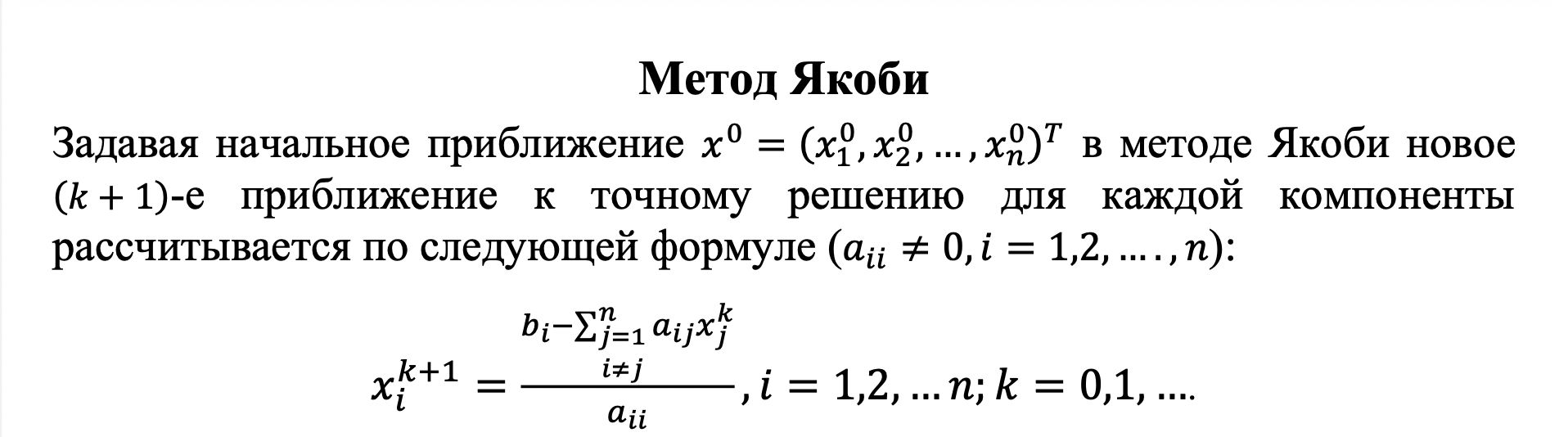
Work 6 Кириленко Константин 351





Условие сходимости:

Чтобы методы сходились, требуется выполнение некоторых условий. ДЛя Якоби должно быть диагональное преобладание. Для Зейделя - спектральный радиус меньше единицы, диагонального преобладание так же достаточно.

#define e 0.00001

// Function for random initialization of the matrix

// and the vector elements

void RandomDataInitialization(long double\* pMatrix, long double\* pVector,

int Size) {

long double s;

int i, j; // Loop variables

srand(unsigned(clock()));

for (i = 0; i < Size; i++) {

pVector[i] = rand() / double(1000);

for (j = 0; j < Size; j++) {

pMatrix[i \* Size + j] = rand() / double(1000);

}

}

for (i = 0; i < Size; i++) {

s = 0;

pVector[i] = rand() / double(1000);

for (j = 0; j < Size; j++) {

if (i != j)

s += pMatrix[i \* Size + j];

}

pMatrix[i \* Size + i] = rand() / double(1000) + s;

}

}

// Function for the execution of Gauss algorithm

void SerialResultCalculation(long double\* pMatrix, long double\* pVector,

long double\* pResult, int Size) {

long double\* g = new long double[Size];

long double\* d = new long double[Size];

int t = 0;

#pragma omp parallel for

for (int i = 0; i < Size; i++) { pResult[i] = 0; g[i] = 0; }

bool flag = false;

long double sum;

do {

flag = false;

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

g[i] = pVector[i];

sum = 0;

#pragma omp parallel for reduction(+: sum)

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

if (i != j) {

sum += (pMatrix[Size \* i + j] \* pResult[j]);

}

}

g[i] -= sum;

g[i] = (1.0 \* g[i]) / pMatrix[Size \* i + i];

}

#pragma omp parallel for

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

if (fabs(g[i] - pResult[i]) >= e) flag = true;

pResult[i] = g[i];

}

} while (flag);

delete[] g;

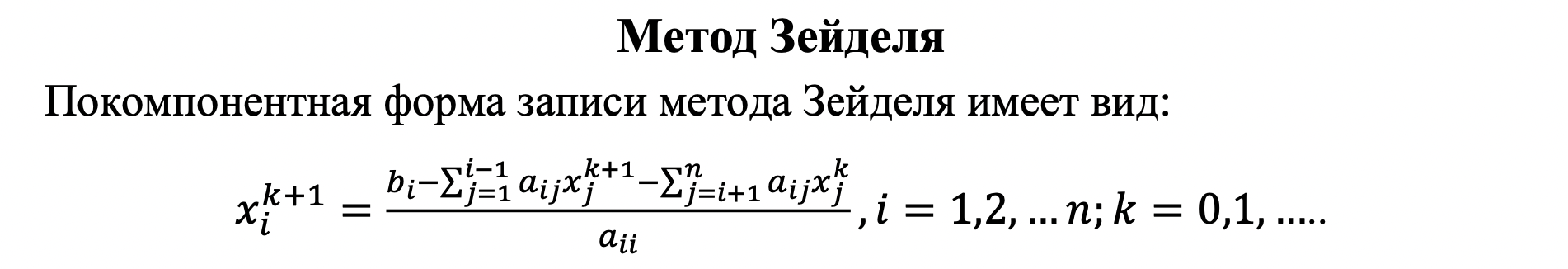
}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Порядок системы | Последовательный алгоритм | Параллельный алгоритм | |
| Время | Ускорение |
| 1 | 10 | 0.000000 | 0.368000 | 0 |
| 2 | 100 | 0.009000 | 0.186000 | 0.04839 |
| 3 | 500 | 0.671000 | 4.252000 | 0.157 |
| 4 | 1000 | 4.967000 | 9.658000 | 0.5143 |
| 5 | 1500 | 14.246000 | 17.503000 | 0.8139 |
| 6 | 2000 | 33.527000 | 42.566000 | 0.78764 |
| 7 | 2500 | 60.984000 | 63.186000 | 0.96515 |
| 8 | 3000 | 94.822000 | 97.411000 | 0.9734 |

Параллельный алгоритм работает на больших порядках быстрее. Чем больше размерность, тем больше ускорение

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Порядок системы | Ускорение алгоритма Гаусса | Ускорение алгоритма  Якоби |
| 1 | 10 | 0 | 0 |
| 2 | 100 | 0.0258 | 0.04839 |
| 3 | 500 | 0.2762 | 0.157 |
| 4 | 1000 | 0.4840 | 0.5143 |
| 5 | 1500 | 0.6706 | 0.8139 |
| 6 | 2000 | 0.9445 | 0.78764 |
| 7 | 2500 | 1.7147 | 0.96515 |
| 8 | 3000 | 1.8018 | 0.9734 |

Алгоритм Якоби не ускоряется, так как из-за особенности алгоритма большинство итераций приходится на цикл, который невозможно распараллелить.



#define e 0.0000000000001

void SerialResultCalculation(long double\* pMatrix, long double\* pVector,

long double\* pResult, int Size) {

long double\* g = new long double[Size];

long double\* d = new long double[Size];

int t = 0;

#pragma omp parallel for

for (int i = 0; i < Size; i++) { pResult[i] = 0; g[i] = 0; }

bool flag = false;

long double sum;

do {

//cout << t << endl;

//t++;

flag = false;

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

g[i] = pVector[i];

sum = 0;

#pragma omp parallel for reduction(+: sum)

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

if (i != j) {

sum += (pMatrix[Size \* i + j] \* g[j]);

}

}

g[i] -= sum;

g[i] = (1.0 \* g[i]) / pMatrix[Size \* i + i];

if (fabs(g[i] - pResult[i]) >= e) flag = true;

pResult[i] = g[i];

}

} while (flag);

delete[] g;

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Порядок системы | Последовательный алгоритм | Параллельный алгоритм | |
| Время | Ускорение |
| 1 | 10 | 0.000014 | 0.011651 | 0,0012 |
| 2 | 100 | 0.000308 | 0.033565 | 0,00917 |
| 3 | 500 | 0.005785 | 0.039082 | 0,14802 |
| 4 | 1000 | 0.025000 | 0.036373 | 0,68732 |
| 5 | 1500 | 0.061652 | 0.051225 | 1,20355 |
| 6 | 2000 | 0.108430 | 0.085845 | 1,26309 |
| 7 | 2500 | 0.196513 | 0.147830 | 1,32931 |
| 8 | 3000 | 0.256172 | 0.177108 | 1,44641 |