#include <iostream>

#include <vector>

#include <math.h>

#include <numeric>

#include <string>

using namespace std;

vector<vector<float>> krilov(vector<vector<float>> A, vector<float> x, int k)

{

    vector<vector<float>> result(k, vector<float>(k + 1));

    // Fill first column with initial vector x

    for (int i = 0; i < k; i++)

        result[i][0] = x[i];

    // Fill remaining columns with matrix-vector products

    for (int j = 1; j <= k; j++)

    {

        for (int i = 0; i < k; i++)

        {

            float sum = 0;

            for (int l = 0; l < k; l++)

                sum += A[i][l] \* result[l][j - 1];

            result[i][j] = sum;

        }

    }

    return result;

}

vector<float> GS(vector<vector<float>> equations)

{

    vector<float> result(equations.size());

    vector<float> previousResult(equations.size());

    float error, sumResults = 0, sumPrevResults = 0;

    do

    {

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

        {

            previousResult[i] = result[i];

            result[i] = equations[i][equations[0].size() - 1];

            for (int j = 0; j < result.size(); j++)

            {

                if (i != j)

                    result[i] -= (result[j] \* equations[i][j]);

            }

            result[i] /= equations[i][i];

        }

        sumResults = accumulate(result.begin(), result.end(), 0);

        sumPrevResults = accumulate(previousResult.begin(), previousResult.end(), 0);

        error = abs((sumResults - sumPrevResults) / sumResults);

    } while (error > 0.00001);

    return result;

}

void printMatrix(vector<vector<float>> matrix)

{

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

    {

        for (int j = 0; j < matrix[i].size(); j++)

            cout << matrix[i][j] << " ";

        cout << endl;

    }

}

// Funcion para ver sis la matriz es diagonalmente dominante

bool isDiagonallyDominant(vector<vector<float>> A)

{

    int n = A.size();

    for (int i = 0; i < n; i++)

    {

        float diagonal = abs(A[i][i]);

        float sum = 0;

        for (int j = 0; j < n; j++)

        {

            if (i != j)

            {

                sum += abs(A[i][j]);

            }

        }

        if (diagonal <= sum)

        {

            return false;

        }

    }

    return true;

}

// Función para convertir una matriz en diagonalmente dominante

void makeDiagonallyDominant(vector<vector<float>> &A)

{

    int n = A.size();

    for (int i = 0; i < n; i++)

    {

        float maxVal = abs(A[i][i]);

        int maxIdx = i;

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

        {

            if (abs(A[j][i]) > maxVal)

            {

                maxVal = abs(A[j][i]);

                maxIdx = j;

            }

        }

        // Intercambia la fila actual con la fila que contiene el elemento máximo

        if (maxIdx != i)

        {

            swap(A[i], A[maxIdx]);

        }

    }

}

// Función que convierte una matriz en diagonalmente dominante si es posible

bool convertToDiagonallyDominant(vector<vector<float>> &A)

{

    if (isDiagonallyDominant(A))

    {

        // La matriz ya es diagonalmente dominante, no hace falta hacer nada

        return true;

    }

    // Intenta hacer la matriz diagonalmente dominante

    makeDiagonallyDominant(A);

    if (isDiagonallyDominant(A))

    {

        return true;

    }

    // La matriz no se pudo convertir en diagonalmente dominante

    return false;

}

int main()

{

    vector<vector<float>> A = {

        {1, 1, 1},

        {0, 2, 2},

        {3, -1, 0}};

    vector<float> x = {1, 0, 0};

    int k = 3;

    // Usamos Krilov para obtener el sistema de ecuaciones

    vector<vector<float>> equations = krilov(A, x, k);

    convertToDiagonallyDominant(equations);

    // Resolvemos el sistema con Gauss-Seidel

    vector<float> valuesGS = GS(equations);

    // Imprimimos el sistema de ecuaciones

    cout << "\nCon Krilov:\n";

    for (int i = 0; i <= k; i++)

    {

        cout << ((i != k) ? ("b" + to\_string(i) + "(") : (string((pow(-1, k) == -1) ? "-(" : "(")));

        for (int j = 0; j < k; j++)

            cout << ((j > 0) ? " " : "") << equations[j][i];

        cout << ")" << ((i != k) ? string((i < k - 1) ? " + " : " =\n") : "\n");

    }

    // Imprimimos el valor de los coeficientes

    cout << "\nCon Gauss-Seidel:\n";

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

        cout << "b" << i + 1 << " = " << valuesGS[i] << endl;

    cout << endl;

    return 0;

}

