Ain Shams University
Faculty of Engineering
Discipline Programs

Computer Programming Major Task Report

Computer Engineering and Software Systems (CESS)

Submitted to:

Dr. Mahmoud Khalil

Eng. Kyrollos

Submitted by:

Habiba Yasser 20P3072

Nadine Hisham 20P9880

Jumana Yasser 20P8421

Table of Contents

1.0 Milestone 1

1.1 Libraries Used:

ConsoleApplication58 (Global Scope)

```
#include <iostream>
#include <cstring>
#include <cmath>
#include<string>
#include<cstdlib>
#include<complex>
```

Figure 1: Libraries

- Cstring , String libraries were used for string functions
- Cmath was used for mathematical functions(cos , sin)
- Cstdlib was used to convert arrays to floats (atof())
- Complex library was used to find imaginary roots for higher orders

1.2 User Input:

```
int main()
{
    int x;
    int degree;
    cout << "Insert your polynomial in terms of x:\n";
    cout << "Example:5x^3 -x^2 +2.5x +1\n";
    char str[1000];
    cin.getline(str, 1000);
```

Figure 2: User Input

- User will input the polynomial they want in the form of the following example:
  5x^3 –x^2 +2.5x +1
- str[1000] is the array in which the user input will be stored to be used.
- str[1000]: 1000 was used in order to store bigger coefficients.

1.3 Errors

- **<u>Error 1:</u>**

```
// error 1
for (int i = 0; i < 1000; i++)
{
    if (str[i] == NULL && (str[i - 1] == '+' || str[i - 1] == '-'))
    {
        error1 = 1;
        break;
    }

    else
    {
        error1 = 0;
    }
}
```

Figure 3: Error 1

- In order to check if the last term is missing after a (+) or (-) we made a for loop with a thousand iterations (the maximum size of the used string).

- We used if conditions to check if the last element of the string is either (+) or (-).

- If the last term was in fact (+) or (-), we created a Boolean variable where we assigned 1 to error1 to show that there is an error.

- **<u>Error 2:</u>**

```
// error 2
for (int i = 0; i < 1000; i++)
{
    if (str[i] == '^')
    {
        if (!(str[++i] >= '0' && str[++i] <= '5'))
        {
            error2 = 1;
            break;
        }
        else  error2 = 0;
    }
}
```

Figure 4: Error 2

- In order to check whether or not after the ‘^’ there were numbers between (0) and (5) ,we made a for loop with a thousand iterations (the maximum size of the used string), that searches for ‘^’ using if condition and then checking if the element after it is not a number between (0) and (5) using a second if condition.

- If the last term was in fact not a number between (0) and (5) we created a Boolean variable where we assigned 1 to error2 to show that there is an error.

- **<u>Error 3:</u>**

```
// error 3
for (int i = 0; i < 1000; i++)
{
    if (str[i] == 'x')
    {
        if (str[++i] >= '0' && str[i] <= '9')
        {
            error3 = 1;
            break;
        }
        else  error3 = 0;
    }
}
```

Figure 5: Error 3

- In order to check whether or not the ‘^’ was missing between ‘x’ and the numbers, we made a for loop with a thousand iterations (the maximum size of the used string), that searches for ‘x’ using if condition and then checking if the element after it is a number between (0) and (9) using a second if condition.

- If there was no ‘^’ between ‘x’ and the numbers we created a Boolean variable where we assigned 1 to error3 to show that there is an error

1.4 Degree

```
for (int i = 0; i < 1000; i++)
{

    if (str[i] == '^')
    {
        i++;
        x = i;
        degree = str[x] - '0';
        break;
    }
    else if ((str[i] == 'x' && (str[i + 1] == '+' || str[i + 1] == '-')) || (str[i] == 'x' && (str[i + 1] == ' ' && (str[i + 2] == '+' || str[i + 2] == '-'))))
    {
        x = 1;
        degree = x;
        break;
    }
}
cout << "degree:" << degree << "\n";
```

Figure 7: Degree of Polynomial

- To check for the degree of the polynomial ,we made a for loop with a thousand iterations (the maximum size of the used string) to search for the first ‘^’ and store the number after it as an integer in ‘degree’. Once it finds the degree it breaks out of the for loop.

- To find the degree if it has a value of 1: we check if the first element of the string is(x) and the second is (+) or (-) OR the first element is (x), the second element is a space, the third element is either (+) or (-). Once the conditions are met the degree is set to be 1.Then, it breaks out of the for loop and prints the degree

1.5 Coefficients

```
    //coefficients
    char* token;
    double y;
    token = strtok(str, "+ ");
    y = atof(token);
    while (token != NULL)
    {
        for (int i = 0; i < size; i++)
        {

            y = atof(token);

            if (*token == '-' && *(token + 1) == 'x')
            {
                y = -1;
            }

            else if (*token == '-')
            {
                y = atof(token);

            }
            else if (*token == 'x')
            {
                y = 1;

            }
            cout << "coefficient:" << y << "\n";
            coeff[i] = y;
            token = strtok(NULL, "+ ");
        }
    }
```

Figure 8: Coefficients

- We defined the size of the dynamic array we'll use to store the coefficients as the degree+1.We defined the dynamic array to store floats and called it coeff.
- We started to tokenize the string, separating the terms between (+) and space.
- Then we define a variable called (y) to store the token into by using atof() to change the token to float.
- If token has the (-) sign and after it is the variable (x), then y = -1. If token has the (-) sign but there is a coefficient before (x) then (y) will equal the negative coefficient. If the token has (x) as the first variable then coefficient is equal to 1, so (y)=1.
- Then we store (y) into the dynamic array coeff[size] to store the coefficients in the dynamic array.

1.6 Using Coefficients

```cpp
    switch (degree)
    {
    case 1:
        frstorder(coeff[0], coeff[1]);
        break;
    case 2:
        scndorder(coeff[0], coeff[1], coeff[2]);
        break;
    case 3:
        thrdorder(coeff[0], coeff[1], coeff[2], coeff[3]);
        break;
    case 4:
        frthorder(coeff[0], coeff[1], coeff[2], coeff[3], coeff[4]);
        break;
    case 5:
        complex<double> p, q, r, s, t;
        complex<double> seed = 0.4 + 0.9i;
        complex<double> a = coeff[0];
        complex<double> b = coeff[1];
        complex<double> c = coeff[2];
        complex<double> d = coeff[3];
        complex<double> e = coeff[4];
        complex<double> f = coeff[5];
        p = 1.0;
        q = p * seed;
        r = q * seed;
        s = r * seed;
        t = s * seed;
        //cout << p << " , " << q << " , " << r << " , " << s << " , " << t << "\n";

        for (int i = 0; i < 3000; ++i)
        {
            p = p - ((p * (p * (p * (p * (a*p + b) + c) + d) + e) + f)/a) / ((p - q) * (p - r) * (p - s) * (p - t));
            q = q - ((q * (q * (q * (q * (a * q + b) + c) + d) + e) + f) / a) / ((q - p) * (q - r) * (q - s) * (q - t));
            r = r - ((r * (r * (r * (r * (a * r + b) + c) + d) + e) + f) / a) / ((r - p) * (r - q) * (r - s) * (r - t));
            s = s - ((s * (s * (s * (s * (a * s + b) + c) + d) + e) + f) / a) / ((s - p) * (s - q) * (s - r) * (s - t));
            t = t - ((t * (t * (t * (t * (a * t + b) + c) + d) + e) + f) / a) / ((t - p) * (t - q) * (t - r) * (t - s));

        }
        cout << "x1: " << p << "\n";
        cout << "x2: " << q << "\n";
        cout << "x3: " << r << "\n";
        cout << "x4: " << s<< "\n";
        cout << "x5: " << t << "\n";

        break;

    }
```

- Using Switch Case for the degree to use when calculating the roots of the polynomial.
- For the 1st Order, since it's a 1st degree polynomial, we only use 2 coefficients
- For the 2nd Order, since it's a 2nd degree polynomial, we only use 3 coefficients
- For the 3rd Order, since it's a 3rd degree polynomial, we only use 4 coefficients
- For the 4th Order, since it's a 4th degree polynomial, we only use 5 coefficients

1.7 Order

- **<u>1st Order:</u>**

```cpp
void frstorder(double a, double b)
{
    double x1 = -1 * (b / a);
    cout << x1;
}
```

Figure 10: 1st Order

- **<u>2nd Order:</u>**

```cpp
void scndorder(double a, double b, double c)
{
    double discriminant, realPart, imaginaryPart, x1, x2;


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

    if (discriminant > 0)
    {
        x1 = (-b + sqrt(discriminant)) / (2 * a);
        x2 = (-b - sqrt(discriminant)) / (2 * a);

        cout << "x1 = " << x1 << endl;
        cout << "x2 = " << x2 << endl;
    }

    else if (discriminant == 0) {

        x1 = -b / (2 * a);
        cout << "x1 =" << x1 << endl;
        cout << "x2 =" << x1 << endl;
    }

    else
    {
        realPart = -b / (2 * a);
        imaginaryPart = sqrt(-discriminant) / (2 * a);

        cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;
        cout << "x2 = " << realPart << "-" << imaginaryPart << "i" << endl;

    }
}
```

Figure 6: 2nd Order

Figure 11: 2nd Order

If determinant > 0,

$$root1 = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

If determinant = 0,

$$root1 = root2 = \frac{-b}{2a}$$

If determinant < 0,

$$root1 = \frac{-b}{2a} + i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b}{2a} - i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

Figure 12: Determinant of 2nd Order

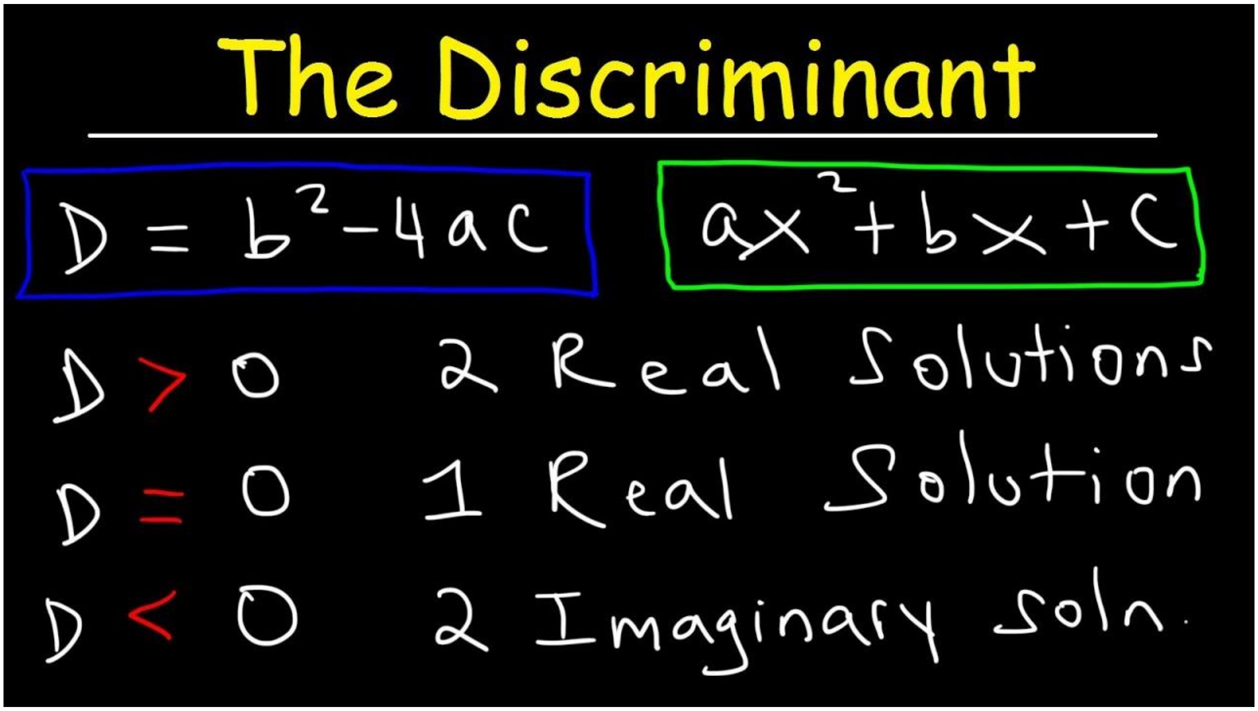


Figure 13: Discriminant

- To get the roots of the second degree polynomial equations, we use the coefficients stored in coeff[size] in the discriminant and determinant rules.
- We find the discriminant to know if the equation has 2 real roots or 2 complex roots or if there are 2 real roots equal to each other.
- If discriminant>0, then there are 2 real solutions. If discriminant=0 then there are 2 real roots equal to each other. If discriminant<0 then there are 2 imaginary solutions.
- Discriminant is used in the equation of the determinant if there are two real solutions or two imaginary solutions.
- Otherwise other rules of the determinant are used without discriminant.

- **3rd Order:**

```
void thrdorder(double a, double b, double c, double d)

{
    double discriminant, x1, x2, x3;

    double p = (b * b - 3 * a * c) / (9 * a * a);
    double q = (9 * a * b * c - 27 * a * a * d - 2 * b * b * b) / (54 * a * a * a);
    double sabet = b / (3 * a);

    // discriminant
    discriminant = p * p * p - q * q;

    if (discriminant > 0)
    {
        double angle = acos(q / (p * sqrt(p)));
        double r = 2 * sqrt(p);
        for (int n = 0; n < 3; n++)
            cout << r * cos((angle + 2 * n * PI) / 3.0) - sabet << "\n";
    }
    else
    {
        double angle1 = cbrt(q + sqrt(-discriminant));
        double angle2 = cbrt(q - sqrt(-discriminant));

        x1 = angle1 + angle2 - sabet;
        cout << angle1 + angle2 - sabet<< "\n";

        double realpart = -0.5 * (angle1 + angle2) - sabet;
        double imaginarypart = (angle1 - angle2) * sqrt(3) / 2;
        if (discriminant == 0)
        {
            cout << realpart << "\n";
            cout << realpart << "\n";
        }
        else
        {
            cout <<realpart << " + " << imaginarypart << " i \n";
            cout <<realpart << " - " << imaginarypart << " i \n";
        }
    }
}
```

Figure 14: 3rd Order

- Discriminant of the 3rd degree equation is calculated the same way as the 2nd degree.
- If discriminant>0 there is at least 1 real root. If discriminant=0 then at least 2 of the produced roots are equal.

Cubic Equation Formula:

$$x_1=(-\ term1 + r_{13}*\cos(q^3/3))$$
$$x_2=(-\ term1 + r_{13}*\cos(q^3+(2*\Pi)/3))$$
$$x_3=(-\ term1 + r_{13}*\cos(q^3+(4*\Pi)/3))$$

Where,

discriminant(Δ) = $q^3 + r^2$
term1 = $\sqrt{(3.0)}*((-t + s)/2)$
$r_{13} = 2 * \sqrt{(q)}$
$q = (3c - b^2)/9$
$r = -27d + b(9c-2b^2)$
$s = r + \sqrt{(\text{discriminant})}$
$t = r - \sqrt{(\text{discriminant})}$

Figure 15: Rule of 3rd degree

- **<u>4th Order</u>**

```
void frthorder(double a, double b, double c, double d, double e)

{
    complex<double> x1, x2, x3, x4, p1, p2, p3, p4, p5, p6;


    p1 = 2.0 * c * c * c - 9.0 * b * c * d + 27.0 * a * d * d + 27.0 * b * b * e - 72.0 * a * c * e;

    p2 = p1 + sqrt(complex<double>(-4.0 * (pow(c * c - 3.0 * b * d + 12.0 * a * e, 3.0)) + p1 * p1));

    p3 = (pow(c, 2.0) - 3.0 * b * d + 12.0 * a * e) / (3.0 * a * pow((p2 / 2.0), (1.0 / 3.0))) + (pow((p2 / 2.0), (1.0 / 3.0))) / (3.0 * a);

    p4 = sqrt(complex<double>((pow(b, 2.0)) / (4.0 * pow(a, 2)) - ((2.0 * c) / (3.0 * a)) + p3));

    p5 = pow(b, 2.0) / (2.0 * pow(a, 2.0)) - (4.0 * c) / (3.0 * a) - p3;

    p6 = (-(pow(b, 3.0) / pow(a, 3.0)) + (4.0 * b * c) / pow(a, 2.0) - ((8.0 * d) / a)) / (4.0 * p4);

    x1 = -(b / (4.0 * a)) - (p4 / 2.0) - (sqrt(complex<double>(p5 - p6))) / 2.0;

    x2 = -(b / (4.0 * a)) - (p4 / 2.0) + (sqrt(complex<double>(p5 - p6))) / 2.0;

    x3 = -(b / (4.0 * a)) + (p4 / 2.0) - (sqrt(complex<double>(p5 + p6))) / 2.0;

    x4 = -(b / (4.0 * a)) + (p4 / 2.0) + (sqrt(complex<double>(p5 + p6))) / 2.0;


    cout << "Root Format:  (real,imaginary) \n";
    cout << "Note:  if the root is real the answer will be shown as follows : (real,0) \n";
    cout << "x1: " << x1 << "\n";
    cout << "x2: " << x2 << "\n";
    cout << "x3: " << x3 << "\n";
    cout << "x4: " << x4 << "\n";
```

Figure 16: 4th Order

<u>We substitute the coefficients in the following equations:</u>

$$\mathbf{P1} = 2c^3 - 9bcd + 27ad^2 + 27b^2e - 72ace$$

$$\mathbf{P2} = p_1 + \sqrt{-4(c^2 - 3bd + 12ae)^3 + p_1^2}$$

$$\mathbf{P3} = \frac{c^2 - 3bd + 12ae}{3a\sqrt[3]{\frac{p_2}{2}}} + \frac{\sqrt[3]{\frac{p_2}{2}}}{3a}$$

Figure 17: 4th Rule (1)

$$p_4 = \sqrt{\frac{b^2}{4a^2} - \frac{2c}{3a} + p_3}$$

$$p_5 = \frac{b^2}{2a^2} - \frac{4c}{3a} - p_3$$

$$p_6 = \frac{-\frac{b^3}{a^3} + \frac{4bc}{a^2} - \frac{8d}{a}}{4p_4}$$

$$x = -\frac{b}{4a} - \frac{p_4}{2} - \frac{\sqrt{p_5 - p_6}}{2}$$

$$\text{or } x = -\frac{b}{4a} - \frac{p_4}{2} + \frac{\sqrt{p_5 - p_6}}{2}$$

$$\text{or } x = -\frac{b}{4a} + \frac{p_4}{2} - \frac{\sqrt{p_5 + p_6}}{2}$$

$$\text{or } x = -\frac{b}{4a} + \frac{p_4}{2} + \frac{\sqrt{p_5 + p_6}}{2}$$

$$\begin{aligned}
x_1 = &-\frac{b}{3a} \\
&-\frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
&-\frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
x_2 = &-\frac{b}{3a} \\
&+\frac{1 + i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
&+\frac{1 - i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
x_3 = &-\frac{b}{3a} \\
&+\frac{1 - i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]} \\
&+\frac{1 + i\sqrt{3}}{6a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4\left(b^2 - 3ac\right)^3}\right]}
\end{aligned}$$

Figure 18: 4th Order Rule (2)

- **5th Order**

```
case 5:
    complex<double> p, q, r, s, t;
    complex<double> seed = 0.4 + 0.9i;
    complex<double> a = coeff[0];
    complex<double> b = coeff[1];
    complex<double> c = coeff[2];
    complex<double> d = coeff[3];
    complex<double> e = coeff[4];
    complex<double> f = coeff[5];
    p = 1.0;
    q = p * seed;
    r = q * seed;
    s = r * seed;
    t = s * seed;
    //cout << p << " , " << q << " , " << r << " , " << s << " , " << t << "\n";

    for (int i = 0; i < 3000; ++i)
    {
        p = p - ((p * (p * (p * (p * (a*p + b) + c) + d) + e) + f)/a) / ((p - q) * (p - r) * (p - s) * (p - t));
        q = q - ((q * (q * (q * (q * (a * q + b) + c) + d) + e) + f) / a) / ((q - p) * (q - r) * (q - s) * (q - t));
        r = r - ((r * (r * (r * (r * (a * r + b) + c) + d) + e) + f) / a) / ((r - p) * (r - q) * (r - s) * (r - t));
        s = s - ((s * (s * (s * (s * (a * s + b) + c) + d) + e) + f) / a) / ((s - p) * (s - q) * (s - r) * (s - t));
        t = t - ((t * (t * (t * (t * (a * t + b) + c) + d) + e) + f) / a) / ((t - p) * (t - q) * (t - r) * (t - s));

    }
    cout << "x1: " << p << "\n";
    cout << "x2: " << q << "\n";
    cout << "x3: " << r << "\n";
    cout << "x4: " << s<< "\n";
    cout << "x5: " << t << "\n";

    break;

}
```

Figure 19: 5th Order

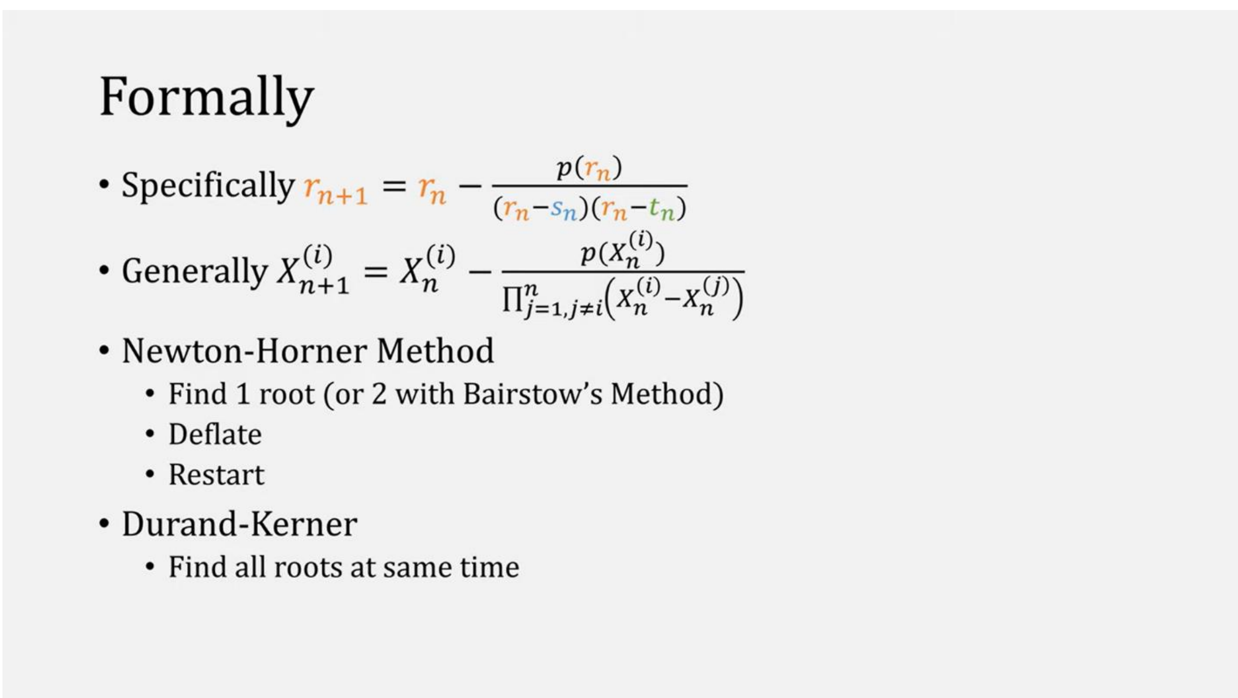


Figure 20: 5th Order Rule

Durand Kerner

$$r_{n+1} = r_n - \frac{p(r_n)}{(r_n - s_n)(r_n - t_n)}$$

$$s_{n+1} = s_n - \frac{p(s_n)}{(s_n - r_n)(s_n - t_n)}$$

$$t_{n+1} = t_n - \frac{p(t_n)}{(t_n - r_n)(t_n - s_n)}$$

- There is nothing special about choosing $0.4 + 0.9i$ except that it is neither a real number nor a root of unity.
- We use Durand Kerner method with all higher order polynomials.

2.0 Test Cases

2.1 Error 1

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
-5x^3 -x^2 +2x +
Error. Rewrite the equation as shown in the example
Press any key to continue . . .
```

2.2 Error 2

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
-5x^3 -x^ +2 +1
Error. Rewrite the equation as shown in the example
Press any key to continue . . .
```

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
5x^ -x^2 +2.5x +1
Error. Rewrite the equation as shown in the example
Press any key to continue . . .
```

2.3 Error 3

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
5x^3 -x2 +2.5x +1
Error. Rewrite the equation as shown in the example
Press any key to continue . . .
```

2.4 1st Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
x+3
degree:1
coefficient:1
coefficient:3
x1: -3
Press any key to continue . . .
```

2.5 1st Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
5x+10
degree:1
coefficient:5
coefficient:10
x1: -2
Press any key to continue . . .
```

2.6 2nd Order

```
C:\Windows\system32\cmd.exe
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
4x^2+2x+1
degree:2
coefficient:4
coefficient:2
coefficient:1
x1 : -0.25+0.433013i
x2 : -0.25-0.433013i
Press any key to continue . . .
```

2.7 2nd Order

```
C:\Windows\system32\cmd.exe
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
-65x^2 +2.7x +1
degree:2
coefficient:-65
coefficient:2.7
coefficient:1
x1 = -0.104992
x2 = 0.146531
Press any key to continue . . .
```

2.8 3rd order

```
C:\Windows\system32\cmd.exe
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
70x^3 +x^2 +52x +3
degree:3
coefficient:70
coefficient:1
coefficient:52
coefficient:3
x1: -0.0575
x2: 0.0216071 + 0.863061 i
x3: 0.0216071 - 0.863061 i
Press any key to continue . . .
```

2.9 3rd Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
-1.5x^3 +10x^2 +17.3x +4
degree:3
coefficient:-1.5
coefficient:10
coefficient:17.3
coefficient:4
8.12631
-1.18202
-0.27762
Press any key to continue . . .
```

2.10 4th Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
131x^4 +13.5x^3 -34x^2 +15x +1
degree:4
coefficient:131
coefficient:13.5
coefficient:-34
coefficient:15
coefficient:1
Root Format:  (real,imaginary)
Note:  if the root is real the answer will be shown as follows : (real,0)
x1: (-0.693029,0)
x2: (-0.0587616,0)
x3: (0.324369,-0.286765)
x4: (0.324369,0.286765)
Press any key to continue . . .
```

2.11 4th Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
11x^4 -2x^3 -6x^2 +22x +99.99
degree:4
coefficient:11
coefficient:-2
coefficient:-6
coefficient:22
coefficient:99.99
Root Format:  (real,imaginary)
Note:  if the root is real the answer will be shown as follows : (real,0)
x1: (-1.24532,1.00476)
x2: (-1.24532,-1.00476)
x3: (1.33623,1.32845)
x4: (1.33623,-1.32845)
Press any key to continue . . .
```

2.12 5th Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
66x^5 +x^4 +50x^3 +3x^2 -9x -3
degree:5
coefficient:66
coefficient:1
coefficient:50
coefficient:3
coefficient:-9
coefficient:-3
x1: (-0.278305,0.170203)
x2: (0.0379034,0.956973)
x3: (0.465652,0)
x4: (-0.278305,-0.170203)
x5: (0.0379034,-0.956973)
Press any key to continue . . .
```

2.13 5th Order

C:\Windows\system32\cmd.exe

```
Insert your polynomial in terms of x:
Example: 5x^3 -x^2 +2.5x +1
6.2x^5 +18.8x^4 +5x^3 -11x^2 +4x -5.3
degree:5
coefficient:6.2
coefficient:18.8
coefficient:5
coefficient:-11
coefficient:4
coefficient:-5.3
x1: (0.720131,0)
x2: (0.0664612,0.562143)
x3: (-1.67955,0)
x4: (-2.20576,0)
x5: (0.0664612,-0.562143)
Press any key to continue . . .
```

3.0 Appendix

```
#include <iostream>
#include <cstring>
#include <cmath>
#include<string>
#include<cstdlib>
#include<complex>

//This a calculator that finds the all the roots (real and complex ones) of max a 5th
degree quintic equation prepared by Habiba Yasser, Nadine Hisham , and Jumana Yasser
using namespace std;

void frthorder(double a, double b, double c, double d, double e);
void thrdorder(double a, double b, double c, double d);
void scndorder(double a, double b, double c);
void frstorder(double a, double b);



const double PI = 3.141592653589793;
int main()
{
	int x;
	int degree;
	bool error1, error2, error3, error4;
	cout << "Insert your polynomial in terms of x:\n";
	cout << "Example: 5x^3 -x^2 +2.5x +1\n";
	char str[1000];
	cin.getline(str, 1000);


	// error 1
	for (int i = 0; i < 1000; i++)
	{
		if (str[i] == NULL && (str[i - 1] == '+' || str[i - 1] == '-'))
		{
			error1 = 1;
			break;
		}

		else
		{
			error1 = 0;
		}
	}

	// error 2
	for (int i = 0; i < 1000; i++)
	{
		if (str[i] == '^')
		{
			if (!(str[++i] >= '0' && str[++i] <= '5'))
			{
				error2 = 1;
```

```
                        break;
                    }
                    else  error2 = 0;
            }
    }
    // error 3
    for (int i = 0; i < 1000; i++)
    {
            if (str[i] == 'x')
            {
                    if (str[++i] >= '0' && str[i] <= '9')
                    {
                            error3 = 1;
                            break;
                    }
                    else  error3 = 0;
            }
    }



    if (error1 == 1 || error2 == 1 || error3 == 1 )
    {
            cout << "Error. Rewrite the equation as shown in the example\n";
            system("pause");
    }

    for (int i = 0; i < 1000; i++)
    {

            if (str[i] == '^')
            {
                    i++;
                    x = i;
                    degree = str[x] - '0';
                    break;
            }
            else if ((str[i] == 'x' && (str[i + 1] == '+' || str[i + 1] == '-')) ||
(str[i] == 'x' && (str[i + 1] == ' ' && (str[i + 2] == '+' || str[i + 2] == '-'))))
            {
                    x = 1;
                    degree = x;
                    break;
            }
    }
    cout << "degree:" << degree << "\n";

    int size = degree + 1;
    float* coeff = new float[size];

    //coefficients
    char* token;
    double y;
    token = strtok(str, "+ ");
    y = atof(token);
    while (token != NULL)
    {
            for (int i = 0; i < size; i++)
```

```
		{
			y = atof(token);

			if (*token == '-' && *(token + 1) == 'x')
			{
				y = -1;
			}

			else if (*token == '-')
			{
				y = atof(token);

			}
			else if (*token == 'x')
			{
				y = 1;

			}
			cout << "coefficient:" << y << "\n";
			coeff[i] = y;
			token = strtok(NULL, "+ ");
		}
	}


	switch (degree)
	{
	case 1:
		frstorder(coeff[0], coeff[1]);
		break;
	case 2:
		scndorder(coeff[0], coeff[1], coeff[2]);
		break;
	case 3:
		thrdorder(coeff[0], coeff[1], coeff[2], coeff[3]);
		break;
	case 4:
		frthorder(coeff[0], coeff[1], coeff[2], coeff[3], coeff[4]);
		break;
	case 5:
		complex<double> p, q, r, s, t;
		complex<double> seed = 0.4 + 0.9i;
		complex<double> a = coeff[0];
		complex<double> b = coeff[1];
		complex<double> c = coeff[2];
		complex<double> d = coeff[3];
		complex<double> e = coeff[4];
		complex<double> f = coeff[5];
		p = 1.0;
		q = p * seed;
		r = q * seed;
		s = r * seed;
		t = s * seed;
		//cout << p << " , " << q << " , " << r << " , " << s << " , " << t << 
"\n";

		for (int i = 0; i < 3000; ++i)
```

```
            {
                p = p - ((p * (p * (p * (p * (a*p + b) + c) + d) + e) + f) / a) /
((p - q) * (p - r) * (p - s) * (p - t));
                q = q - ((q * (q * (q * (q * (a * q + b) + c) + d) + e) + f) / a) /
((q - p) * (q - r) * (q - s) * (q - t));
                r = r - ((r * (r * (r * (r * (a * r + b) + c) + d) + e) + f) / a) /
((r - p) * (r - q) * (r - s) * (r - t));
                s = s - ((s * (s * (s * (s * (a * s + b) + c) + d) + e) + f) / a) /
((s - p) * (s - q) * (s - r) * (s - t));
                t = t - ((t * (t * (t * (t * (a * t + b) + c) + d) + e) + f) / a) /
((t - p) * (t - q) * (t - r) * (t - s));

            }
            cout << "x1: " << p << "\n";
            cout << "x2: " << q << "\n";
            cout << "x3: " << r << "\n";
            cout << "x4: " << s << "\n";
            cout << "x5: " << t << "\n";

            break;

        }


        delete[] coeff;
        coeff = NULL;
}


void frstorder(double a, double b)
{
        double x1 = -1 * (b / a);
        cout << "x1: " << x1 << "\n";
}

void scndorder(double a, double b, double c)
{
        double discriminant, realroot, imaginaryroot, x1, x2;


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

        if (discriminant > 0)
        {
            x1 = (-b + sqrt(discriminant)) / (2 * a);
            x2 = (-b - sqrt(discriminant)) / (2 * a);

            cout << "x1 = " << x1 << "\n";
            cout << "x2 = " << x2 << "\n";
        }

        else if (discriminant == 0)
        {

            x1 = -b / (2 * a);
            cout << "x1 :" << x1 << "\n";
            cout << "x2 :" << x1 << "\n";
        }
```

```
	else
	{
		realroot = -b / (2 * a);
		imaginaryroot = sqrt(-discriminant) / (2 * a);

		cout << "x1 : " << realroot << "+" << imaginaryroot << "i\n";
		cout << "x2 : " << realroot << "-" << imaginaryroot << "i\n";

	}
}
void thrdorder(double a, double b, double c, double d)

{
	double discriminant, x1, x2, x3;

	double p = (b * b - 3 * a * c) / (9 * a * a);
	double q = (9 * a * b * c - 27 * a * a * d - 2 * b * b * b) / (54 * a * a * a);
	double sabet = b / (3 * a);

	// discriminant
	discriminant = p * p * p - q * q;

	if (discriminant > 0)
	{
		double angle = acos(q / (p * sqrt(p)));
		double r = 2 * sqrt(p);
		for (int n = 0; n < 3; n++)
			cout << r * cos((angle + 2 * n * PI) / 3.0) - sabet << "\n";
	}
	else
	{
		double angle1 = cbrt(q + sqrt(-discriminant));
		double angle2 = cbrt(q - sqrt(-discriminant));

		x1 = angle1 + angle2 - sabet;
		cout << angle1 + angle2 - sabet<< "\n";

		double realpart = -0.5 * (angle1 + angle2) - sabet;
		double imaginarypart = (angle1 - angle2) * sqrt(3) / 2;
		if (discriminant == 0)
		{
			cout << realpart << "\n";
			cout << realpart << "\n";
		}
		else
		{
			cout <<realpart << " + " << imaginarypart << " i \n";
			cout <<realpart << " - " << imaginarypart << " i \n";
		}
	}
}

void frthorder(double a, double b, double c, double d, double e)

{
	complex<double> x1, x2, x3, x4, p1, p2, p3, p4, p5, p6;
```

```
	p1 = 2.0 * c * c * c - 9.0 * b * c * d + 27.0 * a * d * d + 27.0 * b * b * e -
72.0 * a * c * e;

	p2 = p1 + sqrt(complex<double>(-4.0 * (pow(c * c - 3.0 * b * d + 12.0 * a * e,
3.0)) + p1 * p1));

	p3 = (pow(c, 2.0) - 3.0 * b * d + 12.0 * a * e) / (3.0 * a * pow((p2 / 2.0), (1.0
/ 3.0))) + (pow((p2 / 2.0), (1.0 / 3.0))) / (3.0 * a);

	p4 = sqrt(complex<double>((pow(b, 2.0)) / (4.0 * pow(a, 2)) - ((2.0 * c) / (3.0 *
a)) + p3));

	p5 = pow(b, 2.0) / (2.0 * pow(a, 2.0)) - (4.0 * c) / (3.0 * a) - p3;

	p6 = (-(pow(b, 3.0) / pow(a, 3.0)) + (4.0 * b * c) / pow(a, 2.0) - ((8.0 * d) /
a)) / (4.0 * p4);

	x1 = -(b / (4.0 * a)) - (p4 / 2.0) - (sqrt(complex<double>(p5 - p6))) / 2.0;

	x2 = -(b / (4.0 * a)) - (p4 / 2.0) + (sqrt(complex<double>(p5 - p6))) / 2.0;

	x3 = -(b / (4.0 * a)) + (p4 / 2.0) - (sqrt(complex<double>(p5 + p6))) / 2.0;

	x4 = -(b / (4.0 * a)) + (p4 / 2.0) + (sqrt(complex<double>(p5 + p6))) / 2.0;


	cout << "Root Format:  (real,imaginary) \n";
	cout << "Note:  if the root is real the answer will be shown as follows : (real,0)
\n";
	cout << "x1: " << x1 << "\n";
	cout << "x2: " << x2 << "\n";
	cout << "x3: " << x3 << "\n";
	cout << "x4: " << x4 << "\n";


	  }
```

Ain Shams University
Faculty of Engineering
Discipline Programs

Computer Programming Major Task Report

Computer Engineering and Software Systems (CESS)

Submitted to:

Dr. Mahmoud Khalil

Eng. Kyrollos

Submitted by:

Habiba Yasser 20P3072

Nadine Hisham 20P9880

Jumana Yasser 20P8421

Table of Contents

1.0 Milestone 2

1.1 Libraries Used:

```
#include <iostream>
#include<cmath>
#include<iomanip>
#include<stdlib.h>
using namespace std;
```

- Cmath library: for mathematical operations.
- Iomanip library: to set field width and decimal precision.
- Stdlib.h library: to use the rand and srand functions to generate random test cases.

1.2 User Input:

```
cout << "Enter number of rows and colums of Matrix A:\n";
cin >> n;

double** mata = new double*[n];
for (int i = 0; i < n; i++)
    mata[i] = new double[n];

cout << "Enter elements of Matrix A:\n";
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        cin >> mata[i][j];
    }
    cout << endl;
}
```

- User will be able enter any size for matrix A and B using dynamic allocation of 2D arrays, as long as the size of the column of matrix A is equal to the size of the row of Matrix B.

1.3 Determinant:

To calculate the determinant, we used the Gauss Jordan method.

First, we check if the value of any of the elements of the matrix's main diagonal is zero (in this case, the matrix is called a singular matrix). If so, the determinant is automatically determined to me zero.

Example:

$$\begin{bmatrix} 2 & 4 & 6 \\ 2 & 0 & 2 \\ 6 & 8 & 14 \end{bmatrix}$$

The Determinant is given by-

$$2(0–16)–4(28–12) + 6(16–0) = -2(16) + 2(16) = 0$$

If a matrix isn't singular, we get the determinant this way:

We will reduce this matrix into an upper triangular matrix using elementary operations.

We can interchange two rows of the matrix; we can multiply any row of the matrix with a scalar and we can add a multiple of a row to another for reducing the matrix into an upper triangular matrix.

Example of an upper triangular matrix:

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Since the matrix is equivalent to the upper triangular matrix, the determinant of both the matrices are equal. Therefore, the determinant of the matrix is equal to the product of the diagonal elements of the resultant upper triangular matrix.

This is true for any matrix.

Example:

Suppose $A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 5 \end{bmatrix}$.

We are going to reduce A into an upper triangular matrix as follows:

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 5 \end{bmatrix} \sim \begin{bmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 2 & 2 \end{bmatrix}; R_2 \to R_2 - R_1, R_3 \to R_3 - 3R_1$$

$$= \begin{bmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & 0 \end{bmatrix}; R_3 \to R_2 + R_3$$

$$= \begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}; R_2 \to -\frac{1}{2}R_2$$

Therefore, $|A| = 1 \cdot 1 \cdot 0 = 0$.

$$|A| = 1(5+11) - 3(5+3) + 1(11-3) = 16 - 3 \cdot 8 + 8 = 24 - 24 = 0.$$

Hence $|A| = |U|$.

Note: The determinant of a matrix is equal to the determinant of the corresponding upper triangular matrix. The determinant of an upper triangular matrix is the product of its diagonal elements.

1.4 Inverse of Matrix A

Similarly, we used the Gauss Jordan method to find matrix A's inverse.

First, we write the identity matrix next to the matrix we want to get the inverse of to get the "augmented matrix"

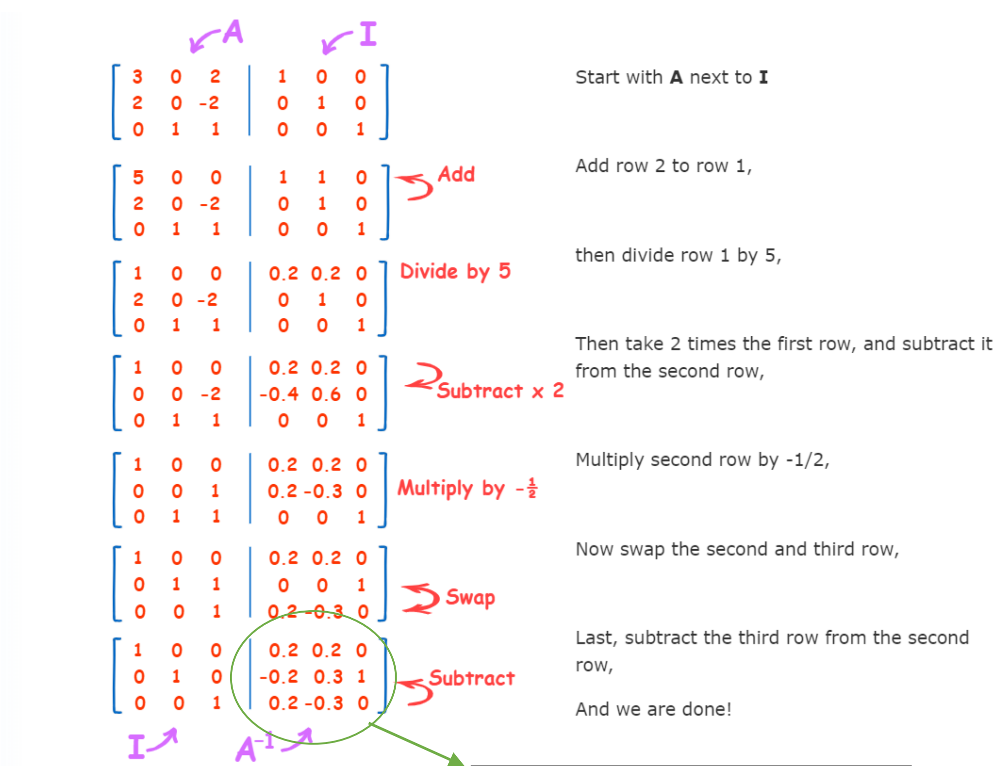
Now we do our best to turn the original matrix into an Identity Matrix. The goal is to make the matrix have **1**s on the diagonal and **0**s elsewhere (an Identity Matrix) ... and the right hand side comes along for the ride, with every operation being done on it as well. We do this till the original matrix turns to an identity matrix.

But we can only do these **"Elementary Row Operations"**:

- **swap** rows
- **multiply** or divide each element in a row by a constant
- replace a row by **adding** or subtracting a multiple of another row to it

And we must do it to the **whole row.**

Example:



this is the inverse of the matrix

2.0 Test Cases

- **Determinant**

```
//if you want the user to insert elements of the Matrix un-comment the part below and comment the un-commented part
/*cout << "Enter number of rows and colums of Matrix A:\n";
cin >> n;
double** mata = new double*[n];
for (int i = 0; i < n; i++)
{
    mata[i] = new double[n];
}


cout << "Enter elements of Matrix A:\n";
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        cin >> mata[i][j];
    }
    cout << endl;
}*/
cout << "Enter number of rows and colums of Matrix A:\n";
int n;
srand(time(0));
cin >> n;
double** mata = new double*[n];
for (int i = 0; i < n; i++)
    mata[i] = new double[n]; for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        mata[i][j] = rand() % 100;
    }
} cout << "n is " << n << endl;
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        cout << setw(10) << mata[i][j];
    }
    cout << endl;
}
```

rand() and srand() functions were used to generate random numbers for matrix A for most of the large numbers. If you want the users to insert the numbers themselves undo the comment above the srand() and rand() function and comment the srand() and rand().

2.1 N=13

```
Enter number of rows and colums of Matrix A:
13
n is 13
       82       11       39       99       97        7       82       51       82        5       77       17       30
       12       33       71        0       65       49       54       86       26       37       65        6       86
       89       33       31       74        2       31       13       32       81       33       10       69       46
       50       71       41       86       60       39        1        0        5       98       88        8       90
       68       35       97       57       30       53       49       61       25       43       35       90       71
       35       64       92       39       43       13       46       44       12       76       81       80       84
       32       57       29       89       22       58       32       35       22       11       66        7       27
        9       64       41       89        9       30       19        7       79       94       95       44       51
       49       28       70        5       82       95       43       53       74       75       43       22       56
       94       66       90       58       54       64       66       96       20       89       80       79       78
        0       14       48       31       55       77       18       87       65       26        6       76       31
       12       98       84       35       86       79       15       73       82       26       16       78       74
       97        7       83       18       59       50       76       58       10       53       53       25       64
determinant is: -4.12564e+22

C:\Users\Jumana\Desktop\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe (process 23368) exited with code 0.
Press any key to close this window . . .
```

Result of determinant calculation

Show solution Recalculate

Result:

Δ = -4.1256436458942418273e+22

2.2 N=25

```
Enter number of rows and colums of Matrix A:
25
n is 25
      52      85       8      76      36       0      22      40      87      19      46      60      32      18       8      89      67      89      28       3      19
     57      84      59      83
      19      28      35      26       3      54      66      34      99       8      26      26      15      64      22      20      16      75      72      51      62
     69       3      89      43
      33      36      43      77      48      94      84      61      66       3       4      92      83       9      54       2      87      37      87      90      33
     38      20      37       9
      33      18      23      84      56      13      74      16       5      79      56       3      89       2      31      25      49      75      14      14      51
     88      99      56      59
      56      15      45       8      21      46      51      97       6      48      33      93      55      95      61      51      90      54      59      46      91
     84      96      97      91
      33      74      72      23      14       4      26      95      55      69      10      69      59      78      92      84      22      87      93      79      74
     93      93       8       2
      77      92      25      66       8      83      40      53      48      45      65      23      22      46       2      74       0      58      94      11      37
     92      18      65      43
      25      97      30      44      64      97      70      52      52      23      71      33      88      38      77      76      38       1      89      15      73
      7       8      56      87
      43      91      89      51      78      69      52      88       5       5      49       7      53      92      93      77      88      48      46       3       1
     18      89      13      35
      30      23       5      40      97      78      98       8      43      97       1      27      93      27      42      79      89      57      42       4       9
     39      15      74      33
      36      33      45      52      11      28      84      48      29      53      60      48      64      50      33      95      34      48      64      35      35
     75      79      31      19
      69      11      33      84      39      80      82      58      88      18      95      87      70      82      32      89       6      94       7      74      99
     10      95      44      14
      87      93      33      72      81      63      96      33      32       7      80      52      90       3      81      98       0      30      89      51      46
     25      69      63      36
      56      88      61      79      78      82      30      59      40      72      44      41      36      24      21      60      85      37      86      36      23
     76      35       4      83
      44      16      40      25      38      68       9      43      72      49       0      66       9      41      40      80      58      96      68      92      74
      1      20      17      20
      65      66      91      60      78      40      40      69       0      81      10      11      78      37      94      61       2      41       1      24      34
     94      88      52      30
      16      53      98      18      66      28      19      78      44      75      21      55      86      98       8      67      63      86      70      27      76
     64      37      89      94
      76      72      41      77      14      12      40      55      32      17      95      88      74      18      72      83      63      18      38      16      53
     59      14      68      71
      43      94      14      80      44      37      27      38      32       6      17      68      46      54      39      61      63       0      40      87      53
     91       9      63      23
      12      40      41      13      24      86       9      31      68      71       7      16       6      93      57      29      54      70       7      25      86
      6      92      40      14
      76      62      87       8      67      66      14      35      65      29      52      17      63      94      42      87      99      77      10      80      94
     51      26      84      28
      51      42      18      70      97      90      60      47       5       4      54      50      21       4      18      62      94      45       2      25      41
     55      62      19      62
      41      40      79      61      28       9      12      24      13      26      33      63      68       6      36      86      17      58      32      28      56
     42       5      76      87
      18      46      17      62      17      13      33      46      29      49      47      47      83       8      21      77      44      46      71      64       1
     97      78      94      74
      71       7      31      37      97      91      67      33      17      75      93      25      62      68      92      47      18      89      66      44      73
     76      34       6      37
determinant is: -4.89758e+49
```

2.3 N=17

```
choose which operation you'd like to perform:
1) Find the determinant
2) Find the Inverse

1
Enter number of rows and colums of Matrix A:
17
n is 17
      32      89      11      11      85      82      87      96      35      41       3      27      52      14      48      76      93
      49      74      97      49       4      56      69      48      85       7       7      75      57      76      65      43      43
      83      45      79      65      98      75       3      28      87      18      73      58      69      27      84      41      68
      71       7      41      98      73      90      94       4      80      81      39      74      74      94      94       0      61
      43      25      62      73      52      72      25      35      57      89      46      31      96      68      14      52      94
      23      18       0      54      81      61      82      93      88      20       8      80      40      45      36      75      50
       8      90      96      59      17      39      32      74      98      81      80      92      16      19      26      57      24
      18      96      73      76      67      41      72      61       3      47       7      75      53      13      96      91      95
      30      63      69       8      62      18      42      92      40      63      51      64      19      16      47      59      52
      55      25       1      19      62      10      60      90       1      90      44      52      58      20      79      99      46
      23      29       7      61      81      12      27      50      39      23      57      16      43      12      74      31      25
      96      25       4      98       6      85      86      92      86      59      66      30      99      93      82      35      34
      63      79       8      50      79      40       3      45      52      69      25      96      63      57       0       2      82
      72      14      29      24      12      96      98      32      54      46      38      89      76      30      36      22      88
      43      40      75      43      80      89      52      26      65      28      33      41      77      89      18      83       5
      13      47      78      56      44      66      23      51      40      11      43      11      22       9      67      29      49
      51      21      73      60      38      45      54      78      85      79      65      68      89      89      44       3      46
determinant is: 2.61194e+32

C:\Users\Jumana\Desktop\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe (process 24216) exited with code 0.
Press any key to close this window . . .
```

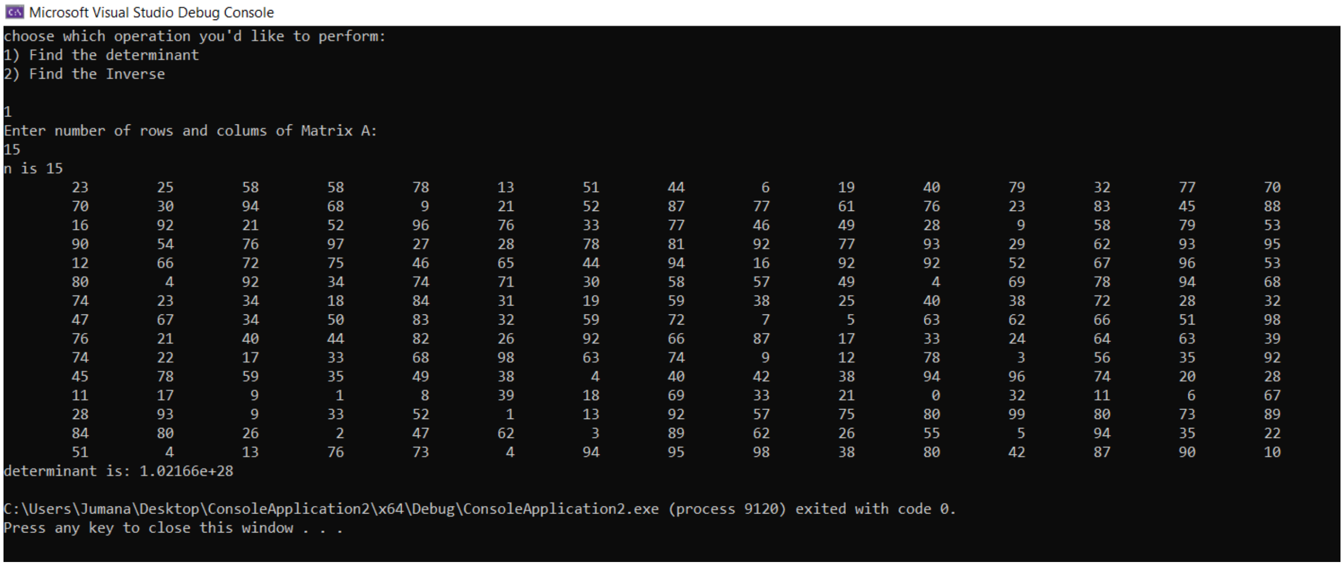
Result of determinant calculation

Show solution   Recalculate

Result:

Δ = 2.6119437570250605577e+32

2.4 N=15



```
Microsoft Visual Studio Debug Console
choose which operation you'd like to perform:
1) Find the determinant
2) Find the Inverse

1
Enter number of rows and colums of Matrix A:
15
n is 15
        23      25      58      58      78      13      51      44       6      19      40      79      32      77      70
        70      30      94      68       9      21      52      87      77      61      76      23      83      45      88
        16      92      21      52      96      76      33      77      46      49      28       9      58      79      53
        90      54      76      97      27      28      78      81      92      77      93      29      62      93      95
        12      66      72      75      46      65      44      94      16      92      92      52      67      96      53
        80       4      92      34      74      71      30      58      57      49       4      69      78      94      68
        74      23      34      18      84      31      19      59      38      25      40      38      72      28      32
        47      67      34      50      83      32      59      72       7       5      63      62      66      51      98
        76      21      40      44      82      26      92      66      87      17      33      24      64      63      39
        74      22      17      33      68      98      63      74       9      12      78       3      56      35      92
        45      78      59      35      49      38       4      40      42      38      94      96      74      20      28
        11      17       9       1       8      39      18      69      33      21       0      32      11       6      67
        28      93       9      33      52       1      13      92      57      75      80      99      80      73      89
        84      80      26       2      47      62       3      89      62      26      55       5      94      35      22
        51       4      13      76      73       4      94      95      98      38      80      42      87      90      10
determinant is: 1.02166e+28

C:\Users\Jumana\Desktop\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe (process 9120) exited with code 0.
Press any key to close this window . . .
```

2.5 N=11

Microsoft Visual Studio Debug Console

```
choose which operation you'd like to perform:
1) Find the determinant
2) Find the Inverse

1
Enter number of rows and colums of Matrix A:
11
n is 11
        30      87      33      89      43      54      96       6       7      76      67
        10      39      54      73      11      94      25      90      80      53      10
        74      61       1      51      93      12      55      77      29      24      49
        40      49      53       4      50      36      88      97      92      19      69
        32      85      28      18      61      60      68      86      73      65      53
        77       9      55      31       9      44      56       3       2      95      60
        86      14      93      30      61      21      53      13      49      91      71
        38      77      49      24      75      69      29      37       4      45      72
        98      27      46      18      64      13      73      25      45      43      22
        65      61      30      46       4      41       3      20      12       4      21
        28      45      90      60      66      45      52      58      46      81      51
determinant is: -4.84332e+20

C:\Users\Jumana\Desktop\ConsoleApplication2\x64\Debug\ConsoleApplication2.exe (process 32668) exited with code 0.
Press any key to close this window . . .
```

Result of determinant calculation

Show solution Recalculate

Result:

Δ = -484332176327937193630

- **<u>Inverse</u>**

2.6

C:\Windows\system32\cmd.exe

```
choose which operation you'd like to perform:
1) Find the determinant
2) Find the Inverse

2
Enter number of rows and colums of Matrix A:
2
Enter elements of Matrix A:
4
3


7
6

Please enter the number of rows of Matrix B (should be the same number as Matrix A):
2
Please enter elements of Matrix B:
1
2

Inverse Matrix:
                   2                   -1
            -2.33333             1.33333
Press any key to continue . . .
```

Result of matrix inversion

Show solution | Recalculate | Continue calculation

Result:

	B_1	B_2
1	2	-1
2	-2.3333333333333333333	1.3333333333333333333

2.7

C:\Windows\system32\cmd.exe

```
2) Find the Inverse

2
Enter number of rows and colums of Matrix A:
3
Enter elements of Matrix A:
4
1
5

3
7
9

4
2
7

Please enter the number of rows of Matrix B (should be the same number as Matrix A):
3
Please enter elements of Matrix B:
3
6
5

Inverse Matrix:
            1.06897            0.103448           -0.896552
           0.517241            0.275862           -0.724138
          -0.758621           -0.137931            0.862069
Press any key to continue . . .
```

	B_1	B_2	B_3
1	1.0689655172413793103	0.10344827586206896551	-0.89655172413793103448
2	0.51724137931034482758	0.27586206896551724136	-0.72413793103448275862
3	-0.75862068965517241379	-0.13793103448275862068	0.86206896551724137931

2.8

C:\Windows\system32\cmd.exe

```
choose which operation you'd like to perform:
1) Find the determinant
2) Find the Inverse

2
Enter number of rows and colums of Matrix A:
2
Enter elements of Matrix A:
8
3

5
4

Please enter the number of rows of Matrix B (should be the same number as Matrix A):
2
Please enter elements of Matrix B:
4
5

Inverse Matrix:
           0.235294           -0.176471
          -0.294118            0.470588
```

	B_1	B_2
1	0.23529411764705882352	-0.17647058823529411764
2	-0.29411764705882352941	0.47058823529411764705

3.0 Appendix

```
#include <iostream>
#include<cmath>
#include<iomanip>
#include<stdlib.h>
using namespace std;//double inverse(double mata[s][s], int n)
double det(double** matrix, int n)
{
double det = 1;
double d;
for (int i = 0; i < n; i++)
{
if (matrix[i][i] == 0)
{
cout << "Mathematical Error!" << endl;;
system("pause");
}
for (int j = i + 1; j < n; j++)
{
d = matrix[j][i] / matrix[i][i]; for (int k = 0; k < n; k++)
{
matrix[j][k] = matrix[j][k] - d * matrix[i][k];
}
}
}
for (int i = 0; i < n; i++)
{
det = det * matrix[i][i];
if (det == 0)
system("pause");
} return det;
}
void inversematprint(double** mata, int n, int m)
{
for (int i = 0; i < n; i++) {
for (int j = n; j < m; j++) {
cout << setw(20) << mata[i][j];
}
cout << endl;
}
return;
}void inversemata(double** mata, int n)
```

```
{
double r; for (int i = 0; i < n; i++)
{
for (int j = 0; j < 2 * n; j++)
{
if (j == (i + n))
mata[i][j] = 1;
}
} for (int i = n - 1; i > 0; i--)
{
if (mata[i - 1][0] < mata[i][0]) {
double* r = mata[i];
mata[i] = mata[i - 1];
mata[i - 1] = r;
}
} for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (j != i)
{
r = mata[j][i] / mata[i][i];
for (int k = 0; k < 2 * n; k++)
{
mata[j][k] -= mata[i][k] * r;
}
}
}
}
double** inv = new double* [n];
for (int i = 0; i < n; i++)
inv[i] = new double[n]; for (int i = 0; i < n; i++)
{
r = mata[i][i];
for (int j = 0; j < 2 * n; j++)
{
mata[i][j] = mata[i][j] / r;
}
} cout << "Inverse Matrix:\n";
inversematprint(mata, n, 2 * n); return;
}
int main()
{
double ratio;
int choice;
int n, b;
```

```
cout << "choose which operation you'd like to perform: \n"
<< "1) Find the determinant \n"
<< "2) Find the Inverse \n" << endl;
cin >> choice;
switch (choice)
{
case 1:
{
cout << "Enter number of rows and colums of Matrix A:\n";
int n;
srand(time(0));
cin >> n;
double** mata = new double* [n];
for (int i = 0; i < n; i++)
mata[i] = new double[n]; for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
mata[i][j] = rand() % 100;
}
} cout << "n is " << n << endl;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
cout << setw(10) << mata[i][j];
}
cout << endl;
}
cout << "determinant is: " << det(mata, n) << endl;
for (int i = 0; i < n; i++)
{
delete[] mata[i];
}
delete[] mata;
break;
}
case 2:
{
cout << "Enter number of rows and colums of Matrix A:\n";
cin >> n; double** mata = new double* [n];
for (int i = 0; i < n; i++)
mata[i] = new double[n]; cout << "Enter elements of Matrix A:\n";
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
```

```
{
cin >> mata[i][j];
}
cout << endl;
} cout << "Please enter the number of rows of Matrix B (should be the same number as Matrix A): "
<< endl;
cin >> b; if (n != b)
{
cout << "Error\n";
system("pause");
} cout << "Please enter elements of Matrix B:" << endl; double** matb = new double* [b];
for (int i = 0; i < b; i++)
matb[i] = new double[b]; for (int i = 0; i < b; i++)
{
cin >> matb[i][1];
}
cout << endl; inversemata(mata, n);
for (int i = 0; i < n; i++)
{
delete[] mata[i];
}
delete[] mata;
break;
}
}
}
```