);



    //Corrector formula
    while(fabs(yi_old-y[i+1])>epsilon)
    {
        yi_old=y[i+1];
        y[i+1]=y[i-1] + (h/3)
*(f(x[i+1],y[i+1])+ 4* f(x[i],y[i])+f(x[i-1],y[i-
1]));
        printf("\n-->Using Corrector Formula
y(1.4)=%lf",y[i+1]);
    }
    printf("\n\n---->Solution With
milne_simpsone_predictor_corrector METHOD=
%lf\n\n",y[i+1]);
}
double runge_kutta_4(double xi,double yi,double
h,double limit)
{
    double yi_1,k0,k1,k2,k3;
    yi_1=yi;
    while(xi<limit)
```

```
    {
        yi=yi_1;
        k0=h*f(xi,yi);
        k1=h*f(xi+(h/2),yi+(k0/2));
        k2=h*f(xi+(h/2),yi+(k1/2));
        k3=h*f(xi+h,yi+k2);
        yi_1=yi + (k0+2*k1+2*k2+k3)/6;
        xi+=h;
    }
    return yi;
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

==========By Runge-Kutta Forth order method

y(1.1)=0.995737

y(1.2)=0.985268

y(1.3)=0.971050

========milne_simpsone_predictor_corrector METHOD========

Using Predictor Formula y(1.4)=0.954478

-->Using Corrector Formula y(1.4)=0.954629

-->Using Corrector Formula y(1.4)=0.954626

---->Solution With milne_simpsone_predictor_corrector METHOD= 0.954626

==========
==========
==========
**

==========
==========
==========
**

Q(9): From the following table estimate y'(1.1) and y'(1.2) using 3 point formulas and 5 point formulas

x 1.0 1.05 1.10 1.15 1.20 1.25 1.30

y 1.0 1.0247 1.0488 1.0724 1.0954 1.1180 1.1402

==========
==========
==========
**

```
#include<stdio.h>
```

```
#include<conio.h>
#include<math.h>

void
_3point_formulas(double[],double[],double);
void
_5point_formulas(double[],double[],double);

void main()
{
    double x[10],y[10],h=0.5;
    x[0]=1.0;
    x[1]=1.05;
    x[2]=1.10;
    x[3]=1.15;
    x[4]=1.20;
    x[5]=1.25;
    x[6]=1.30;
    y[0]=1.0;
    y[1]=1.0247;
    y[2]=1.0488;
    y[3]=1.0724;
```

```
    y[4]=1.0954;
    y[5]=1.1180;
    y[6]=1.1402;

    _3point_formulas(x,y,h);
    _5point_formulas(x,y,h);
    getch();
}

void _3point_formulas(double x[],double
y[],double h)
{

    double x0=x[2],ans;
    int i=2;
    //Endpoint formula
    printf("\n================3 Pont End
Point Formula================\n");
    ans=(1/(2*h)) * (-3 * y[i] + 4*y[i+1]-
y[i+2]);
    printf("\n--->y(1.1)'=%lf",ans);
    i=4;
```

```
    ans=(1/(2*h)) * (-3 * y[i] + 4*y[i+1]-
y[i+2]);
    printf("\n--->y(1.2)'=%lf",ans);
    //Midpoint Formula
    i=2;
    printf("\n===============3 Pont Mid
Point Formula===============\n");
    ans=(1/(2*h)) * (-y[i-1] + y[i+1]);
    printf("\n--->y(1.1)'=%lf",ans);
    i=4;
    ans=(1/(2*h)) * (-y[i-1] + y[i+1]);
    printf("\n--->y(1.2)'=%lf",ans);
    //Endpoint formula
    printf("\n===============3 Pont End
Point Formula===============\n");
    i=2;
    ans=(1/(2*h)) * (y[i-2] - 4*y[i-1]+3*y[i]);
    printf("\n--->y(1.1)'=%lf",ans);
    i=4;
    ans=(1/(2*h)) * (y[i-2] - 4*y[i-1]+3*y[i]);
    printf("\n--->y(1.2)'=%lf",ans);
}
```

```
void _5point_formulas(double x[],double
y[],double h)
{

    double x0=x[2],ans;
    int i=2;
    //Endpoint formula
    printf("\n\n\n===============5 Pont
End Point Formula===============\n");
    ans=(1/(12*h)) * ( -25*y[i] +48*y[i+1]-36*
y[i+2]+16*y[i+3]-3*y[i+4]);
    printf("\n--->y(1.1)'=%lf",ans);

    //Midpoint Formula
    i=2;
    printf("\n===============5 Pont Mid
Point Formula===============\n");
    ans=(1/(12*h)) * ( y[i-2] - 8*y[i-1]+8*
y[i+1]-y[i+2]);
    printf("\n--->y(1.1)'=%lf",ans);
    i=4;
```

```
    ans=(1/(12*h)) * ( y[i-2] - 8*y[i-1]+8*
y[i+1]-y[i+2]);
    printf("\n--->y(1.2)'=%lf",ans);
}
```

==========
==========
==========
**

==========
==========
==========
**

output:

==========3 Pont End Point Formula==========

--->y(1.1)'=0.047800

--->y(1.2)'=0.045600

==========3 Pont Mid Point Formula==========

--->y(1.1)'=0.047700

--->y(1.2)'=0.045600

==========3 Pont End Point Formula==========

--->y(1.1)'=0.047600

--->y(1.2)'=0.045400

==========5 Pont End Point Formula==========

--->y(1.1)'=0.048033

==========5 Pont Mid Point Formula==========

--->y(1.1)'=0.047700

--->y(1.2)'=0.045567

==========
==========
==========
**