УДК 681.3.057: 518.5

БЕЗОШИБОЧНОЕ РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

А.В. Панюков, М.И. Германенко

В статье приведены теоретические и экспериментальные результаты по применению безошибочных вычислений для решения систем линейных алгебраических уравнений. В частности показано, что вычислительная битовая сложность решения систем линейных алгебраических уравнений с невырожденной матрицей не превышает $O(l^{7/2})$, а вычислительная сложность нахождения нормального псевдорешения системы линейных алгебраических уравнений не превышает $O(l^5 \log_2 l)$, где l - число бит требуемых для представления исходных данных. Для уменьшения времени, требуемого для решения данной задачи целесообразно использовать параллельные вычисления. Показано, что при этом осуществляется ускорение в N раз, где N - число компьютеров, на которых решается задача.

Ключевые слова: безошибочные дробно-рациональные вычисления, система линейных алгебраических уравнений, псевдообращение матриц, параллельные вычисления, вычислительная сложность.

Введение

Решение систем линейных алгебраических уравнений является одной из фундаментальных задач математики. В частности, она возникает при решении краевых задач для дифференциальных и интегральных уравнений, к которым сводятся реальные проблемы техники, физики, экономики, математики [1] и др.

Некоторые методы решения данной задачи, такие как метод Гаусса, метод Жордана-Гаусса, метод прогонки, прямое использование формул Крамера и др., определены в терминах точных вычислений. Но использование стандартных типов данных известных языков программирования, существенно сужает множество рациональных чисел, представимых без погрешности. Таким образом, арифметические операции приходится выполнять приближенно, что часто дает неудовлетворительные результаты решения задач.

Интересно заметить, что в последнее время теория и практика решения плохо обусловленных линейных систем развивается в направлении разработки алгоритмов, устойчивых к погрешностям округления промежуточных результатов [2]. Примерами таких методов являются: метод вращений, метод отражений и др. Они содержат операции извлечения квадратного корня, вычисление синуса, косинуса и прочих иррациональных функций, т.е. ориентированы на вычисления с приближенными числами. Методы, не ориентированные на безошибочные вычисления, как правило, не распознают случаи, когда система имеет бесконечное множество решений или не имеет их вообще, выдавая ошибочные ответы. При вычислениях с округлениями, возможно, что 1) не будет найдено ни одного подходящего решения, даже если оно имеется; 2) найдены корни при их отсутствии; 3) найдено только одно решение, при их бесконечном множестве. При безошибочных вычислениях все три случая легко идентифицировать.

Общеизвестные алгоритмы решения систем линейных алгебраических уравнений: метод *Гаусса,* метод Жордана-Гаусса, метод прогонки, - для преобразования данных используют только основные арифметические операции. Но на сегодняшний день нам не известно языков программирования, представляющих программисту целочисленные типы данных с более чем 64 двоичными разрядами. Однако, при использовании в указанных алгоритмах безошибочных вычислений будут исключены все возможные методические погрешности решения (так как все промежуточные операции будут выполняться точно, без округлений), останутся только погрешности, обусловленные неточностью исходных данных.

Использование популярных программ, таких как MS Excel и MathCAD, не ориентированных на безошибочные и символьные вычисления соответственно, приводит к получению неверного результата даже при решении систем линейных уравнений порядка 15. Например, при решении системы $Hx = He$, где

$$H=\left\{\frac{1}{i+j-1}\right\}_{i,j=1,\ldots n},\quad e=\{1\}_{i=1\ldots,n},$$

очевидно, что χ должен быть единичным вектором. Результаты, полученные при n=15, с использованием программ MS Excel и MathCAD, приведены на рис. 1. Следует отметить, что данные программы даже не выдают сообщения, что полученный результат может быть неверным [3,4].

Используя символьное программирование, японские программисты К.Ш. Тан, В.-Х. Стиб, И. Харди [5] разработали класс, позволяющий выполнять операции со сверхдлинными числами.

Возможность обеспечения безошибочных вычислений в программах пользователя дает библиотека GMP [6]. Она разработана под *операционные системы* Unix, Linux и подобные малопопулярные системы в нашей стране, ее использование требует компиляции и дополнительных знаний, что затрудняет использование данной библиотеки для рядового программиста. Стоит также отметить, что для объектов GMP не предоставляется возможность их использования в параллельных вычислениях.

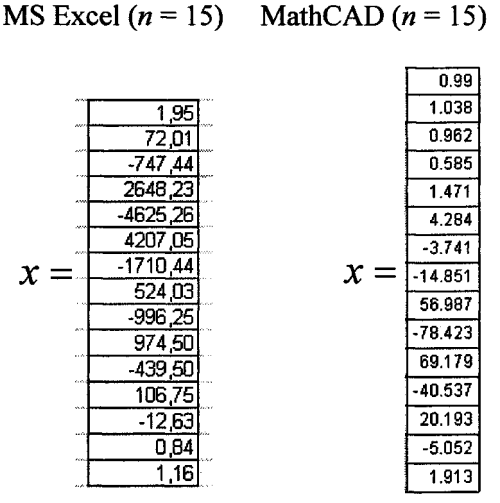

Рис. 1. Результаты эксперимента, при использовании программ MS Excel и MathCAD

1. Программное обеспечение выполнения безошибочных дробно-рациональных операций

Для обеспечения безошибочных дробно-рациональных вычислений были разработаны классы ***overlong*** [7] и ***rational*** [8]. Класс ***overlong***, который существенно расширяет *логические* возможности целочисленных вычислений: диапазон представимых чисел расширяется до $(-(2^{16})^{65536}$, $+(2^{16})^{65536})$. Таким образом, имеется возможность представлять целые числа, имеющие более 600 000 десятичных разрядов.

Ранее доказано [9], что битовая пространственная сложность результата арифметической операции $\circ\in\{+,-,/,\times\}$ над рациональными числами p, q не превосходит величины $L(p\circ q)\le L(p)+L(q)$. Проведенный анализ битовой вычислительной сложности, при выполнении алгоритма столбиком, показал что имеет место следующее неравенство $C(p\circ q)\le L(p)\cdot L(q)$. При использовании быстрого алгоритма умножения [10] вычислительная сложность операций с дробно-рациональными числами не будет превосходить значения

$$O\left(\left(L(p)+L(q)\right)\cdot\log_2\left(L(p)+L(q)\right)\right).$$

Класс ***rational,*** который дает потенциальную возможность использовать в программах пользователя безошибочное выполнение основных арифметических операций над полем рациональных чисел. По определению, тип данных ***rational*** представляет собой пару ***<nmr, dmr>***. Здесь *nmr* - целочисленная переменная типа *overlong,* обозначающая числитель, а *dmr* - целочисленная переменная типа *overlong,* обозначающая знаменатель. Минимальный шаг дискретизации представляемых чисел существенно лучше, чем у стандартных типов данных и равен $2^{-1048575}$. Проведенное практическое исследование, показало, что наиболее оптимальный метод сокращения - попарно сократить операнды до выполнения операции.

Пусть A - (nxm)-матрица, максимальную длину элемента матрицы A обозначим чрез M_A, тогда число цифр, в используемой системе счисления, достаточное для представления матрицы A равно

$$L(A)=\sum_{i=1..n,j=1..m} M_A=n\cdot m\cdot M_A.$$

Если A, B – $(n\times m)$-матрицы, тогда очевидно, что

$$L(A+B)\le L(A)+L(B),\quad C(A+B)\le n\cdot m\cdot(M_A+M_B)\cdot\log_2(M_A+M_B),\quad M_{A+B}\le(M_A+M_B).$$

Если A – $(n\times m)$-матрица, B – $(m\times l)$-матрица, тогда

$$L(A\cdot B)\le nml(M_A+M_B),$$

$$C(A\cdot B)\le O\left(nm^2l(M_A+M_B)^2\log_2(M_A+M_B)\right),\quad M_{A\cdot B}\le m\cdot(M_A+M_B).$$

Разработан класс *matrix* [4], который предназначен для облегчения программирования и улучшения визуального восприятия программ, использующих матричные вычисления. В данный класс встроены методы решения систем линейных уравнений с заданной матрицей, нахождения обратной матрицы и нахождения обобщенной обратной матрицы. Добавлена возможность использования параллельных вычислений.

2. Применение безошибочных вычислений для решения невырожденных линейных систем

Для безошибочного вычисления, как обратной матрицы A^{-1}, так и решения χ системы уравнений можно использовать метод *Жордана-Гаусса*. Фрагменты листинга его программной реализации приведены на рис. 2. Система хранится в матрице *m*, включая столбец свободных членов.

```
#define N 500
#include <stdio.h>
#include <iostream>
#include "rational.h"
using namespace std;
rational m[N][N+1];
int rowN[N];
void SubLine(int i, int j) {
   rational R,z;
   R=m[rowN[i]][j];
   for(int k=j; k < N+1; k++) {
      z=R*m[rowN[j]][k];
      m[rowN[i]][k]-=z;
   }
}
void DivLine(int i) {
   rational R;
   R=m[rowN[i]][i];
   for(int l=i ;l<N+1; l++)
   m[rowN[i]][l]/=R;
}

void Swap(int i, int j){
   k=rowN[i]; rowN[i]= rowN[j];
   rowN[j]=k;
}
```

```
int Solve() {
   rational zero;
   int i,j,k;
   // Инициализация rowN
   for (i=0; i<N; i++) rowN[i]=i;
   for (i=0; i<N; i++){
      j=i;
      // Поиск   ведущей строки
      while((m[rowN[j]][i] ==zero)
                &&(j<N)) j++;
      if (j==N) break;//Определитель0
      if (i!=j) Swap(i,j);
      //Нормализация ведущей строки
      DivLine(i);
      // Ведущее преобразование
     for (k=0;k<i;k++)SubLine(k,i);
     for(k=i+1;k<N;k++)SubLine(k,i);
    }
   return (j!=N);
}

int main(){ ReadFile();
   if (Solve()) WriteResult();
   else cout << "Det=0\n";
   return 0;
}
```

Рис. 2. Алгоритм решения систем уравнений методом Жордана-Гаусса

Для строк матрицы вводятся состояния: *открытая, закрытая* и *ведущая*. Открытыми строками будем называть строки с номером *j*, у которых на i-м шаге выполнения алгоритма модифицированный номер $rowN[j] \ge i$. Первый цикл в теле функции *Solve()* выполняет присваивания $rowN[j] = j$, что соответствует присваиванию всем строкам статуса *открытая*. Число выполнений тела второго цикла равно числу переменных. При *i-м* выполнении тела цикла из открытых строк выбирается строка y, имеющая ненулевой элемент в столбце i. Найденной строке присваивается статус ведущая, при необходимости производится модификация значений $rowN[j]$ и $rowN[i]$. Далее с помощью процедуры *DivLine()* производится нормировка ведущей строки, а с помощью процедуры *SubLine()* осуществляется ведущее преобразование других строк матрицы, состоящее в обнулении элементов i-го столбца, не принадлежащих ведущей строке. После этого ведущая строка переводится в состояние *закрытой*.

При использовании класса *rational* возможно исключить все методические погрешности. Известно, что алгоритм *Жордана-Гаусса* имеет алгебраическую пространственную сложность $O(n^2)$ и алгебраическую вычислительную сложность $O(n^3)$. Данные оценки позволяют определить количество переменных, требуемых для решения задачи и количество арифметических операций с этими переменными. При использовании безошибочных вычислений длина переменной зависит от представляемого ей значения, следовательно, количество переменных не позволяет оценить

ресурсы, необходимые для нахождения результата. Для практического использования алгоритма требуется определить количество бит, требуемых для нахождения результата, а также количество операций с битами. Ответ на данный вопрос дают теоремы изложенные ниже.

Теорема 1 [9]. Пусть $Ax = b$ - система линейных алгебраических уравнений, A - невырожденная матрица η χ η с рациональными элементами, b - n-мерный вектор с рациональными элементами,

$$m = \max\left\{ \max_{i,j=1,2,..,n} L(a_{ij}), \max_{i=1,2,..,n} L(b_i) \right\},$$

тогда

$$L(x) = \sum_{i=1}^{n} L(x_i) \le 2n^2(\log_2 n + (2n+1)m).$$

Теорема 2 [9]. Пусть даны невырожденная n × n система уравнений $Ax = b$ с рациональными коэффициентами, являющаяся приближением некоторой n × n системы уравнений $\tilde{A}\tilde{x} = \tilde{b}$; матрицы абсолютных погрешностей:

$$\Delta_A = (\delta_{ij}) : (\forall i, j = 1,2,...,n)\ \delta_{ij} \le |a_{ij} - \tilde{a}_{ij}|, \quad \Delta_b = (\delta_j) : (\forall j = 1,2,...,n)\ \delta_j \le |b_j - \tilde{b}_j|;$$

а также верхняя оценка числа бит, требуемых для одного элемента исходных данных

$$m = \max\left\{ \max_{i,j=1,2,..,n} \{L(a_{ij}), L(\delta_{ij})\}, \max_{i=1,2,..,n} \{L(b_i), L(\delta_i)\} \right\}.$$

Тогда, для нахождения как решения x, так и гарантированной нормы погрешности $\|\Delta_x\| = \|x - \tilde{x}\|$ *потребуется не более* $O(n^4 m)$ *бит памяти и не более* $O(n^7 m^2)$ *битовых операций.*

Пусть l – число бит, требуемых для представления исходных данных. При ограниченном m будем иметь $l = \Theta(n^2 m)$. Из теоремы 2 следует, что в этом случае зависимость битовой пространственной сложности от величины l не превосходит величины $O(l^2)$, а зависимость битовой вычислительной сложности (при использовании быстрых алгоритмов умножения) от величины l не превосходит $O(l^{7/2})$.

Если A – $(n \times n)$-матрица, то найти обратную матрицу или доказать ее отсутствие можно посредством редукции матрицы $(A|E)$, где E – единичная матрица, к форме $(E|B)$ с помощью метода Жордана–Гаусса. В редуцированной матрице $B = A^{-1}$. Пространственная и вычислительная сложность такой редукции [9] не превосходит $O(l^2)$ и $O(l^{7/2})$ соответственно.

3. Применение безошибочных вычислений для вычисления нормального псевдорешения линейных систем

При решении практических задач система уравнений может быть недоопределенной, переопределенной и несовместной. В этом случае за решение системы принимается ее нормальное псевдорешение

$$x^* = \arg\min\left\{ \|\overline{x}\| \mid \overline{x} \in \operatorname{Arg}\min_{x \in R} \|Ax - b\| \right\}.$$

Для его нахождения можно использовать обобщенную обратную матрицу A^+, называемую также матрицей Мура-Пенроуза. В этом случае $x^* = A^+ b$.

Известно множество методов нахождения матрицы Мура-Пенроуза. Теоретический расчет битовой сложности алгоритмов и практические эксперименты [4] показали, что наиболее оптимальным методом для вычисления обобщенной обратной матрицы является метод Эрмита [12].

Алгоритм Эрмита

Вход: A - исходная (nxm)-матрица системы;

Выход: A^+ - матрицапсевдобратная к A;

Шаг 1. Положить $C = A \cdot A^T$, $B = C \cdot C$.

Шаг 2. Методом Жордана-Гаусса провести редукцию матрицы $(B \mid E)$, где E - единичная матрица, к форме

$$\left(\begin{array}{cc|c} E_r & R & \\ 0 & 0 & P \end{array} \right),$$

где E_r - единичная матрица ранга r .

Шаг 3. Методом Жордана-Гаусса провести редукцию матрицы

$$\left(\begin{array}{cc|c} E_r & 0 & \\ R^T & 0 & E \end{array}\right),$$

где E - единичная матрица, к форме

$$\left(\begin{array}{cc|c} E_r & 0 & \\ 0 & 0 & Q \end{array}\right).$$

Шаг 4. Вычислить матрицу

$$A^+ = A^T \cdot P \cdot Q \cdot C.$$

Шаг 5. Вернуть A^+.

Конец алгоритма.

Теорема 3 [4]. *Пусть l - число бит, требуемых для кодирования всех элементов исходной матрицы A. Тогда, для нахождения обобщенной обратной матрицы Мура-Пенроуза алгоритмом Эрмита потребуется не более* $O(l^4)$ *бит памяти и не* $O(l^5 \log_2 l)$ *операций с ними.*

Многие реальные задачи имеют большую размерность *n,* следовательно, вычислительная и пространственная сложность этих задач будет достаточно большой, поэтому и время, требуемое для решения, будет большим. Увеличить скорость решения во многих случаях позволяет использование параллельных вычислений на нескольких компьютерах одновременно, при этом сократятся как количество операций, выполняемых на одном компьютере, так и память, требуемая для хранения промежуточных результатов.

4. Параллельные вычисления при решении линейных систем

Для увеличения скорости решения реальных задач было принято решение адаптировать разработанные классы ***overlong, rational*** и ***matrix*** к параллельным вычислениям. При организации параллельных вычислений было принято использовать MPICH, которая поддерживает стандарт MPI и имеет GNU лицензию [15].

Передача класса *rational* между узлами внутренними средствами MPI невозможна, так как данный класс содержит два объекта класса *overlong,* которые в свою очередь содержат длину массива и указатель на массив содержащий число. Поэтому объявить структуру ***rational*** по стандарту MPI не возможно по двум причинам:

- необходимо хранить указатель;
- каждый объект ***rational*** может иметь произвольную длину, которая может измениться при следующей математической операции - невозможно создать универсальную структуру.

Очевидными решениями будет упаковка ***rational*** в буфер с целью дальнейшей передачи. Данный вариант позволяет обойтись одной транзакцией, и поэтому взят за основу модификации. Для передачи типов ***overlong, rational*** были переопределены стандартные методы передачи данных в среде MPI [3].

Из описания последовательного алгоритма (рис. 2) следует, что следует распараллелить процесс инициализации, поиск ведущей строки и ведущее преобразование. Из этого следует, что наиболее предпочтительной является декомпозиция исходных данных по строкам, т.е. разрезание матрицы системы на горизонтальные полосы, содержащие примерно равное число строк (рис. 3). При этом, каждая полоса загружается в соответствующий компьютер: нулевая полоса - в нулевой компьютер, первая полоса - в первый компьютер, и т. д., последняя полоса - в последний компьютер.

В начале параллельной реализации производим инициализацию переменных и выделение памяти под обрабатываемый локальным процессом фрагмент матрицы системы, ведущую строку, массив модифицированных номеров, массив флагов открытых строк и буфер передачи данных. Внешний цикл функции *Solve()* выполняется для каждой переменной системы уравнений. При i-м выполнении тела цикла в каждом процессе из открытых строк выбирается строка *j,* имеющая максимальный по абсолютной величине элемент в столбце *i* локального фрагмента матрицы системы.

Затем определяется ведущий процесс, в котором находится максимальный по модулю элемент столбца i. Если модуль максимального элемента равен 0, то выполнение программы прекращается с выводом сообщения *«Det = 0»*. В противном случае строке j ведущего процесса присваивается статус *«ведущая»* и устанавливается модифицированное значение ***rowN[j] = i***.

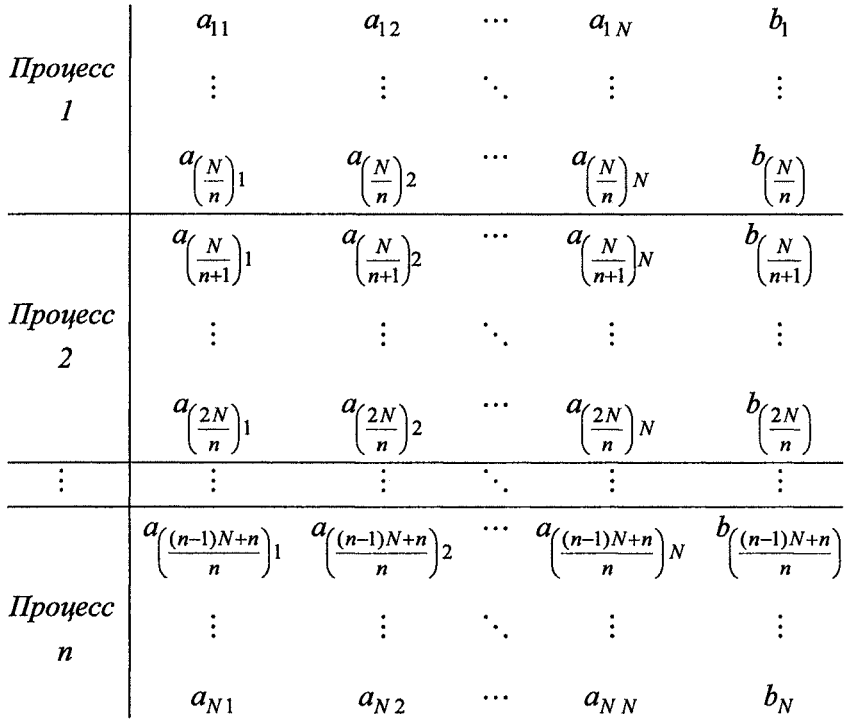

Рис. 3. Декомпозиция данных

Далее в ведущем процессе производится нормировка ведущей строки и ее рассылку остальным процессам. Функция *SubLine()* осуществляется ведущее преобразование строк матрицы, состоящее в обнулении элементов i-го столбца, не принадлежащих ведущей строке. После этого ведущая строка переводится в состояние *закрытой*.

Если исходными данными является *nxn* матрица, *m* - количество бит, требуемых для представления одного элемента исходной матрицы, и задача решается на N компьютерах, тогда ускорение параллельной реализации алгоритма *Жордана-Гаусса* при использовании параллельных вычислений составит [3]:

$$\frac{N}{1+\frac{CN}{m^2n^2}\left(\frac{\upsilon_{процессора}}{\upsilon_{передачи}}\right)}.$$

Если исходными данными является n_1x n_2 матрица, *m* - количество бит, требуемых для представления одного элемента исходной матрицы, тогда коэффициент ускорения параллельной реализации метода Эрмита, выполняемого на N компьютерах не будет превышать ускорения на участке с наибольшей вычислительной сложностью [4]:

$$\frac{N}{1+\frac{CN}{n_2^5m}\left(\frac{\upsilon_{процессора}}{\upsilon_{передачи}}\right)}.$$

5. Вычислительный эксперимент

Вычислительный эксперимент проводился на кластере кафедры ЭММиС ЮУрГУ. Данный кластер состоит из основного и восьми вспомогательных узлов, объединенных в локальную сеть посредством коммутатора Allied Telesyn AT-FS716E 100Base-TX. Параметры узлов сети представлены в таблицах 1 и 2.

Таблица 1.

Конфигурация вспомогательного узла

Процессор	P4 Intel® Celeron® 2,40 ГГц, 256 kb cache (FSB533МГц)
ОЗУ	PC3200 DDR 512Mb
Жесткий диск	Seagate IDE 80Gb (7200 rpm)
Сетевая карта	RealTek RTL8139 100 M

Таблица 2

Конфигурация основного узла

Процессор	P4 Intel® Pentium® Core 2 Duo 2,80 Ггц, cache 1 Mb
ОЗУ	PC3200 DDR 512Mb
Жесткий диск	Seagate SATA-150 160Gb (7200 rpm)
Сетевая карта	Intel(R) PRO/1000

Примерная расчетная пиковая производительность кластера в соответствии с данными, взятыми с сайта производителя, составляет величину ПРППК = 8·2,4·2+2·2,8·2 = 49,6 Gflops. На узлах кластера присутствуют только средства необходимые для функционирования среды ***MPI.*** На основном узле установлены оконные менеджеры *XFCE4* и *GNOME,* а также другие пакеты необходимые для написания, отладки и запуска программ.

Для апробации разработанного программного обеспечения и анализа практической достижимости полученных оценок вычислительной сложности был проведен вычислительный эксперимент. Эксперимент состоял в решении систем линейных алгебраических уравнений $Ax = b$, где $A = \left[1/(i+j+1)\right]_{i,j=0}^{n}$ - матрица Гильберта, $b = [1]_{i=0}^{n}$.

Результаты вычислительного эксперимента приведены на рис. 4.

Проведенные вычислительные эксперименты подтвердил теоретически рассчитанные коэффициенты ускорения (при достаточно больших объемах исходных данных скорость вычисления увеличится в N раз, где N - количество компьютеров, на которых решается задача).

Результаты аналитического исследования и проведенного вычислительного эксперимента показывают, что использование параллельного программирования существенно уменьшает время, требуемое для безошибочного решения систем линейных уравнений.

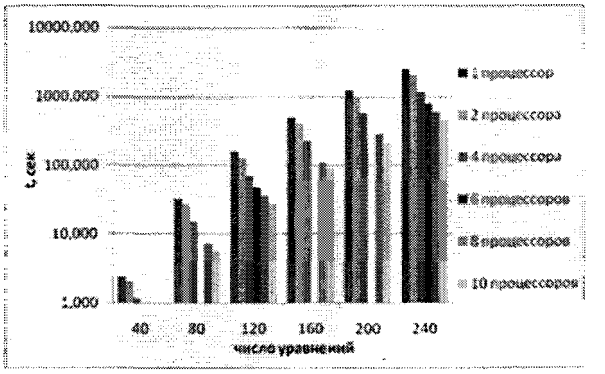

Рис. 4. Время решения систем уравнений на разных количествах процессоров

Заключение

1. Разработанные классы ***overlong, rational*** и ***matrix*** позволяют решать плохо обусловленные невырожденные системы линейных алгебраических уравнений точно за время не более $O(l^{7/2})$. При решении систем с приближенными данными имеется возможность оценки погрешности полученного решения.
2. Применение вычислений без округления позволяют использовать алгоритм Эрмита для вычисления псевдообратной матрицы за время не более $O(l^{5}\log_2 l)$. Это позволяет строить устойчивые алгоритмы решения линейных некорректно поставленных задач.
3. Проведенные теоретические исследования и практические эксперименты показывают возможность и необходимость безошибочного решения систем уравнений. Возможность значительного увеличения скорости при использовании безошибочных вычислений дает использование параллельных вычислительных технологий.

Литература

1. Вержбицкий, В.М. Численные методы (линейная алгебра и нелинейные уравнения): учеб. пособие для вузов / В.М. Вержбицкий - М.: Высшая школа, 2000.

2. Воеводин, В.В. Ошибки округлений и устойчивость в прямых методах линейной алгебры / В. В. Воеводин - М.: Наука, 1969.

3. Панюков, А.В. Распараллеливание алгоритмов решения систем линейных алгебраических уравнений с применением вычислений без округлений / А.В. Панюков, М.И. Германенко, В.В. Горбик // Параллельные вычислительные технологии (ПаВТ'2007) / г. Челябинск, 29 января - 2 февраля 2007 г. - Т. 2. - С. 238-249.

4. Панюков, А.В. Параллельные алгоритмы безошибочного вычисления матрицы Мура-Пенроуза / А.В. Панюков, М.И. Германенко // Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (Санкт-Петербург, 28 января - 1 февраля 2008 г.). - С. 215-223.

5. Тан, К.Ш. Символьный С++: Введение в компьютерную алгебру с использованием объектно-ориентированного программирования / К.Ш. Тан, В.-Х. Стиб, Й. Харда. - М.: Мир, 2001. - 662 с.

6. Официальный сайт библиотеки GMP (http://www.gmplib.org).

7. Свидетельство РосАПО № 990486 «Класс overlong» / А.В. Панюков, М.М. Силаев.

8. Свидетельство РосАПО № 990607 «Класс rational» / А.В. Панюков, М.М. Силаев, М.И. Германенко.

9. Панюков, А.В. Сложность нахождения гарантированной оценки решения приближенно заданной системы линейных алгебраических уравнений / А.В. Панюков, М.И. Германенко // Изв. Челябинского научного центра. - 2000 - № 4(9). - С. 13-17. - http://www.sci.urc.ac.ru/news/2000_3

10. Кнут, Д. Искусство программирования для ЭВМ. Т. 2: Получисленные алгоритмы / Д. Кнут; пер. с англ. - М.: Мир, 1977.

И.Максимов, В.П. Арифметика рациональных чисел и компьютерное исследование интегральных уравнений / В.П. Максимов - Соросовский образовательный журнал. - 1999. - № 3.

12. Люстерник, Л.А. Элементы функционального анализа / Л.А. Люстерник, В.И. Соболев. - М.: Наука.-1965.

13.Шпаковский, Г.И. Программирование для многопроцессорных систем в стандарте MPI / Г.И. Шпаковский, Н.В. Серикова. - Минск: БГУ, 2002.

Поступила в редакцию 7 июля 2008 г.

EXACT SOLVING OF A LINEAR EQUATIONS SET

Theoretical and experimental results of application of exact computation for solving of linear algebraic equations are presented in the paper. In particular it is demonstrated that computational bit complexity of solving of a linear algebraic equations set with non-degenerate matrix are not exceeding $O(l^{7/2})$, and computational bit complexity of computing of normal pseudo solution are not exceeding $O(l^5 \log_2 l)$, where l is bit volume of input data. For computational speedup it is reasonably to use multiprocessing. It is illustrated that computational speedup for considered problems under of exact computation equal to number of processors.

Keywords: exact rational computation, linear algebraic equations set, pseudoinversion of matrix; parallel computation, computational complexity.

Panyukov Anatoly Vasilyievich - Dr. Sc. (Physics and Mathematics), Professor, Head of the Department «Economic-mathematical methods and statistics», South Ural State University.

Панюков Анатолий Васильевич - доктор физико-математических наук, профессор, заведующий кафедрой «Экономико-математические методы и статистика», Южно-Уральский государственный университет.

e-mail: pav@susu.ac.ru

Germanenko Maxim Igorevich - Post-Graduate Student, Economic-Mathematical Methods and Statistics Department, South Ural State University.

Германенко Максим Игоревич - аспирант кафедры «Экономико-математические методы и статистика», Южно-Уральский государственный университет.