Tutorial 3:

Code:

#include <bits/stdc++.h>

using namespace std;

double RMS(const vector<double> &errors) {

    double sum = 0;

    for (const double &error : errors) {

        sum += error \* error;

    }

    return sqrt(sum / errors.size());

}

vector<double> jacobi(int n, const vector<vector<double>> &A, const vector<double> &B, vector<double> guess, int &n1) {

    vector<double> ans = guess;

    vector<double> errors(n);

    do {

        // cout<<RMS(errors)<<endl;

        n1++;

        guess = ans;

        for (int i = 0; i < n; i++) {

            double sum = 0;

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

                if (i != j) {

                    sum += A[i][j] \* guess[j];

                }

            }

            ans[i] = (B[i] - sum) / A[i][i];

        }

        for (int i = 0; i < n; i++) {

            errors[i] = abs(guess[i] - ans[i]);

        }

    } while (RMS(errors) > 1e-10);

    return ans;

}

vector<double> gauss\_seidel\_wsor(int n, const vector<vector<double>> &A, const vector<double> &B, vector<double> guess, int &n2) {

    vector<double> ans = guess;

    vector<double> errors(n);

    do {

        // cout<<RMS(errors)<<endl;

        n2++;

        for (int i = 0; i < n; i++) {

            double x\_old = ans[i];

            double sum = 0;

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

                if (i != j) {

                    sum += A[i][j] \* ans[j];

                }

            }

            ans[i] = (B[i] - sum) / A[i][i];

            errors[i] = abs(ans[i] - x\_old);

        }

    } while (RMS(errors) > 1e-10);

    return ans;

}

double SOR(double lambda, double x\_old, double x\_new) {

    return (lambda \* x\_new) + ((1.0 - lambda) \* x\_old);

}

vector<double> gauss\_seidel\_sor(int n, const vector<vector<double>> &A, const vector<double> &B, vector<double> guess, double l, int &n3) {

    vector<double> ans = guess;

    vector<double> errors(n);

    do {

        // cout<<RMS(errors)<<endl;

        n3++;

        for (int i = 0; i < n; i++) {

            double x\_old = ans[i];

            double sum = 0;

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

                if (i != j) {

                    sum += A[i][j] \* ans[j];

                }

            }

            ans[i] = (B[i] - sum) / A[i][i];

            ans[i] = SOR(l, x\_old, ans[i]);

            errors[i] = abs(ans[i] - x\_old);

        }

    } while (RMS(errors) > 1e-10);

    return ans;

}

int main() {

    cout << "Enter the number of equations ";

    fstream fk;

    fk.open("input.txt",ios::in);

    int n;

    fk >> n;

    vector<vector<double>> A(n, vector<double>(n));

    vector<double> B(n);

    vector<double> iguess(n);

    cout << "Enter elements for matrix A " << endl;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            fk >> A[i][j];

        }

    }

    cout << "Enter elements for matrix B " << endl;

    for (int i = 0; i < n; i++) {

        fk >> B[i];

    }

    cout << "Enter values for initial guess " << endl;

    for (int i = 0; i < n; i++) {

        fk >> iguess[i];

    }

    int n1 = 0, n2 = 0, n3 = 0;

    vector<double> X1 = jacobi(n, A, B, iguess, n1);

    vector<double> X2 = gauss\_seidel\_wsor(n, A, B, iguess, n2);

    cout << "Enter the value of lambda" << endl;

    double l;

    fk >> l;

    fk.close();

    fstream ffk;

    ffk.open("output.txt",ios::out);

    vector<double> X3 = gauss\_seidel\_sor(n, A, B, iguess, l, n3);

    ffk << "The solution matrix obtained via Jacobi's method is " << endl;

    for (const double &x : X1) {

        ffk << x << endl;

    }

    ffk << "Number of iterations taken for Jacobi " << n1 << endl;

    cout<<endl;

    ffk << "The solution matrix obtained via Gauss-Seidel method without SOR is " << endl;

    for (const double &x : X2) {

        ffk << x << endl;

    }

    ffk << "Number of iterations taken for Gauss-Seidel without SOR " << n2 << endl;

    cout<<endl;

    ffk << "The solution matrix obtained via Gauss-Seidel method using SOR is " << endl;

    for (const double &x : X3) {

        ffk << x << endl;

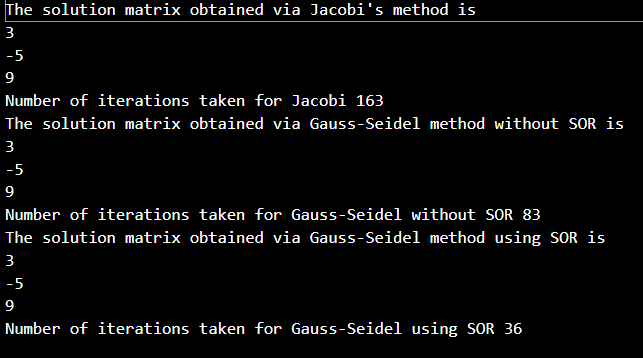
    }

    ffk << "Number of iterations taken for Gauss-Seidel using SOR " << n3 << endl;

    return 0;

}

Output:



Graphs:

Inference:

Gauss Seidel’s Method is effective because of using sor (Successive Over-Relaxation) than without using sor. It takes 83 iterations to solve the equations without using sor and 36 with using sor. This happens because the given system of equations is diagonally dominant, i.e., convergence is guaranteed. So, over- relaxation pushes the guessed values closer to the truth and thus, a highly accurate guess is obtained more quickly.