Лабораторная работа № 11

Решение системы линейных алгебраических уравнений (СЛАУ) методами простых итераций и Зейделя.

Цель работы: изучить метод простых итераций и метод Зейделя для решения системы линейных алгебраических уравнений. Написать программу на Си по поиску решения СЛАУ данными методами.

Краткие теоретические сведения .

Методы решения систем линейных алгебраических уравнений классифицируют на прямые (точные) и итерационные. Прямые методы основаны на выполнении конечного числа арифметических операций, например, это метод Гаусса. Суть итерационных методов, в свою очередь, заключаются в том, чтобы за счет последовательных приближений получить решение системы, определяемое необходимой точностью. Итерационные методы характеризуются большими расчетными объемами, что не мешает им быть по своей структуре достаточно простыми. За счет предыдущих приближений мы получаем новые приближения, и, если система удовлетворяет условию сходимости, то эти приближения все меньше и меньше отличаются от аналитического решения. Для итерационных методов можно выделить три последовательных этапа: 1. Приведение исходной системы вида к итерационной форме 2. Проверка условия сходимости. 3. Решение системы одним из методов. Метод простых итераций Для общего вида систем должно выполняться тождество m=n, где m - количество уравнений в системе, n - количество неизвестных. Т.е. не имеет смысла решать недоопределенные (mn) системы, т.к. они могут быть сведены путем элементарных алгебраических преобразований к нормальным (m=n) системам линейных уравнений. Другими словами, если у вас имеется «ненормальная» система, то прежде, чем использовать метод простых итераций, преобразуйте ее к нормальной. Система линейных уравнений может быть записана в матричной форме, где A – матрица коэффициентов, b – вектор свободных членов, x – вектор неизвестных.

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

int main()

{

double maxd, maxdp = 0, \*B, \*X, \*d, \*\*A, ESP;

int i, j, n;

printf("Enter size of array: ");

scanf("%d", &n);

B = (double\*)malloc(n \* sizeof(double));

X = (double\*)malloc(n \* sizeof(double));

d = (double\*)malloc(n \* sizeof(double));

A = (double\*\*)malloc(n \* sizeof(double\*));

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

{

B[i] = 0; X[i] = 0; d[i] = 0;

A[i] = (double\*)malloc(n \* sizeof(double));

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

{

A[i][j] = 0;

}

}

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

{

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

{

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

scanf("%lf", &A[i][j]);

}

printf("\n");

}

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

{

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

scanf("%lf", &B[i]);

}

printf("\nEnter measurement accurecy: ");

scanf\_s("%lf", &ESP);

do

{

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

{

d[i] = X[i];

X[i] = B[i];

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

{

if (i != j)

{

X[i] -= A[i][j] \* X[j];

}

}

X[i] = X[i] / A[i][i];

d[i] = fabs(d[i] - X[i]);

}

maxd = d[0];

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

{

if (d[i] > maxd) maxd = d[i];

}

if (maxdp == 0)

{

maxdp = maxd;

continue;

}

else

{

if (maxdp - maxd < 0)

{

break;

}

else

{

maxdp = maxd;

}

}

} while (maxd > ESP);

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

{

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

}

system("pause");

}