Roots Of Quadratic Equation

Applications:

1. **Computer Science:** Utilized in algorithms for root-finding in graphics, optimization problems, and machine learning.
2. **Mathematics:** Used to solve polynomial equations and other quadratic equations where traditional analytical methods are not feasible.

Code:

```
#include <stdio.h>

#include <math.h>

int main() {

  printf("\n\nName              : Dishank Kumar\n");

  printf("Section           : ARQ\n");

  printf("University Roll No. : 2022026\n");

  printf("Class No.         : 20\n\n");

  double a, b, c, discriminant, root1, root2, realPart, imagPart;

  printf("Enter coefficients a, b and c: ");

  scanf("%lf %lf %lf", &a, &b, &c);

  discriminant = b * b - 4 * a * c;

  if (discriminant > 0) {

    root1 = (-b + sqrt(discriminant)) / (2 * a);

    root2 = (-b - sqrt(discriminant)) / (2 * a);

    printf("Roots are real and different.\n");

    printf("root1 = %.2lf and root2 = %.2lf\n", root1, root2);

  } else if (discriminant == 0) {

    root1 = root2 = -b / (2 * a);

    printf("Roots are real and the same.\n");

    printf("root1 = root2 = %.2lf\n", root1);

  } else {

    realPart = -b / (2 * a);

    imagPart = sqrt(-discriminant) / (2 * a);
```

```c
    printf("Roots are complex and different.\n");
    printf("root1 = %.2lf + %.2lfi and root2 = %.2lf - %.2lfi\n", realPart, imagPart, realPart, imagPart);
  }
  return 0;
}
```

Output:

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?) { gcc tempC
odeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }


Name                 : Dishank Kumar
Section              : ARQ
University Roll No. : 2022026
Class No.            : 20

Enter coefficients a, b and c: 3
2
1
Roots are complex and different.
root1 = -0.33 + 0.47i and root2 = -0.33 - 0.47i
PS C:\Users\honey\Desktop\CBNST>
```

Bisection Method

Application:

1. **Computer Science**: It's applied in algorithms that require root-finding, such as in graphics for ray tracing, and in machine learning for optimization problems.
2. **Mathematics**: Mathematicians use it to solve polynomial equations and other nonlinear equations where analytical solutions are not feasible

Code:

```
#include <stdio.h>

#include <math.h>

double func(double x) {

  return x * x * x - x - 2;

}

void bisection(double a, double b, double tol) {

  double c;

  if (func(a) * func(b) >= 0) {

    printf("You have not assumed the right a and b.\n");

    return;

  }

  c = a;

  while ((b - a) >= tol) {

    c = (a + b) / 2;


    if (func(c) == 0.0)

      break;

    else if (func(c) * func(a) < 0)

      b = c;

    else

      a = c;

  }

  printf("The value of the root is : %lf\n\n", c);
```

```
}
int main() {
    printf("\n\nName               : Dishank Kumar\n");
    printf("Section            : ARQ\n");
    printf("University Roll No. : 2022026\n");
    printf("Class No.          : 20\n\n");
    double a = -2, b = 3, tol = 0.0001;
    bisection(a, b, tol);
    return 0;
}
```

Output :

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?
) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeR
unnerFile }


Name                : Dishank Kumar
Section             : ARQ
University Roll No. : 2022026
Class No.           : 20

The value of the root is : 1.521337

PS C:\Users\honey\Desktop\CBNST>
```

Secant Method

Applications :

1. **Computer Science**: It's used in algorithms that require root-finding, such as in computer graphics for rendering and ray tracing, and in machine learning for optimization problems.
2. **Mathematics**: Mathematicians use it to solve polynomial equations and other nonlinear equations where traditional analytical methods are not feasible.

Code:

```
#include <stdio.h>

#include <math.h>

double func(double x) {

  return x * x * x - x - 2;

}

void secant(double x0, double x1, double tol, int max_iter) {

  double x2, f0, f1, f2;

  int iter = 0;

  do {

    f0 = func(x0);

    f1 = func(x1);


    if (fabs(f1 - f0) < tol) {

      printf("Mathematical error: Division by zero.\n");

      return;

    }

    x2 = x1 - (x1 - x0) * f1 / (f1 - f0);

    f2 = func(x2);

    x0 = x1;

    x1 = x2;

    iter++;

    if (iter > max_iter) {
```

```
        printf("Not convergent.\n");
        return;
      }
  } while (fabs(f2) > tol);
  printf("The value of the root is : %lf\n", x2);
}

int main() {
  printf("\n\nName              : Dishank Kumar\n");
  printf("Section           : ARQ\n");
  printf("University Roll No. : 2022026\n");
  printf("Class No.         : 20\n\n");
  double x0 = -2, x1 = 3, tol = 0.0001;
  int max_iter = 100;
  secant(x0, x1, tol, max_iter);
  return 0;
}
```

Output :

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?
) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeR
unnerFile }


Name                : Dishank Kumar
Section             : ARQ
University Roll No. : 2022026
Class No.           : 20

The value of the root is : 1.521380
PS C:\Users\honey\Desktop\CBNST>
```

Iteration Method

Applications:

1. **Computer Science:** Utilized in algorithms for root-finding in graphics, optimization problems, and machine learning. For example, it can be used to find the roots of polynomial equations in computer graphics or to optimize functions in machine learning models.
2. **Mathematics:** Used to solve polynomial and other nonlinear equations where traditional analytical methods are not feasible. It is particularly useful in numerical analysis for approximating functions and solving differential equations.

Code:

```
#include <stdio.h>
#include <math.h>
#define f(x) cos(x)-3*x+1
#define g(x) (1+cos(x))/3
int main() {
  printf("\n\nName             : Dishank Kumar\n");
  printf("Section          : ARQ\n");
  printf("University Roll No. : 2022026\n");
  printf("Class No.        : 20\n\n");
  int step = 1, N;
  float x0, x1, e;
  printf("Enter initial guess: ");
  scanf("%f", &x0);
  printf("Enter tolerable error: ");
  scanf("%f", &e);
  printf("Enter maximum iteration: ");
  scanf("%d", &N);
  printf("\nStep\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");
  do {
    x1 = g(x0);
    printf("%d\t%f\t%f\t%f\t%f\n", step, x0, f(x0), x1, f(x1));
```

```
    step = step + 1;
    if (step > N) {
      printf("Not Convergent.");
      return 0;
    }
    x0 = x1;
  } while (fabs(f(x1)) > e);
  printf("\nRoot is %f", x1);
  return 0;
}
```

Output:

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?) { gcc tempC
odeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Name                 : Dishank Kumar
Section              : ARQ
University Roll No.  : 2022026
Class No.            : 20

Enter initial guess: 1
Enter tolerable error: 0.00001
Enter maximum iteration: 20

Step    x0          f(x0)        x1          f(x1)
1       1.000000    -1.459698    0.513434    0.330761
2       0.513434    0.330761     0.623688    -0.059333
3       0.623688    -0.059333    0.603910    0.011391
4       0.603910    0.011391     0.607707    -0.002162
5       0.607707    -0.002162    0.606986    0.000411
6       0.606986    0.000411     0.607124    -0.000078
7       0.607124    -0.000078    0.607098    0.000015
8       0.607098    0.000015     0.607102    -0.000003

Root is 0.607102
PS C:\Users\honey\Desktop\CBNST>
```

Regula Falsi Method

Applications :

1. **Computer Science**: It's utilized in algorithms for root-finding in graphics, optimization problems, and machine learning.
2. **Mathematics**: Mathematicians use it to solve polynomial equations and other nonlinear equations where traditional analytical methods are not feasible.

Code:

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

double func(double x) {
  return x * x * x - x - 2;
}
void regulaFalsi(double a, double b, double tol, int max_iter) {
  double c;
  int iter = 0;
  if (func(a) * func(b) >= 0) {
    printf("You have not assumed the right a and b.\n");
    return;
  }
  do {
    c = (a * func(b) - b * func(a)) / (func(b) - func(a));
    if (func(c) == 0.0)
      break;
    else if (func(c) * func(a) < 0)
      b = c;
    else
      a = c;
    iter++;
```

```
    if (iter > max_iter) {
      printf("Not convergent.\n");
      return;
    }
  } while (fabs(func(c)) > tol);
  printf("The value of the root is : %lf\n", c);
}
int main() {
  double a = -2, b = 3, tol = 0.0001;
  int max_iter = 100;
  regulaFalsi(a, b, tol, max_iter);
  return 0;
}
```

Output :

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?
) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeR
unnerFile }


Name                 : Dishank Kumar
Section              : ARQ
University Roll No.  : 2022026
Class No.            : 20

The value of the root is : 1.521364
PS C:\Users\honey\Desktop\CBNST>
```

Newton Raphson Method

Applications:

1. **Computer Science:** Utilized in algorithms for root-finding in graphics, optimization problems, and machine learning.
2. **Mathematics:** Used to solve polynomial and other nonlinear equations where traditional analytical methods are not feasible.

Code:

```
#include <stdio.h>

#include <math.h>

double func(double x) {

  return x * x * x - x - 2; // Example function: x^3 - x - 2

}

double derivFunc(double x) {

  return 3 * x * x - 1; // Derivative of the function: 3x^2 - 1

}

void newtonRaphson(double x0, double tol, int max_iter) {

  double x1;

  int iter = 0;

  printf("Step\t x0\t\t f(x0)\t\t x1\t\t f(x1)\n");

  do {

    double f0 = func(x0);

    double df0 = derivFunc(x0);

    if (df0 == 0.0) {

      printf("Mathematical error: Division by zero.\n");

      return;

    }

    x1 = x0 - f0 / df0;

    printf("%d\t %f\t %f\t %f\t %f\n", iter, x0, f0, x1, func(x1));

    if (fabs(x1 - x0) < tol) {

      printf("The value of the root is : %lf\n", x1);
```

```
        return;
    }
    x0 = x1;
    iter++;
    if (iter > max_iter) {
        printf("Not convergent.\n");
        return;
    }
  } while (fabs(func(x1)) > tol);
}
int main() {
  printf("\n\nName              : Dishank Kumar\n");
  printf("Section           : ARQ\n");
  printf("University Roll No. : 2022026\n");
  printf("Class No.         : 20\n\n");
  double x0 = 1.0, tol = 0.0001;
  int max_iter = 100;
  newtonRaphson(x0, tol, max_iter);
  return 0;
}
```

Output:

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?)
{ gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunne
rFile }


Name                : Dishank Kumar
Section             : ARQ
University Roll No. : 2022026
Class No.           : 20

Step    x0              f(x0)           x1              f(x1)
0       1.000000        -2.000000       2.000000        4.000000
1       2.000000        4.000000        1.636364        0.745304
2       1.636364        0.745304        1.530392        0.053939
3       1.530392        0.053939        1.521441        0.000367
4       1.521441        0.000367        1.521380        0.000000
The value of the root is : 1.521380
PS C:\Users\honey\Desktop\CBNST>
```

Newton Forward Interpolation Method

Applications :

1. **Computer Science:** Utilized in graphics and data visualization to estimate values between known data points.
2. **Mathematics:** Used in numerical analysis to approximate functions and solve differential equations

Code :

```
#include <stdio.h>

void forward(float x[], float y[][4], int n);

int main() {

  printf("\n\nName             : Dishank Kumar\n");

  printf("Section          : ARQ\n");

  printf("University Roll No. : 2022026\n");

  printf("Class No.        : 20\n\n");

  int i, j, n;

  float x[4], y[4][4];

  printf("Enter the number of arguments (max 4):\n");

  scanf("%d", &n);

  printf("Enter the values of x:\n");

  for (i = 0; i < n; i++) {

    scanf("%f", &x[i]);

  }

  printf("Enter the values of y:\n");

  for (i = 0; i < n; i++) {

    scanf("%f", &y[i][0]);

  }

  forward(x, y, n);

  return 0;

}
```

```
void forward(float x[], float y[][4], int n) {
  int i, j;
  float a, h, u, sum, p;
  printf("Enter the interpolation point for forward method:\n");
  scanf("%f", &a);
  for (j = 1; j < n; j++) {
    for (i = 0; i < n - j; i++) {
      y[i][j] = y[i + 1][j - 1] - y[i][j - 1];
    }
  }
  printf("\nThe forward difference table is:\n");
  for (i = 0; i < n; i++) {
    for (j = 0; j < n - i; j++) {
      printf("%f\t", y[i][j]);
    }
    printf("\n");
  }
  p = 1.0;
  sum = y[0][0];
  h = x[1] - x[0];
  u = (a - x[0]) / h;
  for (j = 1; j < n; j++) {
    p = p * (u - j + 1) / j;
    sum = sum + p * y[0][j];
  }
  printf("The value of y at x=%0.1f is %0.3f\n", a, sum);
}
```

Output :

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?
) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeR
unnerFile }


Name                 : Dishank Kumar
Section              : ARQ
University Roll No. : 2022026
Class No.            : 20

Enter the number of arguments (max 4):
4
Enter the values of x:
1891
1901
1911
1921
Enter the values of y:
20
37
50
69
Enter the interpolation point for forward method:
1899

The forward difference table is:
20.000000       17.000000       -4.000000       10.000000
37.000000       13.000000       6.000000
50.000000       19.000000
69.000000
The value of y at x=1899.0 is 34.240
PS C:\Users\honey\Desktop\CBNST> 
```

Newton Backward Interpolation Method

Applications:

1. **Computer Science:** Utilized in graphics and data visualization to estimate values between known data points, especially when the required value is near the end of the dataset.
2. **Mathematics:** Used in numerical analysis to approximate functions and solve differential equations where the data points are near the end of the range.

Code:

```
#include <stdio.h>

void backward(float x[], float y[][4], int n);

int main() {

  printf("\n\nName             : Dishank Kumar\n");

  printf("Section          : ARQ\n");

  printf("University Roll No. : 2022026\n");

  printf("Class No.        : 20\n\n");

  int i, j, n;

  float x[4], y[4][4];

  printf("Enter the number of arguments (max 4):\n");

  scanf("%d", &n);

  printf("Enter the values of x:\n");

  for (i = 0; i < n; i++) {

    scanf("%f", &x[i]);

  }

  printf("Enter the values of y:\n");

  for (i = 0; i < n; i++) {

    scanf("%f", &y[i][0]);

  }

  backward(x, y, n);

  return 0;

}
```

```
void backward(float x[], float y[][4], int n) {
    int i, j;
    float a, h, u, sum, p;
    printf("Enter the interpolation point for backward method:\n");
    scanf("%f", &a);
    for (j = 1; j < n; j++) {
        for (i = j; i < n; i++) {
            y[i][j] = y[i][j - 1] - y[i - 1][j - 1];
        }
    }
    printf("\nThe backward difference table is:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j <= i; j++) {
            printf("%f\t", y[i][j]);
        }
        printf("\n");
    }
    p = 1.0;
    sum = y[n - 1][0];
    h = x[1] - x[0];
    u = (a - x[n - 1]) / h;
    for (j = 1; j < n; j++) {
        p = p * (u + j - 1) / j;
        sum = sum + p * y[n - 1][j];
    }
    printf("The value of y at x=%0.1f is %0.3f\n", a, sum);
}
```

Output:

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?)
{ gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunne
rFile }


Name                   : Dishank Kumar
Section                : ARQ
University Roll No. : 2022026
Class No.              : 20

Enter the number of arguments (max 4):
4
Enter the values of x:
1891
1901
1911
1921
Enter the values of y:
20
37
50
69
Enter the interpolation point for backward method:
1915

The backward difference table is:
20.000000
37.000000       17.000000
50.000000       13.000000       -4.000000
69.000000       19.000000       6.000000        10.000000
The value of y at x=1915.0 is 56.320
PS C:\Users\honey\Desktop\CBNST>
```

Gauss Forward Interpolation Method

Applications:

Code:

```
#include <stdio.h>
#include <math.h>
float p_cal(float p, int n) {
    float temp = p;
    for (int i = 1; i < n; i++) {
        if (i % 2 == 1)
            temp *= (p - i);
        else
            temp *= (p + i);
    }
    return temp;
}

int fact(int n) {
    int f = 1;
    for (int i = 2; i <= n; i++) {
        f *= i;
    }
    return f;
}
int main() {
    printf("\n\nName            : Dishank Kumar\n");
    printf("Section           : ARQ\n");
    printf("University Roll No. : 2022026\n");
    printf("Class No.         : 20\n\n");
    int n;
    printf("Enter the number of data points:\n");
    scanf("%d", &n);
```

```
float x[n], y[n][n];
printf("Enter the values of x:\n");
for (int i = 0; i < n; i++) {
  scanf("%f", &x[i]);
}
printf("Enter the values of y:\n");
for (int i = 0; i < n; i++) {
  scanf("%f", &y[i][0]);
}
for (int i = 1; i < n; i++) {
  for (int j = 0; j < n - i; j++) {
    y[j][i] = round((y[j + 1][i - 1] - y[j][i - 1]) * 10000) / 10000.0;
  }
}
printf("\nThe forward difference table is:\n");
for (int i = 0; i < n; i++) {
  printf("%f\t", x[i]);
  for (int j = 0; j < n - i; j++) {
    printf("%f\t", y[i][j]);
  }
  printf("\n");
}
float value;
printf("Enter the value of x to predict y:\n");
scanf("%f", &value);
float sum = y[n / 2][0];
float p = (value - x[n / 2]) / (x[1] - x[0]);
for (int i = 1; i < n; i++) {
  sum += (p_cal(p, i) * y[(n - i) / 2][i]) / fact(i);
}
printf("\nValue at %0.2f is %0.4f\n", value, sum);
```

```
    return 0;
}
```

Output:

```
Name                  : Dishank Kumar
Section               : ARQ
University Roll No. : 2022026
Class No.             : 20

Enter the number of data points:
5
Enter the values of x:
1901
1911
1921
1931
1941
Enter the values of y:
100
140
230
290
350

The forward difference table is:
1901.000000     100.000000      40.000000       50.000000       -80.000000   110.000000
1911.000000     140.000000      90.000000       -30.000000      30.000000
1921.000000     230.000000      60.000000       0.000000
1931.000000     290.000000      60.000000
1941.000000     350.000000
Enter the value of x to predict y:
1915

Value at 1915.00 is 164.1440
PS C:\Users\honey\Desktop\CBNST>
```

Gauss Backward Interpolation Method

Applications:

1. **Computer Science:** Utilized in graphics and data visualization to estimate values between known data points, especially when the required value is near the end of the dataset.
2. **Mathematics:** Used in numerical analysis to approximate functions and solve differential equations where the data points are near the end of the range.

Code:

```
#include <stdio.h>
#include <math.h>
float p_cal(float p, int n) {
    float temp = p;
    for (int i = 1; i < n; i++) {
        if (i % 2 == 1)
            temp *= (p + i);
        else
            temp *= (p - i);
    }
    return temp;
}

int fact(int n) {
    int f = 1;
    for (int i = 2; i <= n; i++) {
        f *= i;
    }
    return f;
}

int main() {
    printf("\n\nName            : Dishank Kumar\n");
```

```
printf("Section           : ARQ\n");
printf("University Roll No. : 2022026\n");
printf("Class No.         : 20\n\n");
int n;
printf("Enter the number of data points:\n");
scanf("%d", &n);
float x[n], y[n][n];
printf("Enter the values of x:\n");
for (int i = 0; i < n; i++) {
   scanf("%f", &x[i]);
}
printf("Enter the values of y:\n");
for (int i = 0; i < n; i++) {
   scanf("%f", &y[i][0]);
}
for (int i = 1; i < n; i++) {
   for (int j = n - 1; j >= i; j--) {
      y[j][i] = round((y[j][i - 1] - y[j - 1][i - 1]) * 10000) / 10000.0;
   }
}
printf("\nThe backward difference table is:\n");
for (int i = 0; i < n; i++) {
   printf("%f\t", x[i]);
   for (int j = 0; j <= i; j++) {
      printf("%f\t", y[i][j]);
   }
   printf("\n");
}
float value;
printf("Enter the value of x to predict y:\n");
scanf("%f", &value);
```

```
  float sum = y[n - 1][0];
  float p = (value - x[n - 1]) / (x[1] - x[0]);
  for (int i = 1; i < n; i++) {
    sum += (p_cal(p, i) * y[n - 1][i]) / fact(i);
  }
  printf("\nValue at %0.2f is %0.4f\n", value, sum);
  return 0;
}
```

Output:

```
PS C:\Users\honey\Desktop\CBNST> cd "c:\Users\honey\Desktop\CBNST\" ; if ($?) { gcc tempC
odeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }


Name                : Dishank Kumar
Section             : ARQ
University Roll No. : 2022026
Class No.           : 20

Enter the number of data points:
5
Enter the values of x:
1901
1911
1921
1931
1941
Enter the values of y:
20
35
67
100
156

The backward difference table is:
1901.000000     20.000000
1911.000000     35.000000       15.000000
1921.000000     67.000000       32.000000       17.000000
1931.000000     100.000000      33.000000       1.000000        -16.000000
1941.000000     156.000000      56.000000       23.000000       22.000000       38.000000
Enter the value of x to predict y:
1929

Value at 1929.00 is 86.5552
PS C:\Users\honey\Desktop\CBNST>
```

LAGRANGE INTERPOLATION METHOD

Applications:

1. **Data Interpolation**: Find intermediate values between known data points in fields like physics, economics, and engineering.
2. **Numerical Integration**: Used in Newton-Cotes formulas to approximate integrals from discrete data.
3. **Curve Fitting**: Fit curves through given data points to analyze trends or approximate functions.
4. **Computer Graphics**: Generate smooth transitions and animations by interpolating curves.
5. **Error Estimation**: Estimate and analyze errors in numerical approximations.

```
#include <stdio.h>

void main(){

  printf("Name : Mahesh Semwal\n");

  printf("Sec : A-rq\n");

  printf("Roll no. :  36\n\n");

  float point, Y = 0, p;

  int i, j, n;

  printf("Enter number of data: ");

  scanf("%d", &n);

  float x[n], y[n];

  printf("Enter data as X and Y:\n");

  for (i = 0; i < n; i++)

  {

    scanf("%f%f", &x[i], &y[i]);

  }

  printf("Enter interpolation point: ");

  scanf("%f", &point);

  for (i = 0; i < n; i++){

    p = 1;

    for (j = 0; j < n; j++){

      if (i != j){
```

```
            p = p * (point - x[j]) / (x[i] - x[j]);
        }
    }
    Y = Y + p * y[i];
  }
  printf("Interpolated value at %.4f is %f", point, Y);
}
```

```
manish@ASUS-VivoBook MINGW64 C:/Users/manish/AppData/Local/Programs/Microsoft VS Code (main)
$ ./a.exe
Name : Mahesh Semwal
Name : Mahesh Semwal
Sec : A-rq
Roll no. :  36

Enter number of data: 4
Name : Mahesh Semwal
Sec : A-rq
Roll no. :  36

Enter number of data: 4
Enter data as X and Y:
5 12
6 15
8 16
11 19
Enter interpolation point: 10
Interpolated value at 10.0000 is 16.555553(env)
```

SIMPSON’S ONE-THIRD RULE

Applications:

1. **Physics**: Calculating areas under velocity-time graphs to determine displacement.
2. **Engineering**: Estimating work done by varying forces.
3. **Probability**: Computing probabilities for complex probability density functions.
4. **Economics**: Approximating integrals in cost, revenue, and demand functions.
5. **Astronomy**: Determining distances using integral-based formulas in orbital mechanics.

```
#include<stdio.h>

double fun(double x){

  return 1.0/(1+x*x);

}

int main(){

  printf("Name : Mahesh Semwal\n");

  printf("Sec : A-rq\n");

  printf("Roll no. : 36\n\n");

  int n;

  double a,b;

  printf("enter the value of n: ");

  scanf("%d",&n);

  printf("Enter the value of a and b: ");

  scanf("%lf%lf",&a,&b);

  double h=(b-a)/(n-1);

  double y[n];

  for(int i=0;i<n;i++){

    y[i]=fun(a+i*h);

  }

  double sum=y[0]+y[n-1];

  for(int i=1;i<n-1;i++){

    if(i%2!=0){

      sum=sum+4*y[i];
```

```
    }
    else{
      sum=sum+2*y[i];
    }
  }
  printf("array y is: \n");
  for(int i=0;i<n;i++){
    printf("%lf ",y[i]);
  }
  sum=(h*sum)/3;
  printf("\nintegral value is: %lf",sum);

}
```

```
manish@ASUS-VivoBook MINGW64 C:/Users/manish/AppData/Local/Programs/Microsoft VS Code (main)
$ ./a.exe
Name : Mahesh Semwal
Sec : A-rq
Roll no. : 36

enter the value of n: 5
Enter the value of a and b: 0 1
array y is:
1.000000 0.941176 0.800000 0.640000 0.500000
integral value is: 0.785392(env)
```

SIMPSON’S THREE-EIGHT RULE

Applications:

1. **Physics**: Accurately calculates work done by forces with cubic or higher-order variations.
2. **Engineering**: Used in systems like fluid dynamics or heat transfer, where the function has non-linear behavior.
3. **Probability**: Computes probabilities for distributions with complex higher-order density functions.
4. **Economics**: Approximates integrals in models with rapidly changing variables, like cost or revenue curves.

```
#include<stdio.h>
double fun(double x){
  return 1.0/(1+x*x);
}

int main(){
  printf("Name : Mahesh Semwal\n");
  printf("Sec : A-rq\n");
  printf("Roll no. : 36\n\n");
  int n;
  double a,b;
  printf("enter the value of n: ");
  scanf("%d",&n);
  printf("Enter the value of a and b: ");
  scanf("%lf%lf",&a,&b);
  double h=(b-a)/(n-1);
  double y[n];
  for(int i=0;i<n;i++){
    y[i]=fun(a+i*h);
  }
  double sum=y[0]+y[n-1];
  for(int i=1;i<n-1;i++){
```

```
        if(i%3==0){
          sum=sum+2*y[i];
        }
        else{
          sum=sum+3*y[i];
        }
    }
    printf("array y is: \n");
    for(int i=0;i<n;i++){
        printf("%lf ",y[i]);
    }
    sum=(3.0*h*sum)/8;
    printf("\nintegral value is: %lf",sum);
}
```

```
manish@ASUS-VivoBook MINGW64 C:/Users/manish/AppData/Local/Programs/Microsoft VS Code (main)
$ ./a.exe
Name : Mahesh Semwal
Sec : A-rq
Roll no. : 36

enter the value of n: 7
Enter the value of a and b: 0 1
array y is:
1.000000 0.972973 0.900000 0.800000 0.692308 0.590164 0.500000
integral value is: 0.785396(env)
```

TRAPEZOIDAL

Applications:

1. **Physics**: Calculating approximate displacement from velocity-time data.
2. **Engineering**: Estimating flow rates or heat transfer in systems with linear variations.
3. **Medicine**: Computing drug concentration over time in pharmacokinetics.
4. **Environmental Science**: Measuring total pollutant levels from concentration-time curves.
5. **Geography**: Approximating areas of irregular land plots.

```
#include<stdio.h>

#include<math.h>

double fun(double x){

  return 1/(10+pow(x,3));

}


int main(){

  printf("Name : Mahesh Semwal\n");

  printf("Sec : A-rq\n");

  printf("Roll no. : 36\n\n");

  int n;

  double a,b;

  printf("enter the value of n: ");

  scanf("%d",&n);

  printf("Enter the value of a and b: ");

  scanf("%lf%lf",&a,&b);

  double h=(b-a)/(n-1);

  double y[n];

  for(int i=0;i<n;i++){

    y[i]=fun(a+i*h);

  }

  double sum=y[0]+y[n-1];
```

```
    for(int i=1;i<n-1;i++){
        sum=sum+2*y[i];
    }
    printf("array y is: \n");
    for(int i=0;i<n;i++){
        printf("%lf ",y[i]);
    }
    sum=(h*sum)/2;
    printf("\nintegral value is: %lf",sum);

}
```

```
manish@ASUS-VivoBook MINGW64 C:/Users/manish/AppData/Local/Programs/Microsoft VS Code (main)
$ ./a.exe
Name : Mahesh Semwal
Sec : A-rq
Roll no. : 36

enter the value of n: 5
Enter the value of a and b: 0 1
array y is:
0.100000 0.099844 0.098765 0.095952 0.090909
integral value is: 0.097504(env)
```

RUNGE-KUTTA METHOD

Applications:

1. **Physics**: Solving equations for motion, oscillations, and wave phenomena (e.g., pendulums, projectile motion).
2. **Engineering**: Modeling dynamic systems like robotics, vehicle motion, and control systems.
3. **Electronics**: Analyzing RLC circuits and transient responses.
4. **Astronomy**: Simulating orbital mechanics and celestial dynamics.
5. **Biology**: Modeling population dynamics and biological growth using differential equations.

```
#include <stdio.h>

float dydx(float x, float y)

{

  return x+y;

}


float rungeKutta(float x0, float y0, float x, float h)

{

  int n = (int)((x - x0) / h);

  float k1, k2, k3, k4, k5;

  float y = y0;

  for (int i = 1; i <= n; i++)

  {

    k1 = h * dydx(x0, y);

    k2 = h * dydx(x0 + 0.5 * h, y + 0.5 * k1);

    k3 = h * dydx(x0 + 0.5 * h, y + 0.5 * k2);

    k4 = h * dydx(x0 + h, y + k3);

    y = y + (1.0 / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4);

    x0 = x0 + h;

  }
```

```c
    return y;
}

int main()
{
    printf("Name : Mahesh Semwal\n");
    printf("Sec : A-rq\n");
    printf("Roll no. : 36\n\n");
    float x0 = 0, y = 1, x = 0.4, h = 0.1;
    printf("given value of x0 is: %f\n",x0);
    printf("given value of y at x=x0 is: %f\n",y);
    printf("\nThe value of y at x=0.4 is : %f",rungeKutta(x0, y, x, h));
    return 0;
}
```

```
manish@ASUS-VivoBook MINGW64 C:/Users/manish/AppData/Local/Programs/Microsoft VS Code (main)
$ ./a.exe
Name : Mahesh Semwal
Sec : A-rq
Roll no. : 36

given value of x0 is: 0.000000
given value of y at x=x0 is: 1.000000

The value of y at x=0.4 is : 1.583648(env)
```