**«Решение систем линейных уравнений методом Гаусса»**

**Цель**: Решая предоставленные задания, изучить решение систем линейных уравнений методом Гаусса

**Задачи**:

Разработать программу, решающую системы линейных уравнений методом Гаусса

Решить предоставленные задачи

**Код программы**:

function gauss\_Stand(a,q) {

let time = Date.now();

const n = a.length;

let x = [];

//Приведение матрицы к треугольному виду

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

let tmp = a[i][i];

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

a[i][j] = (a[i][j]/tmp);

}

for (let k = i + 1; k < n; k++) {

for (let j = i+1; j < (n+1); j++) {

a[k][j] = (a[k][j]-(a[i][j] \* a[k][i]));

}

}

}

//Обратный ход

x[n - 1] = +(a[n - 1][n] / a[n - 1][n - 1]).toFixed(q);

for (let i = n - 2; i >= 0; i--) {

let sum=0;

for (let j = i + 1; j < n; j++) {

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

}

x[i] = +(a[i][n] - sum).toFixed(q);

}

console.log("Время выполнения: ",((Date.now() - time)/100));

return x;

}

function gauss\_Opt(a,q) {

let time = Date.now();

//прямой ход

const n = a.length;

let x = [];

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

for (let k = i + 1; k < n; k++) {

let temp = a[k][i];

for (let j = i; j < (n+1); j++) {

a[k][j] = +(a[k][j]-a[i][j]\*(temp/a[i][i])).toFixed(q);

}

}

}

//Обратный ход

x[n - 1] = +(a[n-1][n] / a[n - 1][n - 1]).toFixed(q);

for (let i = n - 2; i >= 0; i--) {

let sum=0;

for (let j = i + 1; j < n; j++) {

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

}

x[i] = +((a[i][n] - sum)/a[i][i]).toFixed(q);

}

console.log("Время выполнения: ",((Date.now() - time)/100));

return x;

}

function gauss\_Stolb(a,q){

const n = a.length;

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

//Поиск максимального элемента в первом столбце

let max = Math.abs(a[j][j]);

let i = j;

for(let m = j; m<n; m++) {

if (Math.abs(a[m][j]) > max) {

i = m;

max = Math.abs(a[m][j]);

}

}

if (i != j){

// перестановка i-ой строки, содержащей главный элемент j-ой строки

let tmp = a[j];

a[j]=a[i];

a[i]=tmp;

}

}

return gauss\_Stand(a,q);

}

function gauss\_Strok(a, q) {

const n = a.length;

let aTmp=[];

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

aTmp[i]=i+1;

}

a.push(aTmp);

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

//Поиск максимального элемента в строке

let max = Math.abs(a[i][i]);

for (let j=i;j<n-1;j++){

max = a[i][j]>max ? Math.abs(a[i][j]) : max;

}

let numMax = a[i].indexOf(max);

if (numMax != i){

// перестановка j-го столбца, содержащей главный элемент k=i-ой строки

for (let k=0;k<=n;k++) {

let tmp = a[k][i];

a[k][i] = a[k][numMax];

a[k][numMax] = tmp;

}

}

}

let numer = a.pop();

numer.pop();

let res = gauss\_Stand(a,q);

res.push(numer);

return res;

}

function gauss\_StrokAndStolb(a,q) {

const n = a.length;

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

//Поиск максимального элемента в первом столбце

let max = Math.abs(a[j][j]);

let i = j;

for(let m = j; m<n; m++) {

if (Math.abs(a[m][j]) > max) {

i = m;

max = Math.abs(a[m][j]);

}

}

if (i != j){

// перестановка i-ой строки, содержащей главный элемент j-ой строки

let tmp = a[j];

a[j]=a[i];

a[i]=tmp;

}

}

let aTmp=[];

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

aTmp[i]=i+1;

}

a.push(aTmp);

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

//Поиск максимального элемента в строке

let max = Math.abs(a[i][i]);

for (let j=i;j<n-1;j++){

max = a[i][j]>max ? Math.abs(a[i][j]) : max;

}

let numMax = a[i].indexOf(max);

if (numMax != i){

// перестановка j-го столбца, содержащей главный элемент k=i-ой строки

for (let k=0;k<=n;k++) {

let tmp = a[k][i];

a[k][i] = a[k][numMax];

a[k][numMax] = tmp;

}

}

}

let numer = a.pop();

numer.pop();

let res = gauss\_Stand(a,q);

res.push(numer);

return res;

}

function gauss\_jord(a,q) {

let time = Date.now();

const n = a.length;

let x =[];

//Приведение матрицы к диагональномму виду

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

let tmp = a[i][i];

for (let j = i; j <= n; j++) {

a[i][j] = (a[i][j]/tmp).toFixed(q);

}

for (let l = i+1; l < n; l++) {

tmp = a[l][i];

for (let j = i+1; j <= n; j++) {

a[l][j] = +(a[l][j]-a[i][j] \* tmp).toFixed(q);

}

a[l][i]=Number(0).toFixed(q);

}

}

for (let i=(n-2); i>=0; i--) {

let sum = 0;

for (let j = (n-1); j > i; j--) {

sum+=Number(a[i][j]);

a[i][j]=Number(0).toFixed(q);

}

a[i][n]=(a[i][n]-sum).toFixed(q);

}

//Обратный ход

x[n - 1] = +(a[n - 1][n] / a[n - 1][n - 1]).toFixed(q);

for (let i = n - 2; i >= 0; i--) {

let sum=0;

for (let j = i + 1; j < n; j++) {

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

}

x[i] = +(a[i][n] - sum).toFixed(q);

}

console.log("Время выполнения: ",((Date.now() - time)/100));

return x;

}

**Входные данные**:

1. [[5,7,6,5,23],

[7,10,8,7,32],

[6,8,10,9,33],

[5,7,9,10,31]]

1. [[5,7,6,5,23.1],

[7,10,8,7,31.9],

[6,8,10,9,32.9],

[5,7,9,10,31.1]]

**Результаты вычислений**:

Стандартный метод Гаусса:

1. Время выполнения: 0, [ 1, 1, 1, 1 ]
2. Время выполнения: 0, [ 14.6, -7.2, -2.5, 3.1 ]

Оптимальный метод Гаусса:

1. Время выполнения: 0.01, [ 1, 1, 1, 1 ]
2. Время выполнения: 0, [ 14.6, -7.2, -2.5, 3.1 ]

Метод Гаусса-Жордана:

1. Время выполнения: 0.01, [ 1, 1, 1, 1 ]
2. Время выполнения: 0.01, [ 1.02, -0.2, 0.65, 3.1 ]

Метод Гаусса c выбором главного элемента по столбцам:

1. Время выполнения: 0, [ 1, 1, 1, 1 ]
2. Время выполнения: 0,[ 14.6, -7.2, -2.5, 3.1 ]

Метод Гаусса c выбором главного элемента по строкам:

1. Время выполнения: 0, [ 1, 1, 1, 1, [ 2, 3, 1, 4 ] ]
2. Время выполнения: 0, [ -7.2, -2.5, 14.6, 3.1, [ 2, 3, 1, 4 ] ]

Метод Гаусса c выбором главного элемента по строкам и столбцам:

1. Время выполнения: 0, [ 1, 1, 1, 1, [ 2, 3, 4, 1 ] ]
2. Время выполнения: 0, [ -7.2, -2.5, 3.1, 14.6, [ 2, 3, 4, 1 ] ]