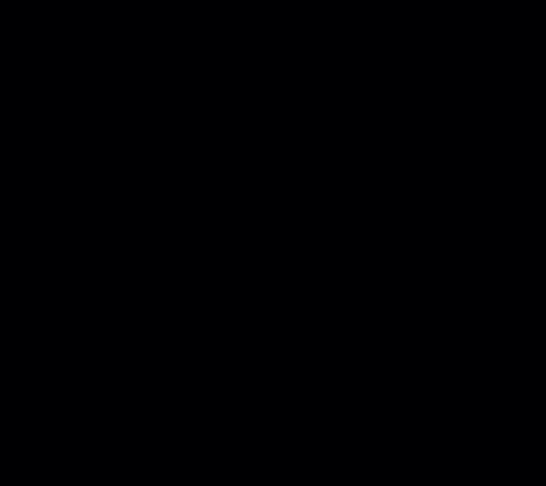


Eigen Value Calculator



→Eigenvalues are **the special set of scalar values that is associated with the set of linear equations most probably in the matrix equations**. The eigenvectors are also termed as characteristic roots. It is a non-zero vector that can be changed at most by its scalar factor after the application of linear transformations.

Mini Project

//Mini Project on :--→

Eigen Value Calculator

→Submitted to 'Mrs. Layanvya'

→Submitted by Arshaan Iqbal

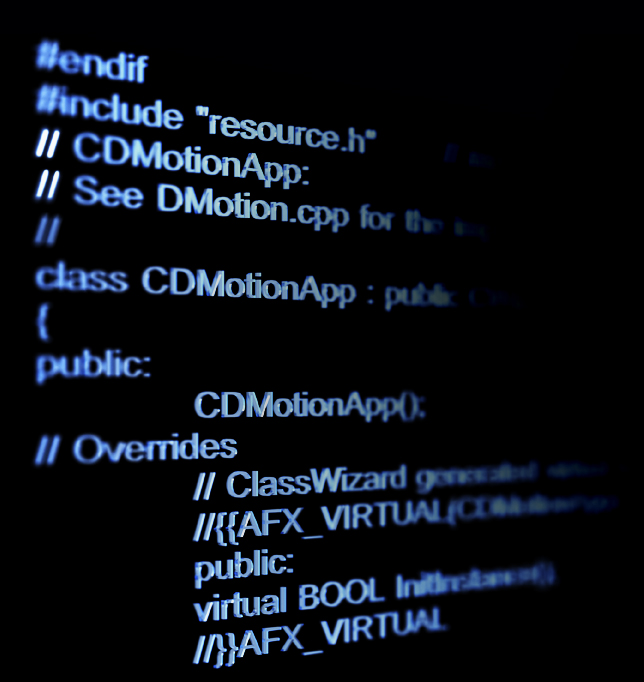
RA2111003011319

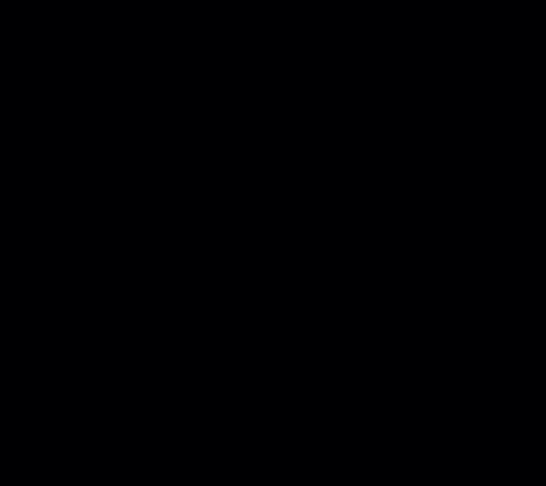
(H2)

Medikondur Koustubh RA2111003011303

(H2)

        }





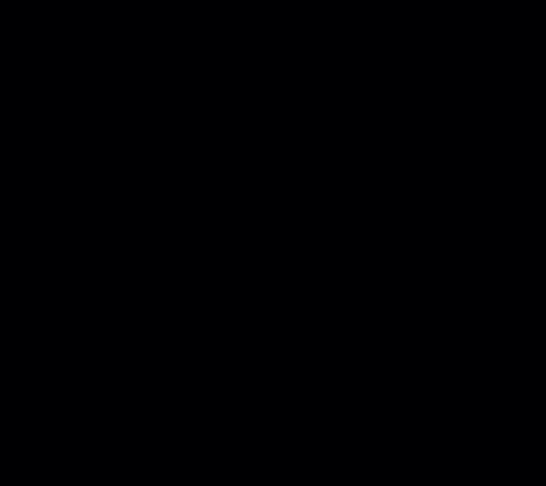
Project Statement → In this article we are going to develop an **Algorithm for Power Method** for computing largest or dominant Eigen value and corresponding Eigen vector.

Let A be the square matrix of order n i.e. An x n. Then Power Method starts with one initial approximation to Eigen vector corresponding to largest Eigen value of size n x 1. Let this initial approximation be Xn x 1.

After initial assumption, we calculate **AX** i.e. product of matrix A and X. From the product of AX we divide each element by largest element (by magnitude) and express them as λ1X1. Obtained value of λ1 and X1 are next better approximated value of largest Eigen value and corresponding Eigen vector.

Similarly, for the next step, we multiply A by X1. From the product of AX1 we divide each element by largest element (by magnitude) and express them as λ2X2. Obtained value of λ2 and X2 are next better approximated value of largest Eigen value and corresponding Eigen vector.

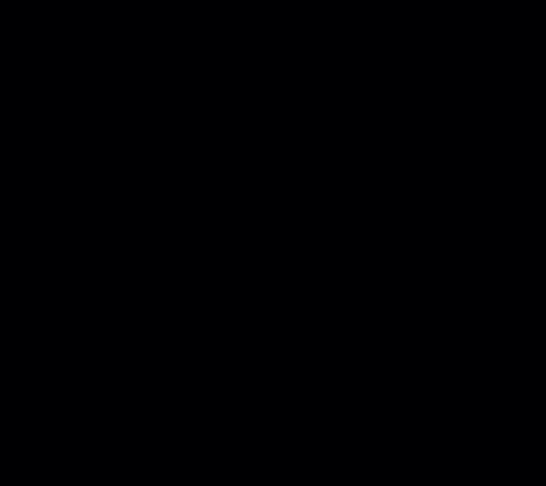
And then we will repeat this process until largest or dominant Eigen value and corresponding Eigen vector are obtained within desired accuracy



# Project Title → Power Method Using C Programming for Finding Dominant Eigen Value and Eigen Vector

g





## Algorithm for Power Method→

1. Start

2. Read Order of Matrix (n) and Tolerable Error (e)

3. Read Matrix A of Size n x n

4. Read Initial Guess Vector X of Size n x 1

5. Initialize: Lambda\_Old = 1

6. Multiply: X\_NEW = A \* X

7. Replace X by X\_NEW

8. Find Largest Element (Lamda\_New) by Magnitude from X\_NEW

9. Normalize or Divide X by Lamda\_New

10. Display Lamda\_New and X

11. If |Lambda\_Old - Lamda\_New| > e then

set Lambda\_Old = Lamda\_New and goto

step (6) otherwise goto step (12)

12. Stop

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

        {

            printf("Mini Project on Eigen Value Calculator\n Submitted to 'Mrs. Layanvya'\n Submitted by Arshaan Iqbal       (RA2111003011319)\n              Medikondur Koustubh (RA2111003011303)\n              (H2)\n");

        }

            #define SIZE 10

            float a[SIZE][SIZE], x[SIZE],x\_new[SIZE];

            float temp, lambda\_new, lambda\_old, error;

            int i,j,n, step=1;

*//Ask the user to enter Order of Matrix*

            printf("Enter Order of Matrix: ");

            scanf("%d", &n);

*//Ask the user to enter Tolerable Error*

            printf("Enter Tolerable Error: ");

            scanf("%f", &error);

*//Reading the Matrix*

            printf("Enter Coefficient of Matrix:\n");

            for(i=1;i<=n;i++)

            {

                for(j=1;j<=n;j++)

                    {

                        printf("a[%d][%d]=",i,j);

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

                    }

            }

*// Ask the user to enter Initial guess vector*

                printf("Enter Initial Guess Vector:\n");

                for(i=1;i<=n;i++)

                {

                    printf("x[%d]=",i);

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

                }

*// Initializing Lambda\_Old*

                lambda\_old = 1;

*//Multiplying*

                up:

                for(i=1;i<=n;i++)

                {

                    temp = 0.0;

                    for(j=1;j<=n;j++)

                    {

                        temp = temp + a[i][j]\*x[j];

                    }

                        x\_new[i] = temp;

                }

*//Replacing*

                for(i=1;i<=n;i++)

                {

                    x[i] = x\_new[i];

                }

*// Finding largest intg*

                lambda\_new = fabs(x[1]);

                for(i=2;i<=n;i++)

                {

                    if(fabs(x[i])>lambda\_new)

                    {

                        lambda\_new = fabs(x[i]);

                    }

                }

*// Normalising*

                for(i=1;i<=n;i++)

                {

                    x[i] = x[i]/lambda\_new;

                }

*// Final step → Display the output*

                printf("\n\nSTEP-%d:\n", step);

                printf("Eigen Value = %f\n", lambda\_new);

                printf("Eigen Vector:\n");

                for(i=1;i<=n;i++)

                {

                    printf("%f\t", x[i]);

                }

*//Checking Accuracy*

                if(fabs(lambda\_new-lambda\_old)>error)

                {

                    lambda\_old=lambda\_new;

                    step++;

*//using goto statement → (up)*

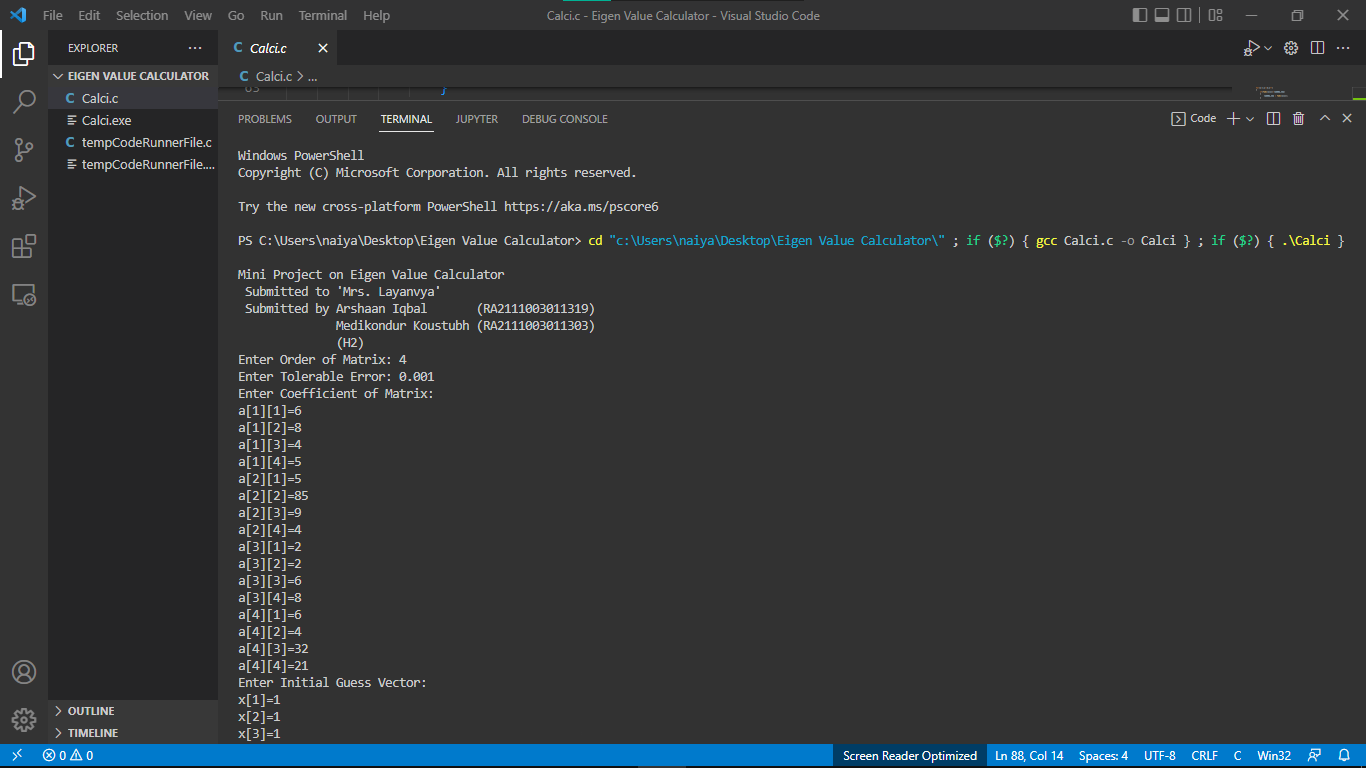
                    goto up;

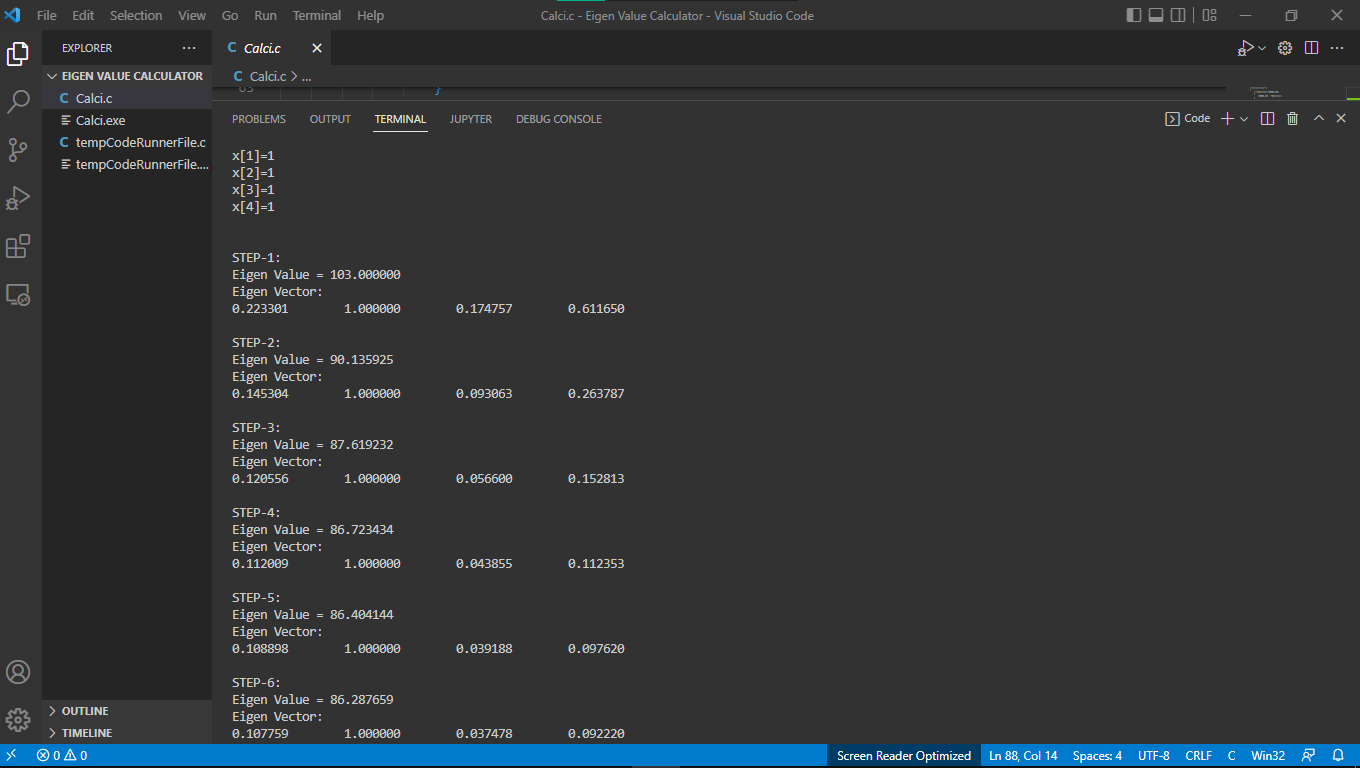
                }

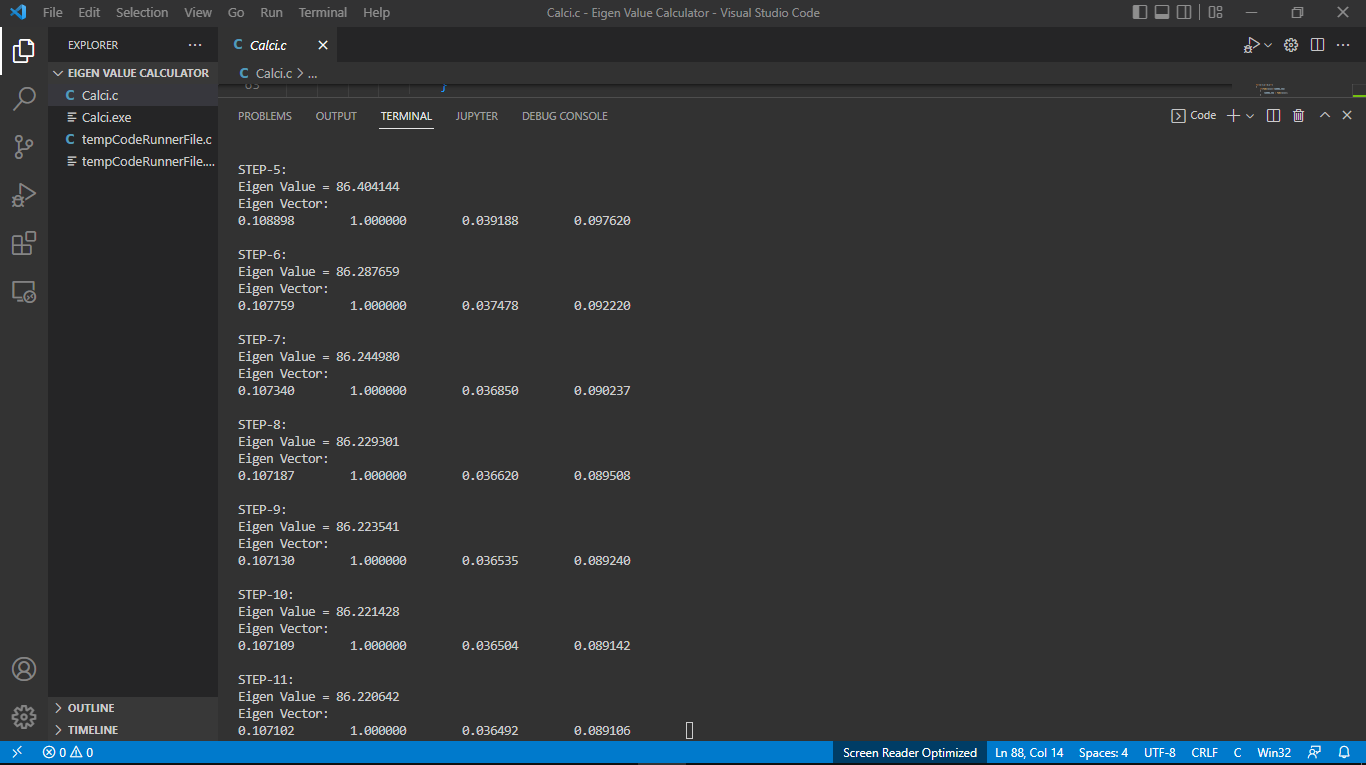
                getch();

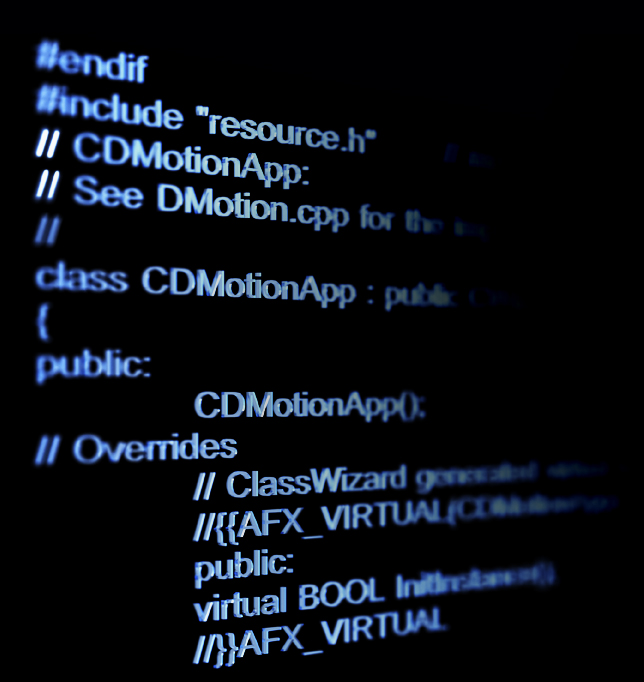
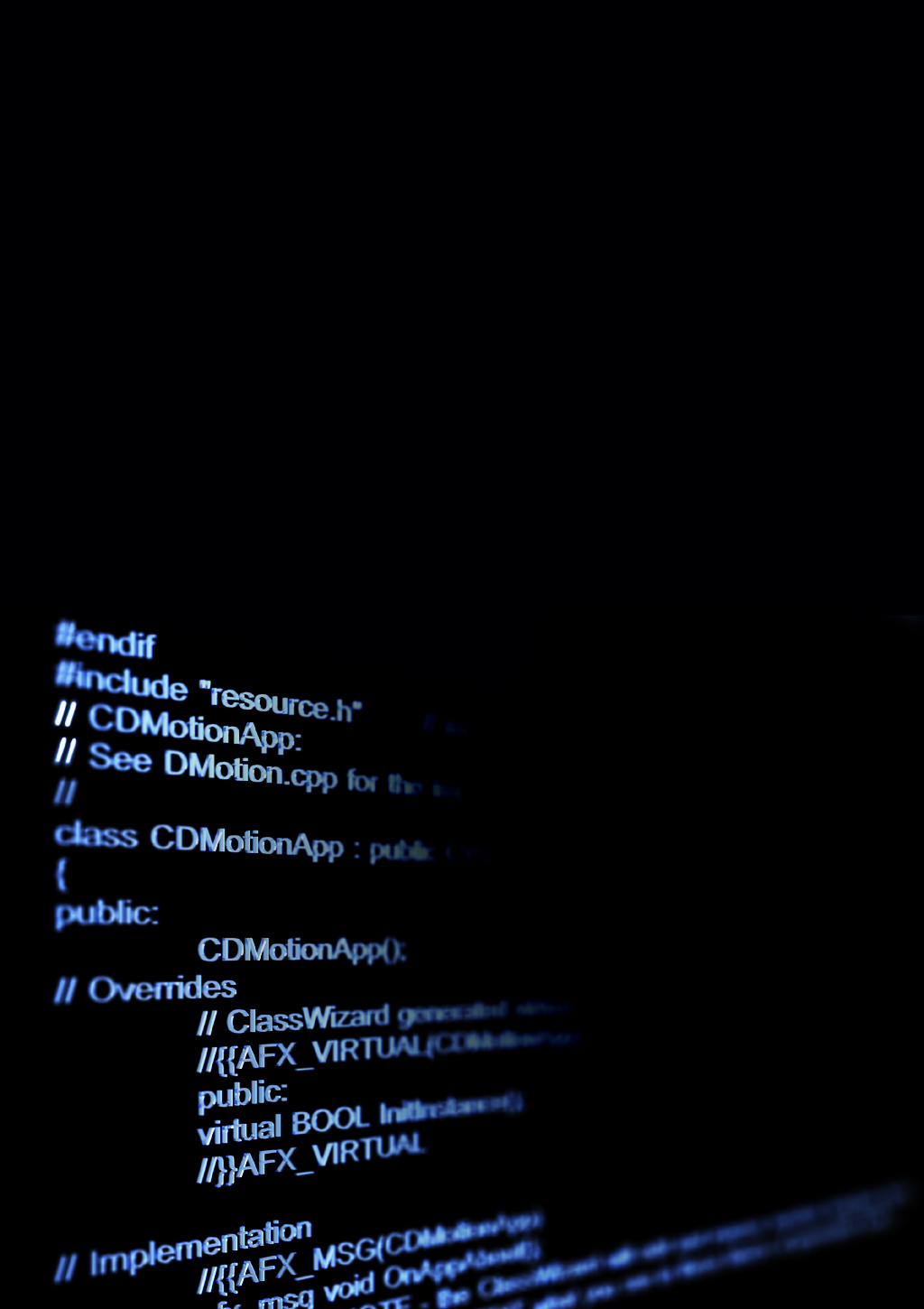
                return(0);

}









Result → Successfully developed an **Algorithm for Power Method** for computing largest or dominant Eigen value and corresponding Eigen vector.