**Aishwary Sharma**

**2022CH11437**

**Jacobi Method:**

#include<bits/stdc++.h>

using namespace std;

float Error(vector<float>&v1, vector<float>&v2){

float error = 0;

for(int i = 0; i < v1.size(); i++){

error += pow((v1[i]-v2[i]),2);

}

return sqrt(error);

}

void Jacobimethod(){

ifstream in("Input1.txt");

ofstream out("Output1.txt");

int n;

in >> n ;

vector <vector <float> > A (n , vector<float>(n,0));

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

out << "Coeffecients of equation " << i+1 << " - " << endl;

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

in >> A[i][j];

out << A[i][j] << endl;

}

}

vector<float> B(n,0);

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

in >> B[i];

}

vector<float> X0 (n);

out << "Initial guess - ";

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

in >> X0[i];

out << X0[i]<< " " ;

}

out << endl ;

vector <vector <float> > kmat ;

kmat.push\_back(X0);

float error = 1;

int k = 0;

vector <float> Errorlst;

while(error >= 1e-06){

vector<float> sol(n) ;

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

// xi = bi - Sigma aijxi

float sum = 0;

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

if(j == i){

continue;

}

sum += A[i][j]\*kmat[k][j];

}

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

sol[i]= x;

}

kmat.push\_back(sol);

k++;

error = Error(kmat[k],kmat[k-1]);

Errorlst.push\_back(error);

}

for(int j = 1; j< k+1; j++){

out << "Output of iteration "<< j << ": ";

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

out << kmat[j][i] << " " ;

}

out << endl;

}

out << "Error after each iteration- " << endl ;

for(auto x : Errorlst){

out << x << endl ;

}

}

int main() {

Jacobimethod();

}

**Gauss Seidel:**

#include<bits/stdc++.h>

using namespace std;

float Error(vector<float>&v1, vector<float>&v2){

float error = 0;

for(int i = 0; i < v1.size(); i++){

error += pow((v1[i]-v2[i]),2);

}

return sqrt(error);

}

void GaussSidel(){

ifstream in("Input1.txt");

ofstream out("Output2.txt");

int n;

in >> n ;

vector <vector <float> > A (n , vector<float>(n,0));

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

out << "Coeffecients of equation " << i+1 << " - " << endl;

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

in >> A[i][j];

out << A[i][j] << endl;

}

}

vector<float> B(n,0);

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

in >> B[i];

}

vector<float> X0 (n);

out << "Initial guess - ";

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

in >> X0[i];

out << X0[i]<< " " ;

}

out << endl ;

vector <vector <float> > kmat ;

kmat.push\_back(X0);

float error = 1;

int k = 0;

vector<float> Errorlst ;

while(error >= 1e-06){

vector<float> sol(n,0) ;

kmat.push\_back(sol);

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

// xi = bi - Sigma aijxi

float sum = 0;

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

if(j == i){

continue;

}

else if(j<i){

sum += A[i][j]\*kmat[k+1][j];

}

else if(j>i){

sum += A[i][j]\*kmat[k][j];

}

}

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

kmat[k+1][i] = x;

}

k++;

error = Error(kmat[k],kmat[k-1]);

Errorlst.push\_back(error);

}

for(int j = 1; j< k+1; j++){

out << "Output of iteration "<< j << ": ";

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

out << kmat[j][i] << " " ;

}

out << endl;

}

out << "Error after each iteration- " << endl ;

for(auto x: Errorlst){

out << x << endl ;

}

}

int main() {

GaussSidel();

}

**Gauss Seidel with SOR (w = 1.5)**

#include<bits/stdc++.h>

using namespace std;

float Error(vector<float>&v1, vector<float>&v2){

float error = 0;

for(int i = 0; i < v1.size(); i++){

error += pow((v1[i]-v2[i]),2);

}

return sqrt(error);

}

void GaussSidel(){

ifstream in("Input1.txt");

ofstream out("Output3.txt");

int n;

in >> n ;

vector <vector <float> > A (n , vector<float>(n,0));

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

out << "Coeffecients of equation " << i+1 << " - " << endl;

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

in >> A[i][j];

out << A[i][j] << endl;

}

}

vector<float> B(n,0);

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

in >> B[i];

}

vector<float> X0 (n);

out << "Initial guess - ";

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

in >> X0[i];

out << X0[i]<< " " ;

}

out << endl ;

vector <vector <float> > kmat ;

kmat.push\_back(X0);

float error = 1;

int k = 0;

vector<float> Errorlst ;

while(error >= 1e-06){

vector<float> sol(n,0) ;

kmat.push\_back(sol);

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

// xi = bi - Sigma aijxi

float sum = 0;

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

if(j == i){

continue;

}

else if(j<i){

sum += A[i][j]\*kmat[k+1][j];

}

else if(j>i){

sum += A[i][j]\*kmat[k][j];

}

}

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

x = 1.5\*x - 0.5\*kmat[k][i];

kmat[k+1][i] = x;

}

k++;

error = Error(kmat[k],kmat[k-1]);

Errorlst.push\_back(error);

}

for(int j = 1; j< k+1; j++){

out << "Output of iteration "<< j << ": ";

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

out << kmat[j][i] << " " ;

}

out << endl;

}

out << "Error after each iteration- " << endl ;

for(auto x: Errorlst){

out << x << endl ;

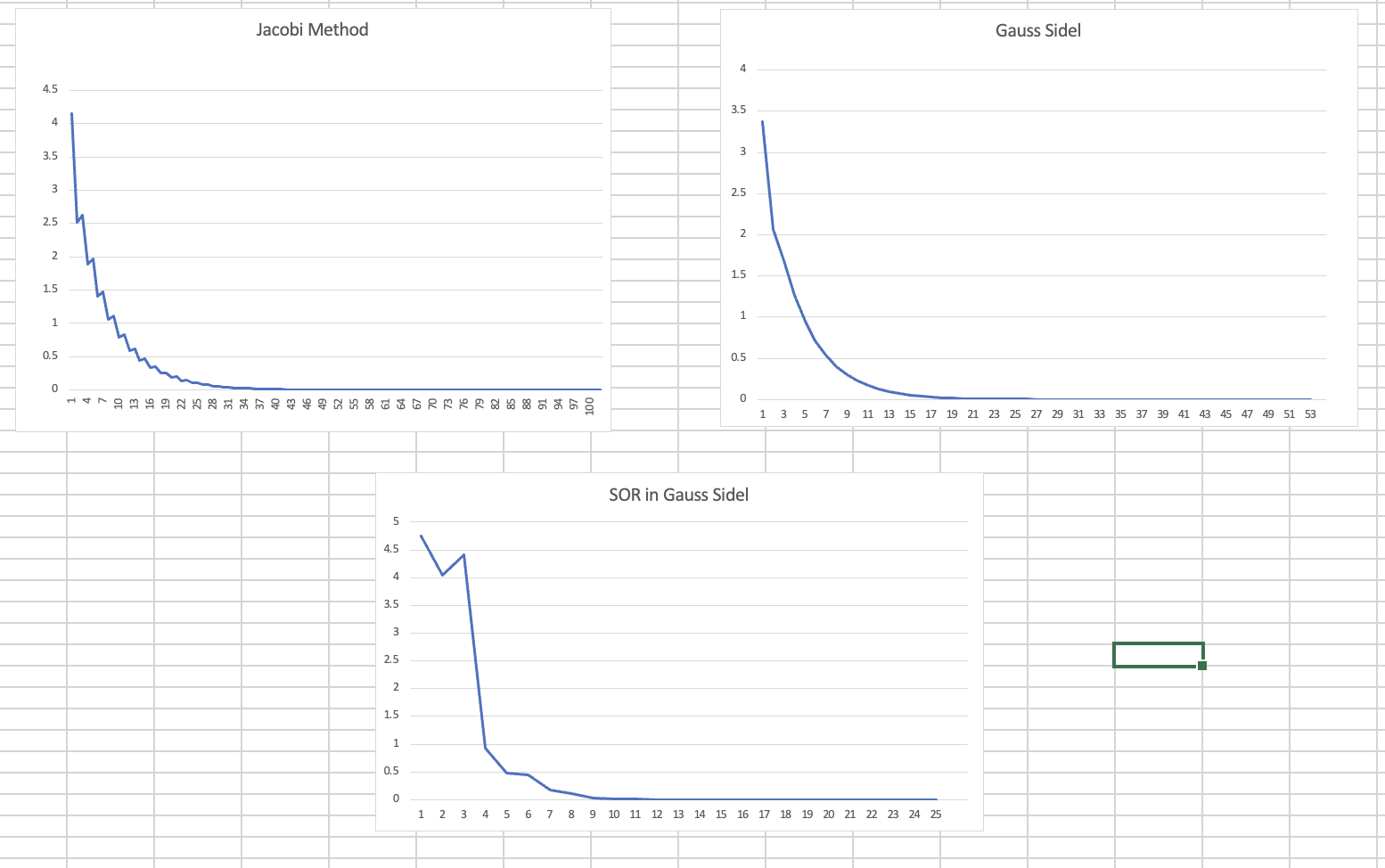
}

}

int main() {

GaussSidel();

}

**Graphs- Error vs iterations**

**Input:**

3

2 1 0

1 2 1

0 1 1

1

2

4

0 0 0

**Conclusion:**

We can clearly see for the Jacobi method, the number of iterations required is 100. Gauss Seidel which is a more efficient method which can be seen as it converges in 50 iterations. The convergence of Gauss Seidel technique is speeded up using successive over-relaxation (SOR). Using SOR technique with w = 1.5 the solution is reached in 22 iterations