Физический факультет МГУ имени М.В. Ломоносова

Параллельное программирование
для ресурсоёмких задач численного моделирования в физике

Д.Н. Янышев, В.О. Милицин, И.А. Буткарев

Лекция № 8

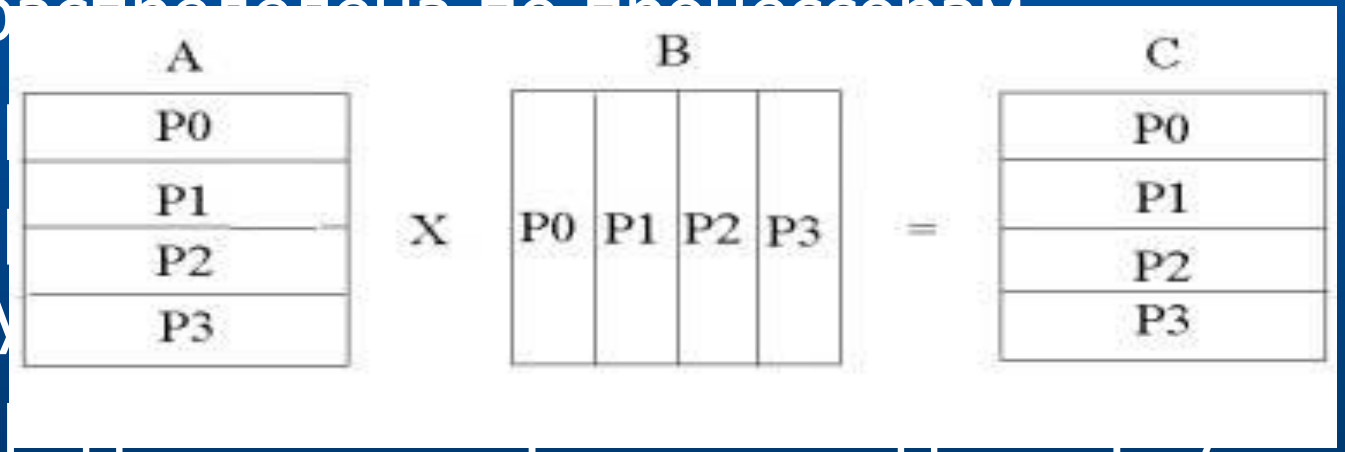
Содержание лекции

- Использование MPI
  - Перемножение матриц
  - Решение СЛАУ итерационным методом
  - Решение СЛАУ методом Гаусса
  - Минимизация функции

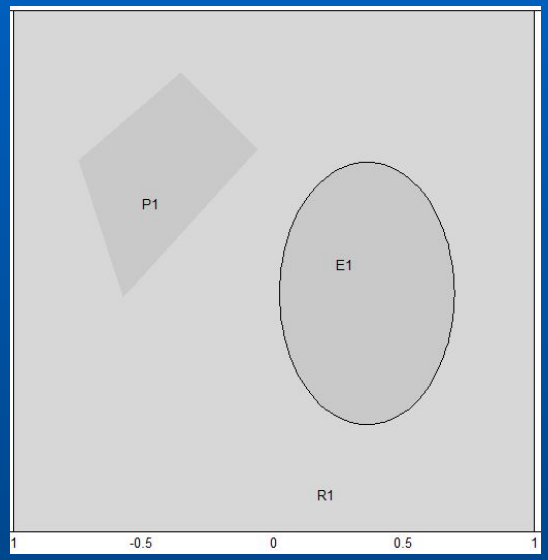
Перемножение матриц C=AB

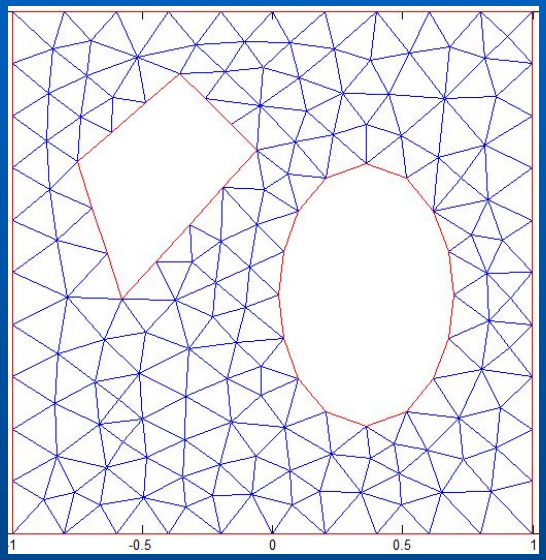
Каждая из трех матриц (A,B и C) может быть распределена одним из 4-х способов:

- не р
- расп
- расп
  сетку
- расп
  сетку

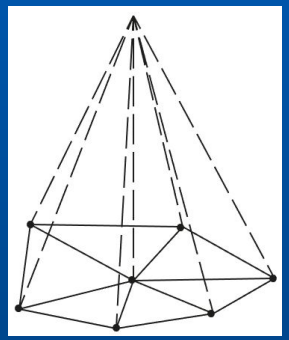


Метод конечных элементов





$\varphi_n(M)$



$Lu=f$

Базис: φ_n, $n=1,2,\ldots,N$

$$u_N = \sum_{n=1}^{N} c_n \varphi_n$$

c_n из условия ортогональности невязки:

$$(Lu_N - f, \varphi_n) = 0, \qquad n = 1,2,\ldots,N$$

Решение СЛАУ итерационным методом

Одношаговые (двухслойные) итерационные методы решения СЛАУ $\mathbf{A}y=f$:

$$\mathbf{B}_{n+1}\frac{y_{n+1}-y_n}{\tau_{n+1}}+\mathbf{A}y_n=f,\quad n=0,1,\ldots$$

$\mathbf{B}_{n+1}$ — итерационная матрица ($det\ \mathbf{B}_{n+1}\neq 0$),
$\tau_{n+1}>0$ — итерационный параметр.
Если $\mathbf{B}_{n+1}=\mathbf{B}$, $\tau_{n+1}=\tau$, то метод стационарный.
Если $\mathbf{B}\neq\mathbf{E}$, то метод неявный.

Метод релаксации ($\mathbf{A}y=f$, $\mathbf{B}=\mathbf{E}$):

$$y_{n+1} = y_n - \tau\left(\mathbf{A}y_n - f\right)$$

Достаточное условие сходимости:

$$0 < \tau < \frac{2}{\|\mathbf{A}\|}, \quad \|\mathbf{A}\| = \sup_{\|x\|\neq 0} \frac{(\mathbf{A}x, x)}{(x, x)}$$

Метод Якоби ($\mathbf{A}y=f$, $\mathbf{B}=\mathbf{D}$, $\tau=1$):

$$\mathbf{D}y_{n+1} = \mathbf{D}y_n - \left(\mathbf{A}y_n - f\right)$$

Решение СЛАУ методом Гаусса

Рассмотрим задачу

$$\mathbf{A}y = f$$

где $\mathbf{A}$ – матрица размерности NxN,
y, f - векторы размерности N

$$a + b + c = 6$$
$$2a - 3b + c = -9$$
$$-a + 2b/3 - c = 2$$

$$a = 1$$
$$b = 2$$
$$c = 3$$

Решение СЛАУ методом Гаусса

Рассмотрим задачу

$$\mathbf{A}y = f$$

где $\mathbf{A}$ – матрица размерности NxN,
y, f - векторы размерности N

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

$$\mathbf{L}g = f$$

$$\mathbf{U}y = g$$

$\mathbf{L}$ – нижнетреугольная матрица
$\mathbf{U}$ – верхнетреугольная матрица
с 1 на главной диагонали

Элементы матриц L и U:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}, \qquad при\ i \geq j, \quad l_{11} = a_{11}, \quad u_{11} = 1$$

$$u_{ij} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \right), \qquad при\ i < j$$

Может возникнуть ситуация, когда некоторое l_{ii} << 1, тогда необходимо производить выбор главного элемента по столбцу, по строке, либо по подматрице.

В данном случае удобно осуществлять выбор по столбцу, тогда обычная перенумерация строк позволит получить достаточно большой элемент на главной диагонали. Если такого элемента нет, матрица плохо обусловлена, и решать систему данным методом нельзя.

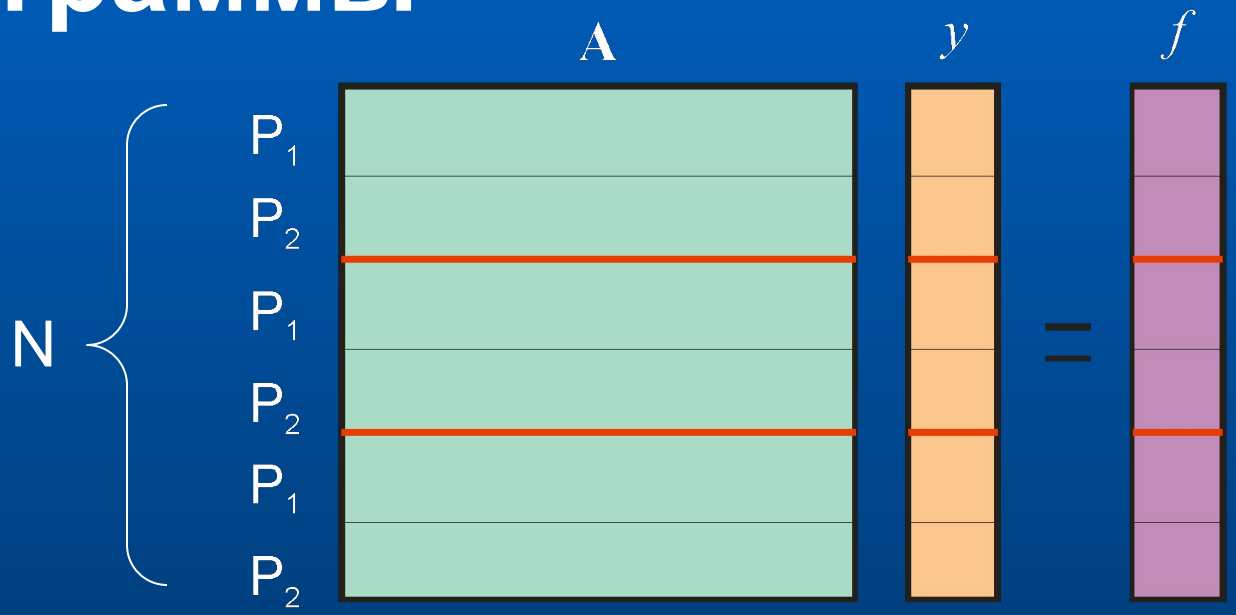
Последовательная программа:

```
double A[1:n,1:n], LU[1:n,1:n];  #предполагается, что A
                                 #инициализирована
int ps[1:n];  #индексы ведущих строк
double pivot; int pivotRow;  #ведущее значение и строка
double mult; int t; #временные переменные
#инициализация ps и LU
for [i = 1 to n] {
   ps[i] = i;
   for [j = 1 to n]
      LU[i,j] = A[i,j];
}
# исключение Гаусса с помощью ведущих элементов
for [k = 1 to n-1] {  #итерации вниз по главной диагонали
   pivot = abs(LU[ps[k],k]);
   pivotRow = k;
```

```
for [i = k+1 to n] {  #выбор главного элемента в столбце k
    if (abs(LU[ps[i],k]) > pivot) {
      pivot = abs(LU[ps[i],k]); pivotRow = i;
    }
  }
  if (pivotRow != k) {  #перестановка строк с помощью
                        #перестановки указателей
    t = ps[k]; ps[k] = ps[pivotRow]; ps[pivotRow] = t;
  }
  pivot = LU[ps[k],k];
  for [i = k+1 to n] {  #для всех строк в подматрице
    mult = LU[ps[i],k]/pivot;  #вычислить множитель
    LU[ps[i],k] = mult;        #и сохранить его
    for [j = k+1 to n]  #исключение по столбцам
      LU[ps[i],j] = LU[ps[i],j] - mult*LU[ps[k],j];
  }
}
```

```
double sum, y[1:n], f[1:n];
#прямой ход для решения Lg = f с записью g в y
for [i = 1 to n] {
   sum = 0.0;
   for [j = 1 to i - 1]
      sum = sum + LU[ps[i],j] * y[j];
   y[i] = f[ps[i]] - sum;
}
#обратный ход для решения Uy = g относительно y
for [i = n to 1 by -1] {
   sum = 0.0;
   for [j = i+1 to n]
      sum = sum + LU[ps[i],j] * y[j];
   y[i] = (y[i] - sum) / LU[ps[i],i];
}
```

Схема параллельной программы



Поскольку LU-разложение работает с уменьшающимися подматрицами, объем работы также уменьшается по мере выполнения исключений.

Параллельная программа

```
process Worker(w = 1 to PR) {
  double LU[1:n/PR,1:n];  #свои строки LU
  int ps[1:n/PR];         #индексы ведущих строк
  double pivot, mult, pivotRow[n];
  int myRow;
  декларация других локальных переменных;
  инициализация ps и своих строк в LU;
  #исключение Гаусса с главными элементами
  for [k = 1 to n-1] {  #итерации вниз по главной диагонали
    поиск наибольшего главного элемента в столбце k
                                  своих строк;
    обмен pivot с другими процессами;
    выбор глобального максимального элемента и
                          обновление ps;
```

```
if (владелец ведущей строки)
   рассылка копий pivotRow другим процессам;
else
   получение pivotRow;
#исключение своих строк в LU с помощью pivot и
#pivotRow
for [i = k+1 to n st (i%PR == )] {  #для своей полосы
   myRow = i/PR;  #преобразовать индекс строки
   mult = LU[ps[myRow],k]/pivot;  #вычислить
                                  #множитель
   LU[ps[myRow],k] = mult;        #и сохранить его
   for [j = k+1 to n]  #исключение по столбцам
      LU[ps[myRow],j] = LU[ps[myRow],j] - mult *
                                        pivotRow[j];
   }
}
}
```

Минимизация функции

$$y = \arg \inf_{x \in (a,b)} f(x)$$

Метод золотого сечения:

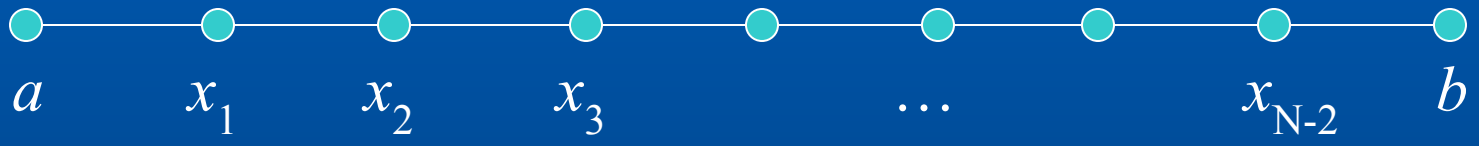
if $(f(c) < f(d))$
    отбросить b
else
    отбросить a

$$b - d = c - a$$

$$\frac{c-a}{b-a} = \frac{d-c}{d-a} = \xi$$

$$\xi = \frac{2}{3+\sqrt{5}}$$

Имеем N процессоров



$$f(x_k) = \min_{0<i<N-1} f(x_i)$$

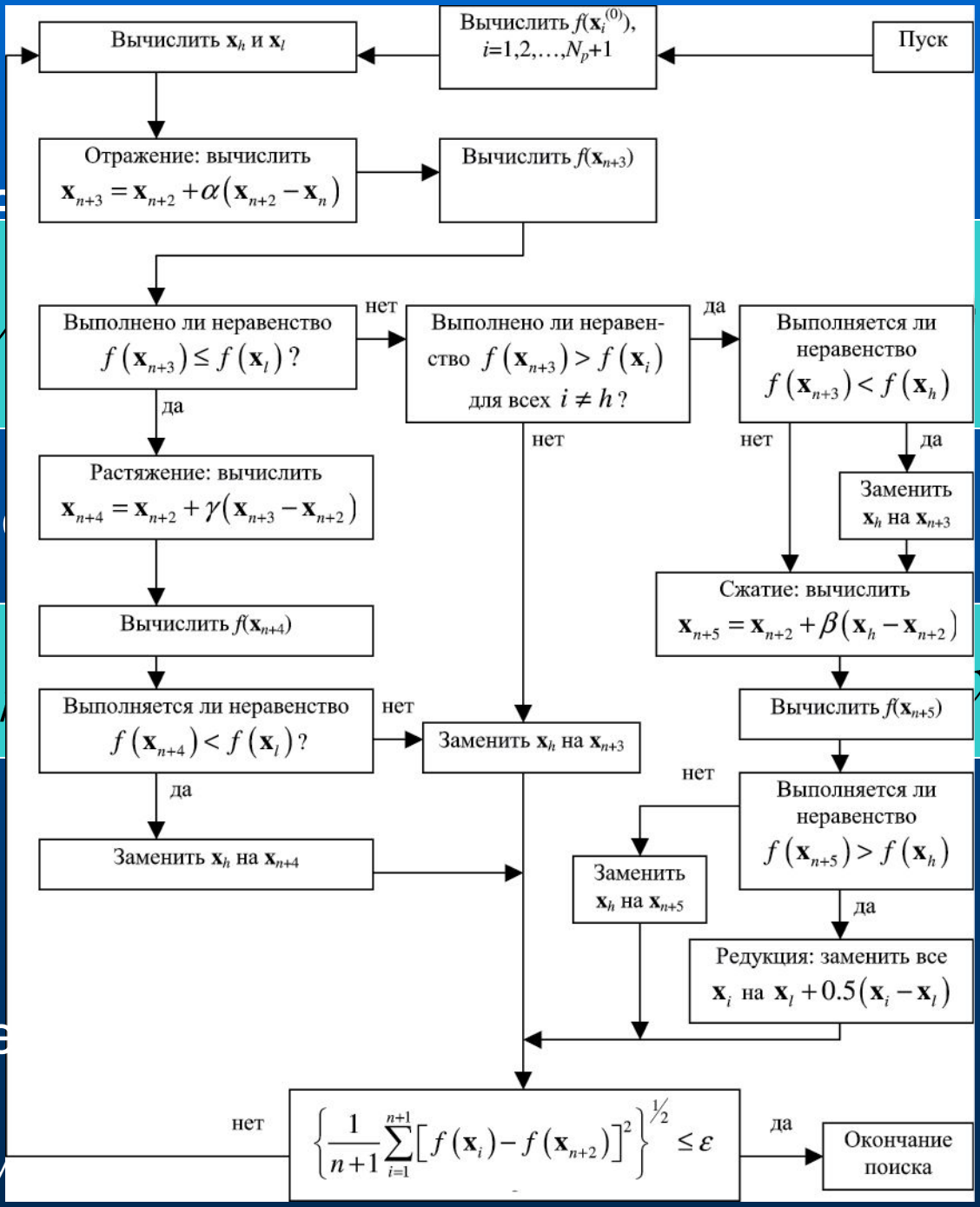
$$a' = x_{k-1}$$

$$b' = x_{k+1}$$

Миним
переме

Метод ко

Метод Не



Обратные задачи

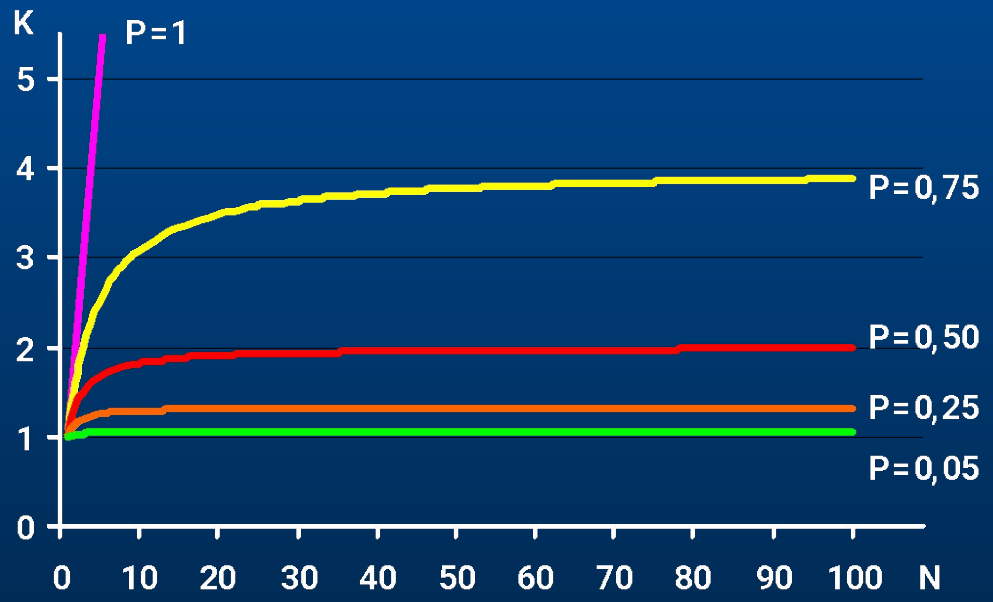
Требуется по наблюдаемому (задача распознавания) или требуемому (задача синтеза) поведению системы определить ее параметры.

Решение этих задач сводится к поиску минимума некоторого функционала, для чего требуется многократно решать прямые задачи.

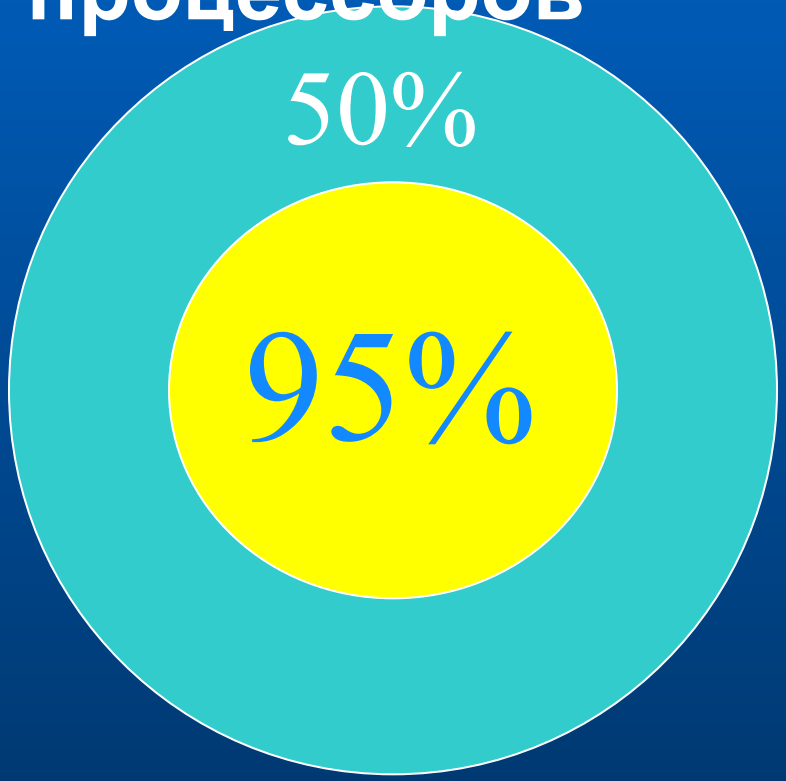
Закон Амдала

- коэффициент ускорения выполнения программы
  - *S* - доля последовательной части
  - *P* - доля параллельной части
  - *N* - число независимых ветвей/процессоров

$$K = \frac{1}{S + P/N}$$



Оптимизация распределения процессоров



1000 процессоров
Ускорение:

1000	1,998	19,627
500	25,7	
250	29,7	
200	30,4	
125	30,9	
100	30,6	
50	27,6	

Практическое задание

- Решение системы уравнений методом Гаусса

Основные параметры, по которым оценивается задание:

- возможность того, что число уравнений не кратно числу процессоров
- блочное либо циклическое распределение по процессорам
- выбор главного элемента (есть/нет)
- хранение матрицы системы (по частям/целиком)
- LU-разложение / Гаусс / приведение системы к диагональному виду
- распределение матрицы по строкам / столбцам