Новосибирский государственный университет
Факультет информационных технологий
Кафедра параллельных вычислений

Фрагментация алгоритмов реализации симплекс-метода и разработка параллельных программ

Подготовила
магистрантка ФИТ НГУ
А.И. Черникова
Научный руководитель
д.т.н., проф. В.Э. Малышкин

Новосибирск
2013

План

- Введение
- Наивный алгоритм
- Модифицированный алгоритм
- Результаты
- Направления дальнейшей работы

Введение

Технология фрагментированного программирования (ТФП) помогает решать проблемы программирования:

1. Автоматизация распределения ресурсов
2. Максимально параллельное исполнение на всех доступных ресурсах
3. Динамическая балансировка загрузки программы

Введение

Требования ТФП к представлению алгоритма:

- алгоритм состоит из фрагментов данных и фрагментов вычислений;
- число и размер фрагментов настраиваются;
- фрагменты примерно одинакового размера;
- используется математическая запись для переносимости на другое оборудование.

S.Kireev, V.Malyshkin Fragmentation of Numerical Algorithms for Parallel Subroutines Library // The Journal of Supercomputing. Vol. 57. Number 2. 2011. pp. 161-171

Актуальность и новизна

- ТФП позволяет сделать создание параллельных программ доступным для пользователей, неспециализирующихся в данной области.

- Система фрагментированного программирования (СФП) находится на стадии разработки, необходимо создание библиотеки фрагментированных подпрограмм для нее.

Цель работы

1. Исследовать возможности симплекс-метода для фрагментации.
2. Разработать фрагментированные алгоритмы симплекс-метода.
3. Реализовать алгоритмы в виде фрагментированных параллельных программ.

Задача линейного программирования

$$\max \vec{c} \cdot \vec{x}$$

$$A \cdot \vec{x} = \vec{b}$$

$$x_i \geq 0, \overline{i = 0, N-1}$$

G. B. Dantzig, Linear Programming and Extensions, 1949

Наивный (канонический) алгоритм симплекс-метода

- Прост в реализации
- Имеет ряд недостатков
  - Неэффективен для решения задач большого размера.
  - Устранение накопления ошибки округления ресурсозатратно.
- Выбран для отработки методики фрагментации алгоритмов

G. B. Dantzig, Linear Programming and Extensions, 1949

Наивный (канонический) алгоритм симплекс-метода

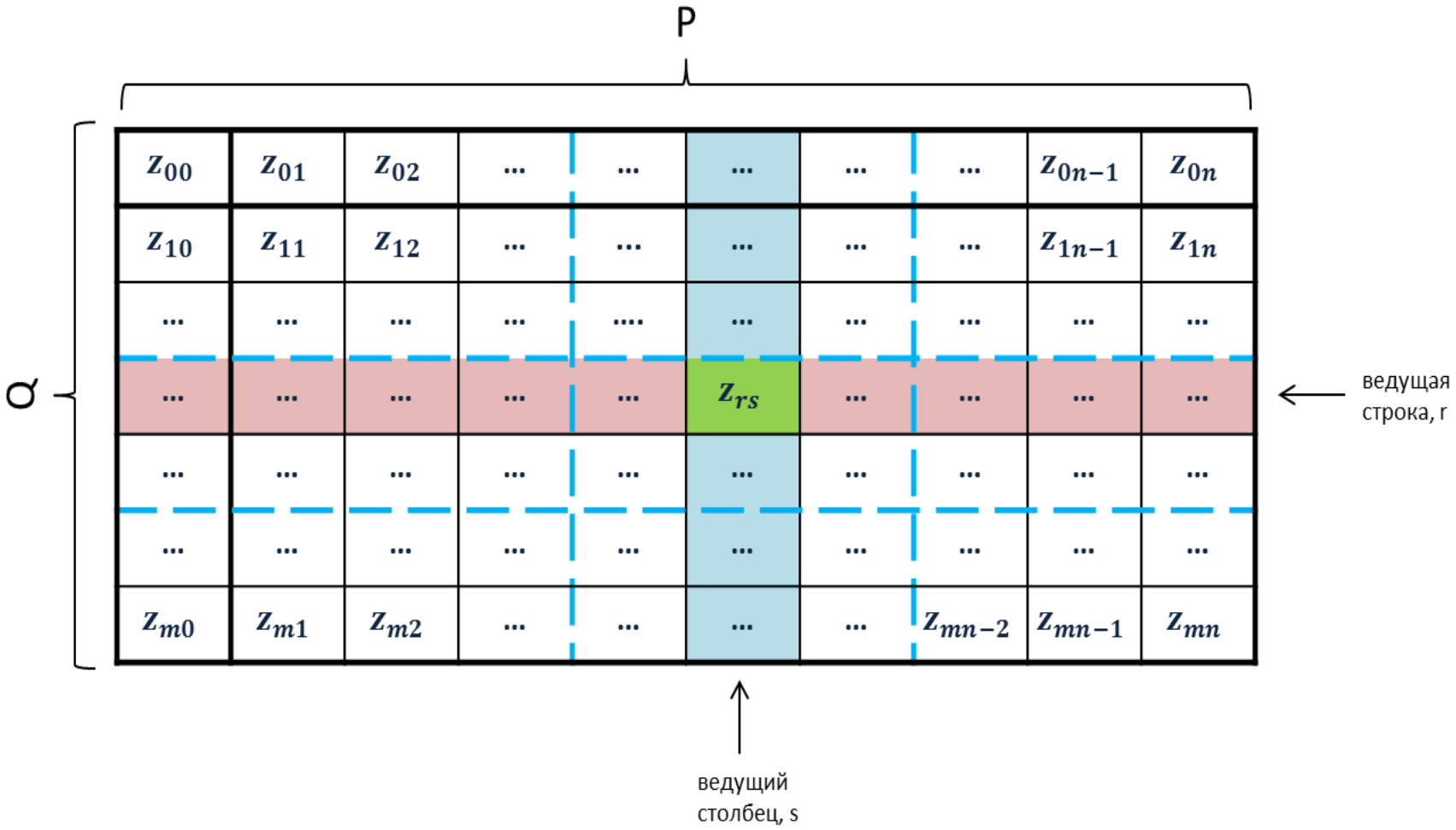
Данные

- Симплекс-таблица
- Ведущий столбец
- Ведущая строка
- Ведущий элемент

Операции

- Поиск номера ведущего столбца
- Поиск номера ведущей строки
- Пересчет симплекс-таблицы

Наивный (канонический) алгоритм симплекс-метода



Наивный (канонический) алгоритм симплекс-метода

Данные

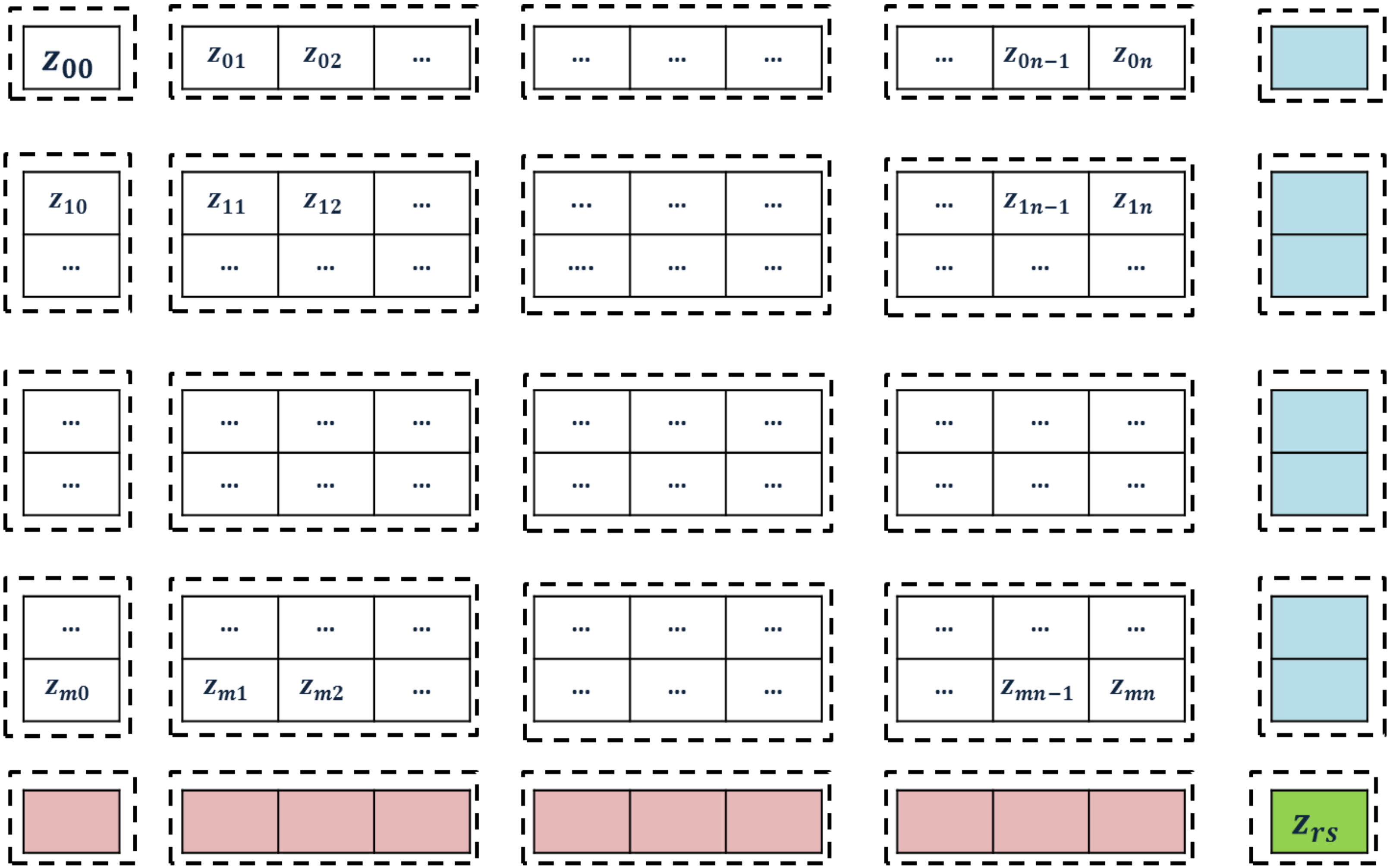
- Симплекс-таблица
- Ведущий столбец
- Ведущая строка
- Ведущий элемент

Операции

- Поиск номера ведущего столбца
- Поиск номера ведущей строки
- Пересчет симплекс-таблицы

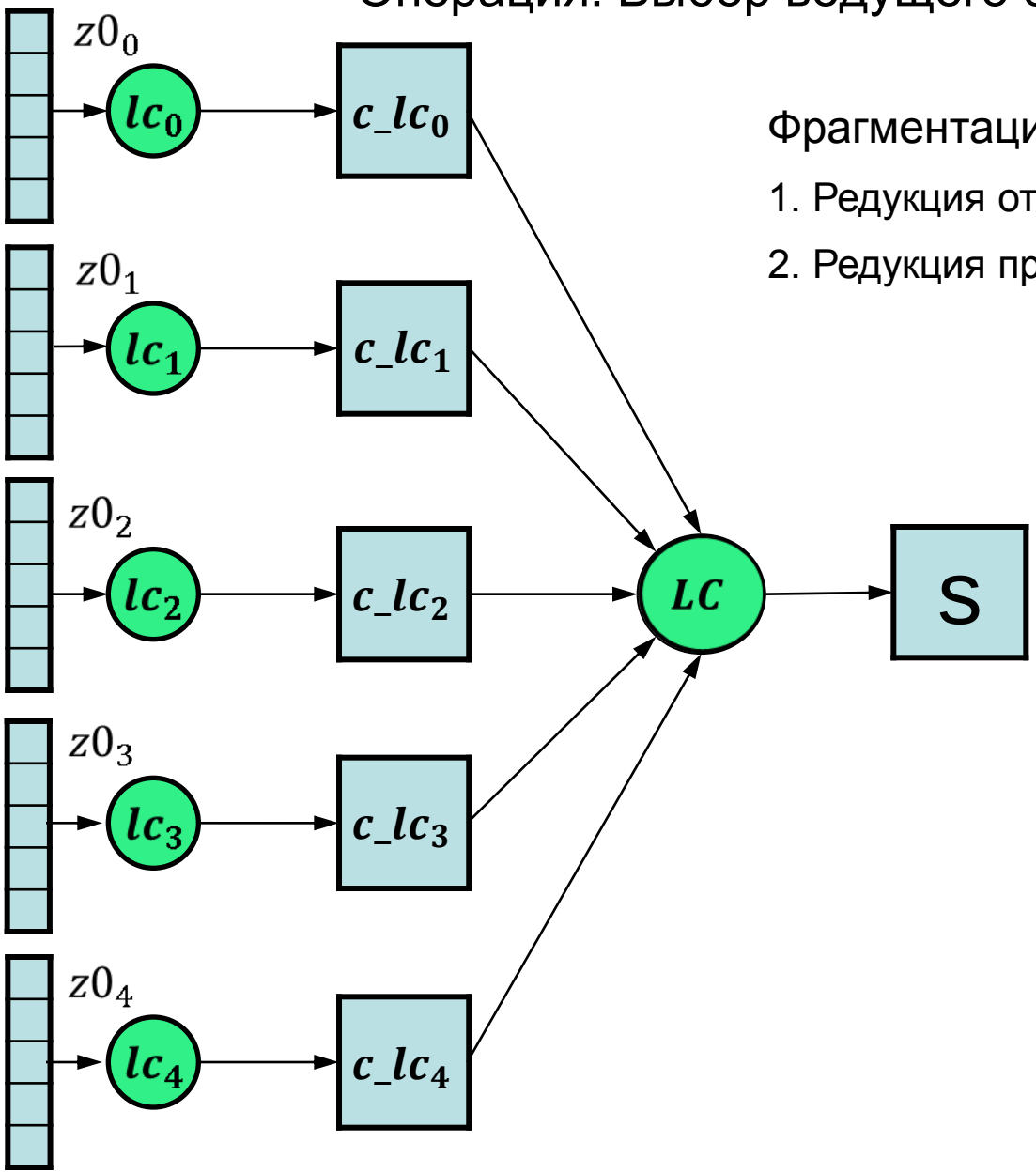
Фрагментация наивного алгоритма

Фрагментация данных



Фрагментация наивного алгоритма

Операция: Выбор ведущего столбца



Фрагментация операции редукции:

1. Редукция отдельных фрагментов
2. Редукция промежуточных результатов

Модифицированный алгоритм симплекс-метода

- Более сложен в реализации, чем наивный.
- Преодолевает недостатки наивного алгоритма:
  - Используется для решения задач большого размера.
  - Устранение накопления ошибки округления не требует значительных затрат.
- Используется в современных пакетах: ЛП-ВЦ, lp_solve 5.x и др.
- Выбран для создания фрагментированной подпрограммы для включения в библиотеку.

B.A. Murtagh, Advanced Linear Programming: Computation and Practice, 1981

Модифицированный алгоритм симплекс-метода

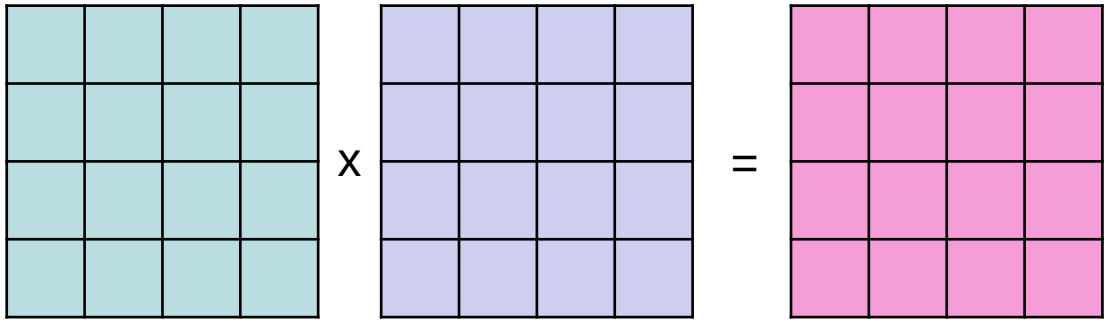
- Данные алгоритма:
  1. множество базисных столбцов;
  2. множество небазисных столбцов;
  3. вектор коэффициентов при целевой функции (базисные и небазисные компоненты);
  4. вектор правой части;
  5. и др.

- Операции алгоритма:
  1. выбор вводимого в базис элемента (редукция);
  2. выбор выводимого из базиса элемента (редукция);
  3. матричные операции (алгебраическая сумма, умножение);
  4. решение СЛАУ;
  5. переход к новому базису.

Матричные операции

Умножение

Фрагментация данных:



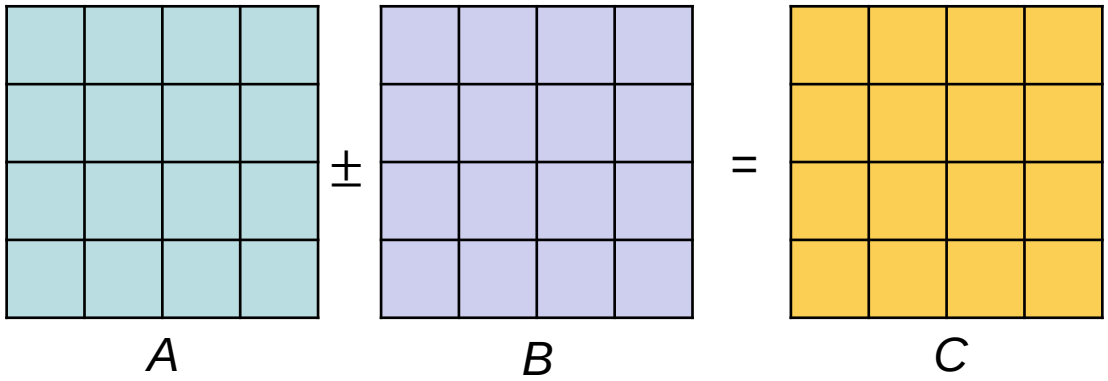
Фрагментированный алгоритм

$$A_{i,k} \times B_{k,j} = D_{i,k,j} \quad \forall\, i,j,k$$

$$\sum_{k} D_{i,k,j} = C_{i,j} \quad \forall\, i,j,k$$

Сумма

Фрагментация данных:



Фрагментированный алгоритм

$$A_{i,k} \pm B_{k,j} = D_{i,j} \quad \forall\, i,j$$

Операция: Решение СЛАУ

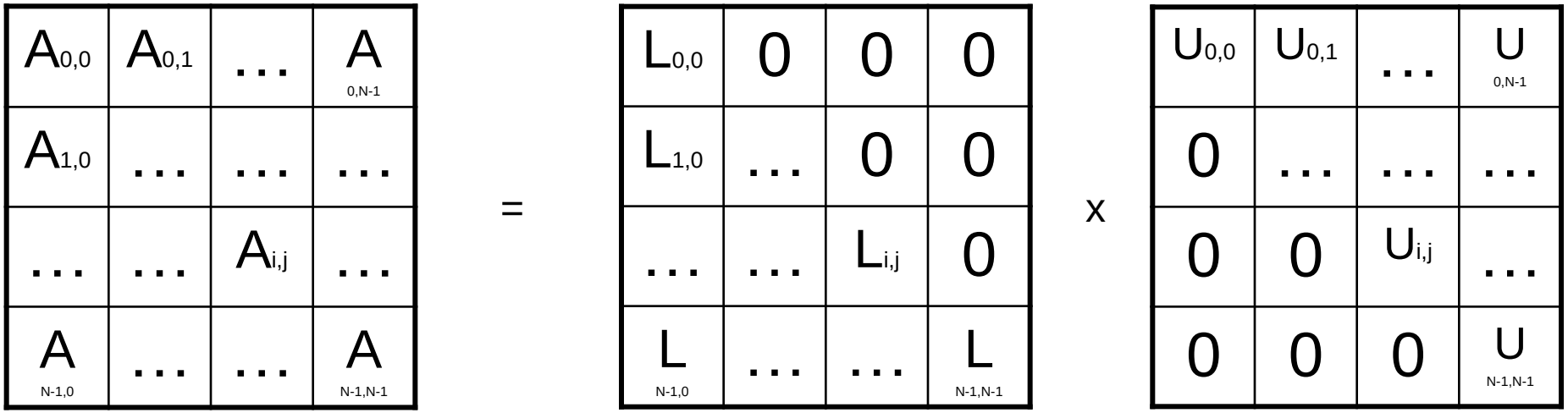
$$Fx = g$$

Решение СЛАУ с использованием LU-разложения:

1. найти LU-разложение основной матрицы системы: $F = LU$;
2. найти решение системы уравнений $Ly = g$;
3. найти решение исходной задачи из системы уравнений $Ux = y$.

Операция: LU-разложение

Фрагментация матриц:



Выведена зависимость между фрагментами матриц:

$$A_{i,j} = \begin{cases} \sum_{k=0}^{i} L_{i,k} \cdot U_{k,j}, i < j \\ \sum_{k=0}^{j} L_{i,k} \cdot U_{k,j}, i \geq j \end{cases} \qquad i,j = \overline{0, N-1}$$

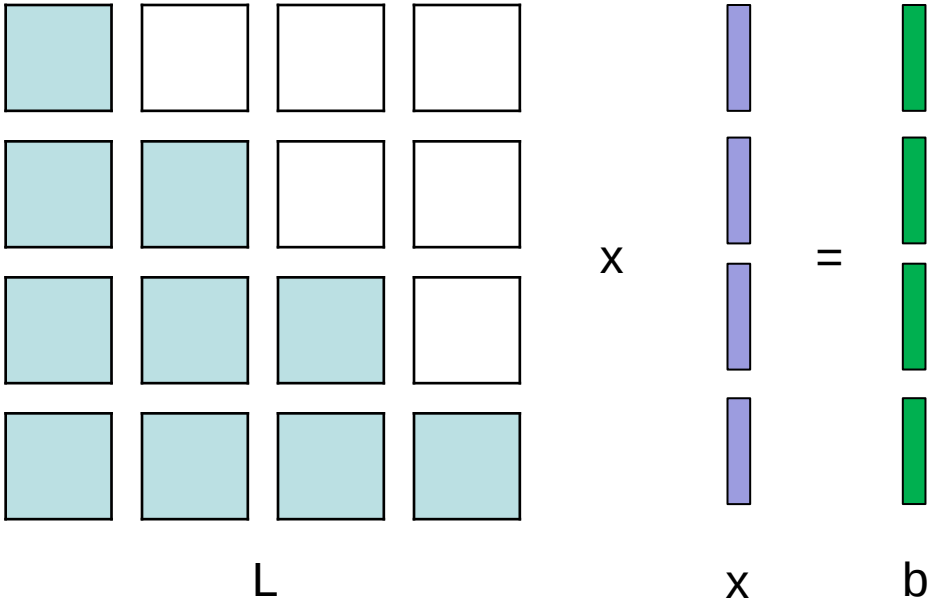
На основании зависимости получен фрагментированный алгоритм LU-разложения.

Фрагментированный алгоритм решения СЛАУ методом исключений

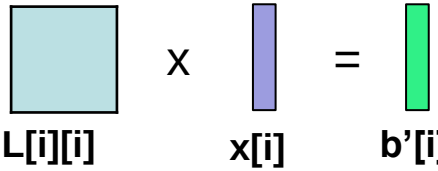
Алгоритм решения СЛАУ: *Lx=b*

```
L_GE(in: L, b; out: x)
{
  b'[0] = b[0];
  for i = 0…N-1
  {
    fr_L_GE(in: L[i][i], b'[i]; out: x[i]);
    for j = i+1…N-1
    {
      b'[j] = b[j] – L[j][i]*x[i];
    }
  }
}
```

Фрагментация данных:



fr_L_GE(in: L[i][i], b'[i]; out: x[i])



LU-разложение с полной перестановкой

1. Алгоритмы, реализующие LU-разложение неустойчивы
2. Перестановка строк и столбцов исходной матрицы повышает устойчивость метода

$$PFQ = LU$$

Фрагментированный алгоритм решения СЛАУ был дополнен локальными перестановками.

Результаты

- Рассмотрены два алгоритма симплекс-метода. В алгоритмах были выделены фрагменты данных и вычислений. Размер фрагментов вынесен в параметр алгоритма.
- Алгоритмы подготовлены к записи в системе фрагментированного программирования.
- Реализованы последовательные и параллельные фрагментированные программы на языке C++.

Направления дальнейшей работы

1. Разработка варианта фрагментированного алгоритма LU-разложения с глобальными перестановками.
2. Разработка более эффективного алгоритма операции перехода к новому базису.
3. Модификация фрагментированного модифицированного алгоритма симплекс-метода для использования разреженных матриц.
4. Реализация модифицированного алгоритма симплекс-метода в виде фрагментированной программы в СФП LuNA.

Апробация работы

- Работа представлена на конференциях:
  1. Конференция молодых ученых ИВМиМГ СО РАН, 2013
  2. 51-ая международная научная студенческая конференция «Студент и научно-технический прогресс», 2013
  3. IV Всероссийская конференция студентов Элитного технического образования «Ресурсоэффективным технологиям энергию и энетузиазм молодых»

- Публикации
  1. А.И. Черникова, Фрагментация двух алгоритмов симплекс-метода (тезисы) // Сборник материалов 51-ой международной научной студенческой конференции «Студент и научно-технический прогресс», Новосибирск, 2013, с. 20
  2. А.И. Черникова, Фрагментация двух алгоритмов симплекс-метода (статья) // Сборник трудов конференции молодых ученых ИВМиМГ СО РАН, Новосибирск, 2013 (в печати)
  3. А.И. Черникова, Фрагментация алгоритма симплекс-метода и разработка фрагментированных программ (тезисы) // Ресурсоэффективным технологиям – энергию и энтузиазм молодых: сборник докладов IV Всероссийской конференции студентов Элитного технического образования, Томск, 2013, с. 146-147

Спасибо за внимание!

Новосибирский государственный университет
Факультет информационных технологий
Кафедра параллельных вычислений

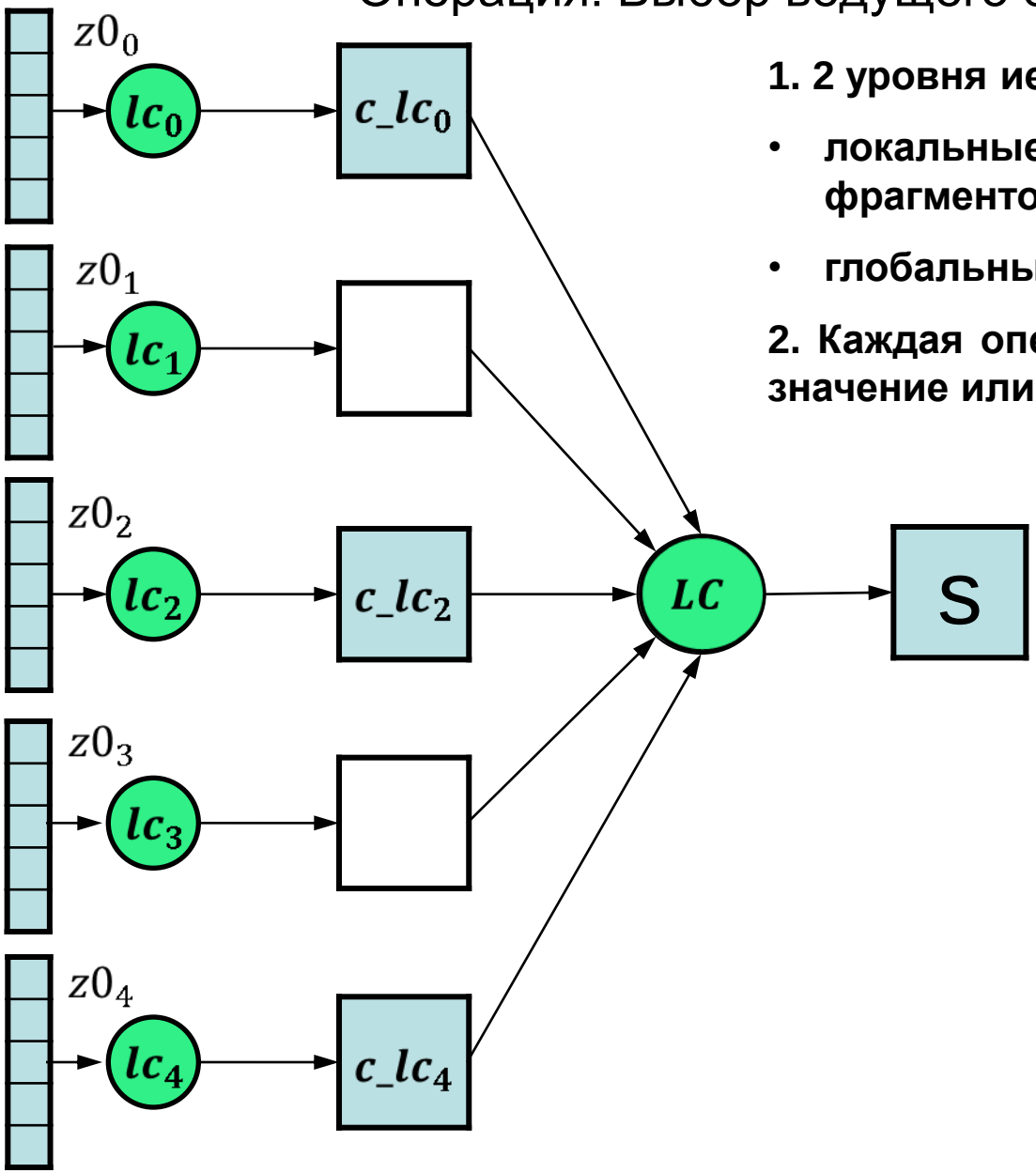
Фрагментация алгоритмов реализации симплекс-метода и разработка параллельных программ

Подготовила
магистрантка ФИТ НГУ
А.И. Черникова
Научный руководитель
д.т.н., проф. В.Э. Малышкин

Новосибирск
2013

Фрагментация наивного алгоритма

Операция: Выбор ведущего столбца



1. 2 уровня иерархии операций:

- **локальные – операции внутри фрагментов;**
- **глобальные – фрагменты вычислений.**

2. Каждая операция может вырабатывать значение или пустой фрагмент.

Модифицированный алгоритм

1. Выбрать базис B
2. $x_B \leftarrow B \cdot x_B = b$
3. $\pi \leftarrow B^T \cdot \pi = c_B$
4. $d = \pi^T \cdot N - c_N$
5. Если среди элементов d нет отрицательных элементов, то КОНЕЦ, решение найдено; иначе $d_s = \min d_j < 0, j = \overline{0, M-1}$, где s – номер выводимого из базиса вектора.
6. $\alpha_s = B^{-1} a_s$
7. Если среди элементов B_s нет положительных, то КОНЕЦ, решения не существует;
   иначе $\frac{A_{r,s}}{x_{B\,r}} = \min \frac{A_{i,s}}{x_{B\,i}} > 0, i = \overline{0, M-1}$, где r – номер вводимого в базис вектора.
8. Заменить в базисе A_s на A_r, перейти к шагу 2.

$$\max \vec{c} \cdot \vec{x}$$

$$A \cdot \vec{x} = \vec{b}$$

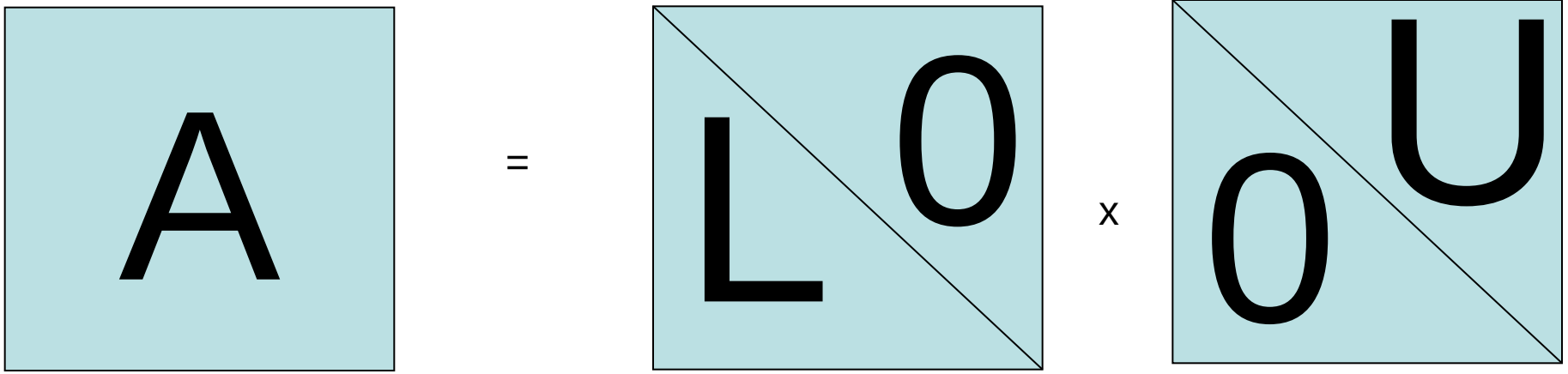
$$\overline{x_i \geq 0, i = 0, N-1}$$

$$A = \begin{bmatrix} B \\ N \end{bmatrix}$$

$$c = \begin{bmatrix} c_B \\ c_N \end{bmatrix}$$

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$$

Операция: LU-факторизация



$A_{0,0}$	$A_{0,1}$	…	$A_{0,N-1}$
$A_{1,0}$	…	…	…
…	…	$A_{i,j}$	…
$A_{N-1,0}$	…	…	$A_{N-1,N-1}$

=

$L_{0,0}$	0	0	0
$L_{1,0}$	…	0	0
…	…	$L_{i,j}$	0
$L_{N-1,0}$	…	…	$L_{N-1,N-1}$

x

$U_{0,0}$	$U_{0,1}$	…	$U_{0,N-1}$
0	…	…	…
0	0	$U_{i,j}$	…
0	0	0	$U_{N-1,N-1}$

$$A_{i,j} = \begin{cases} \sum_{k=0}^{i} L_{i,k} \cdot U_{k,j}, i < j \\ \sum_{k=0}^{j} L_{i,k} \cdot U_{k,j}, i \geq j \end{cases} \qquad i,j = \overline{0, N-1}$$

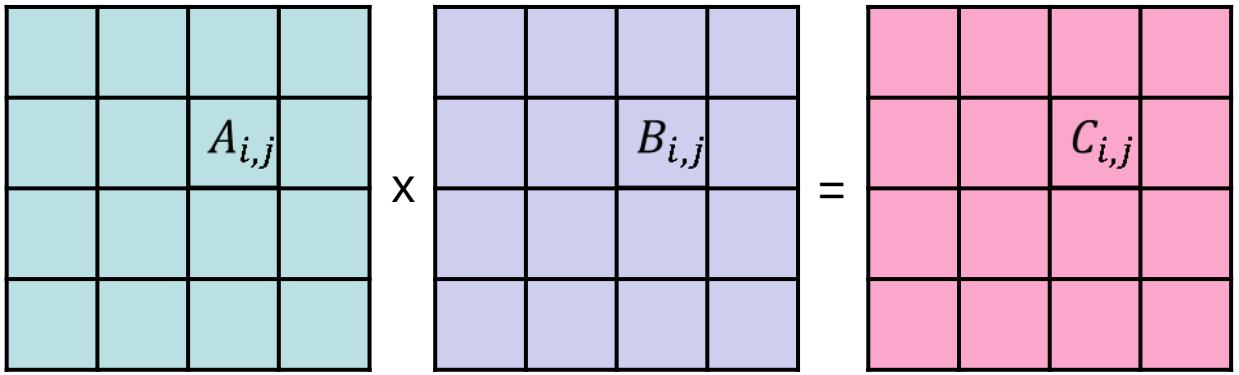
Фрагментация LU-факторизации с полной перестановкой

- 2 уровня перестановок: локальный и глобальный
- Локальные перестановки – внутри фрагментов
- Глобальные перестановки – обмен данными между фрагментами

$$A = (P^{-1} \cdot L) \cdot (U \cdot Q^{-1}) \; **$$

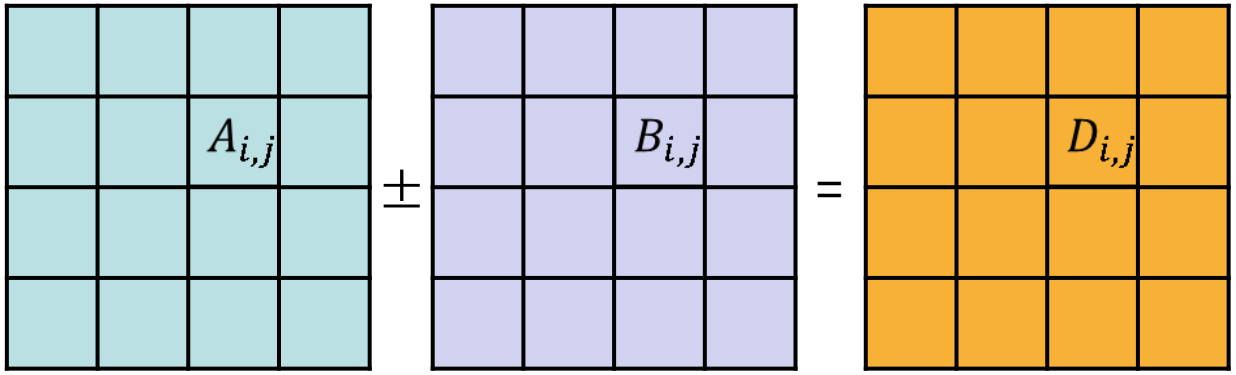
Матричные операции

Умножение




```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
    for(k=0;k<n;k++)
    {
      C[i][j] = C[i][j] + A[i][k]*B[k][j];
    }
```

Сумма




```
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  {
    D[i][j] = AS(A[i][j], B[i][j], sign);
  }
```