Программное обеспечение безошибочных дробно-рациональных вычислений и его применение для решения линейных систем

М.И. Германенко

В работе приведены теоретические и экспериментальные результаты по применению безошибочных вычислений для нахождения нормального псевдорешения систем линейных алгебраических уравнений. Для ускорения времени, требуемого для решения данной задачи целесообразно использовать параллельные вычисления. Показано, что полученное ускорение имеет порядок N раз, где N – число компьютеров, на которых решается задача.

1. 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

$$H = \left\{ \frac{1}{i+j+1} \right\}_{i,j=0,\ldots n}, e = \{1\}_{i=0\ldots,n}.$$

очевидно, что x должен быть единичным вектором. Результаты, полученные при n=15, с использованием программ MS Excel и MathCAD, приведены на рис. 1. Кроме того, программы

даже не выдают сообщения, что полученный результат может быть неверным [3,4]. Используя символьное программирование, японские программисты К.Ш.Тан, В.-Х.Стиб, Й.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.

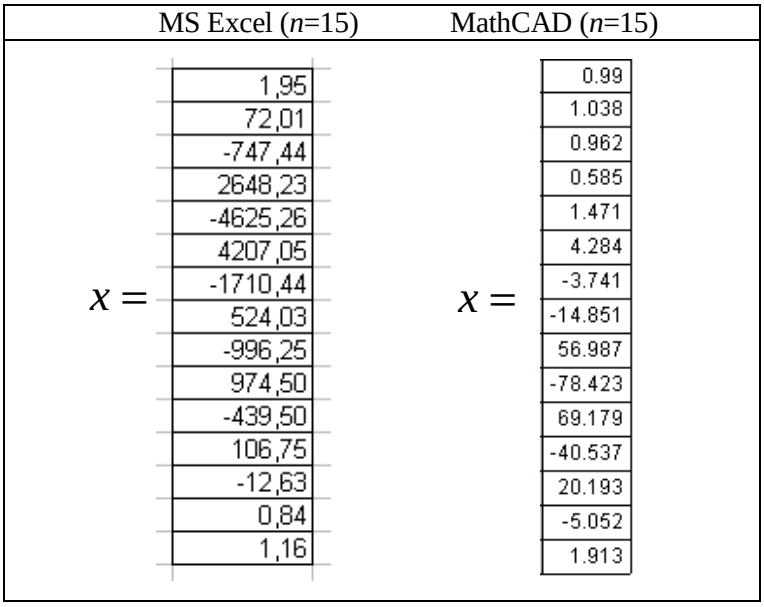

Рис. 1. Результаты эксперимента, при использовании программ MS Excel и MathCAD

В работе приведены теоретические и экспериментальные результаты по применению безошибочных вычислений для нахождения нормального псевдорешения систем линейных алгебраических уравнений. Для ускорения времени, требуемого для решения данной задачи целесообразно использовать параллельные вычисления. Показано, что полученное ускорение имеет порядок N раз, где N – число компьютеров, на которых решается задача.

2. 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 Тоома-Кука [10] вычислительная сложность умножения не будет превосходить значения $O\left(\max(L(p) \cdot L(q))^{1+3,5/\sqrt{\lg n}}\right)$.

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.

Для использования классов ***overlong*** и ***rational*** достаточно подключить заголовочный файл *ExactComp.h* в свою программу, после этого можно пользоваться объектами данных классов как объектами стандартных типов данных. Над объектами классов определены все основные арифметические операции, бинарные отношения, операторы ввода/вывода. Объем памяти, требуемый для представления объекта, зависит от значения представляемого числа.

В работе [4] показано, что битовая пространственная и вычислительная сложности сложения $n \times m$ матрицы A и B не превосходят величин

$$L(A+B) \le L(A)+L(B),\ C(A+B) \le nm(M_A + M_B)$$

соответственно, а битовая пространственная и вычислительная сложности умножения $n \times m$ матрица A и $m \times l$ матрица B не превосходят величин

$$L(A \cdot B) \le nml(M_A + M_B),\ C(A \cdot B) \le O\left(nm^2 lC_*(A,B)\right)$$

соответственно, где C_* – битовая вычислительная сложность выбранного метода умножения. Здесь M_A – число бит требуемых для представления матрицы A.

Разработанный класс ***matrix*** [4], предназначен для облегчения программирования, а также улучшений визуального восприятия программ, использующих матричные вычисления. Память, требуемая для представления переменных-матриц, автоматически выделяется, а ненужная – удаляется. В данный класс встроены:

- основные арифметические операции с матрицами;
- методы решения систем линейных уравнений с заданной матрицей;
- нахождения обратной матрицы;
- нахождения обобщенной обратной матрицы.

В класс добавлена возможность использования параллельных вычислений. Для использования возможностей данного класса достаточно подключить его к программе пользователя.

3. Применение безошибочных вычислений для решения линейных систем

Для безошибочного вычисления, как обратной матрицы A^{-1}, так и решения x системы уравнений можно использовать метод *Жордана-Гаусса.* 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.

Лемма 1 [9]. Пусть $A = (a_{ij})$ – невырожденная целочисленная матрица $n \times n$, $m = \max\limits_{i,j=1,2,..,n} L(a_{ij})$. Тогда

$$L(\det A) \le n(\log_2 n + m).$$

Лемма 2 [9]. Пусть $A = (a_{ij})$ – невырожденная $n \times n$ рациональна матрица, $m = \max\limits_{i,j=1,2,..,n} L(a_{ij})$. Тогда

$$L(\det A) \le n(\log_2 n + (2n+1)m).$$

Теорема 1 [13]. *Пусть $Ax = b$ – система линейных алгебраических уравнений, A – невырожденная матрица $n \times n$ 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

Теорема 2 [13]. *Пусть даны невырожденная $n \times n$ система уравнений $Ax = b$ с рациональными коэффициентами, являющаяся приближением некоторой $n \times n$ системы уравнений $\tilde{A}\tilde{x} = \tilde{b}$; матрицы абсолютных погрешностей:*

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

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.

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

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 $O(l^{5/2})$.

Численные методы решения краевых задач для линейных дифференциальных и интегральных уравнений сводятся к решению систем линейных алгебраических уравнений с $(2\kappa+1)$-диагональной матрицей. Для решения таких систем используется метод прогонки. С одной стороны, данный метод имеет низкую алгебраическую вычислительную сложность (т. е. кол-во операций с числами) равна $O(n)$, где n – порядок системы. С другой стороны он не позволяет решать плохо обусловленные системы. Для устранения последнего можно использовать безошибочные вычисления. Однако, при безошибочных вычислениях, как и в случае алгоритма Жордана-Гаусса, адекватной оценкой сложности будет не алгебраическая, а битовая вычислительная сложность (т. е. кол-во операций с битами).

Теорема 3. Пусть даны невырожденная система уравнений $Ax = b$, A – трехдиагональная $n \times n$ матрица с рациональными коэффициентами, b – $n \times k$ матрица свободных членов, пусть также дана 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 решения потребуется не более

$$O\left(\frac{mn^3}{3}(5k+1) \right)$$

бит памяти и не более

$$O\left(k \frac{m^3 n^6}{3} \right)$$

операций с ними.

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

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

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 $(n\times m)$-матрица системы;

Выход: A^+ – матрица псевдобратная 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

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

где E_r – единичная матрица ранга r.

Шаг 3. Методом Жордана-Гаусса провести редукцию матрицы $\left(\begin{array}{cc|c} E_r & 0 & E \\ R^T & 0 & \end{array}\right)$, где E – единичная матрица, к форме

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

Шаг 4. Вычислить матрицу

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

Шаг 5. Вернуть A^+.

Фрагмент программной реализации листинга программы приведен на рис. 2.

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

Для апробации разработанного программного обеспечения и анализа практической достижимости полученных оценок вычислительной сложности был проведен второй вычислительный эксперимент. Он состоял в решении систем линейных алгебраических уравнений

$$Hx = e, \quad H = \left\{ h_{ij} = \frac{1}{i+j-1} \right\}_{i,j=1}^{n}, \quad e = \{e_i = 1\}_{i=1}^{n}.$$

Матрица H, известная как матрица Гильберта, является плохообусловленной. Подобные системы, с использованием приближенных вычислений, не удается решить уже при $n \geq 5$. Были решены все системы линейных уравнений, для n = 3, …, 250. Расчеты осуществлялись по методу Жордана-Гаусса, методу Гаусса и методу прогонки.

Для нахождения необходимого числа слов памяти, требуемых для решения поставленной задачи, были внесены некоторые изменения в класс ***overlong*** и в саму программу решения систем. В программу решения систем линейных алгебраических уравнений были заведены глобальные переменные *mem* и *maxmem*. В начале вычисления эти переменные равны 0. При выделении новой памяти в классе ***overlong*** значение переменной *mem* ее увеличивается на количество выделенной памяти, а при освобождении значение переменной *mem* уменьшается на количество освобожденной памяти. Если после выделения памяти ***mem*** > ***maxmem***, то ***maxmem*** присваивается значение ***mem***.

При решении системы методом прогонки также использовалась матрица Гильберта с ненулевыми элементами только на трех главных диагоналях. Результаты вычислительных экспериментов приведены на рис. 3 и рис. 4. Проведенные практические эксперименты являются подтверждением теоретических исследований, а также показывают не только возможность, но и необходимость безошибочного решения систем линейных алгебраических уравнений. Необходимость вызвана тем, что при приближенном решении конечный результат будет существенно отличаться от правильного, но этого можно избежать, если выполнять все операции точно.

```
// нахождение обобщенной обратной матрицы с помощью метода Эрмита
Matrix ErmitMethod(Matrix A){
  Matrix E(A.GetM(), A.GetM()), P(A.GetM(), A.GetM());
  Matrix Rt (A.GetM(), A.GetM()), B, C, Q(A.GetM(), A.GetM());
  // построим квадратную матрицу
  C = (A*A.T()); B = C*C;
  // составляем единичную матрицу
  E.Fill("E");
  // добовляем справа к матрице A единичную матрицу E
  B.AddMtx(E, true);
  // выполняем редукцию методом Ж-Г
  B.JGMethod();
  // разделяем матрицу M на две, R - левая часть, Q - правая
  for(int i=0; i<M.GetM(); i++)
    for(int j=0; j<M.GetN(); j++){
    // сразу транспонируем
    Rt(i,j) = M(j,i);
    P(i,j) = M(i,j+E.GetN());
  }
  Rt.AddMtx(E, true);
  Rt.JGMethod();

  for(int i=0; i<M.GetM(); i++)
    for(int j=0; j<M.GetN(); j++) Q(j,i) = Rt(i,j+Rt.GetY());
  // выполнение 4го шага алгоритма
  A_ = (A.T() * Q) * (P * C);
  return A_;
}
```

Рис. 2. Листинг программы метода Эрмита

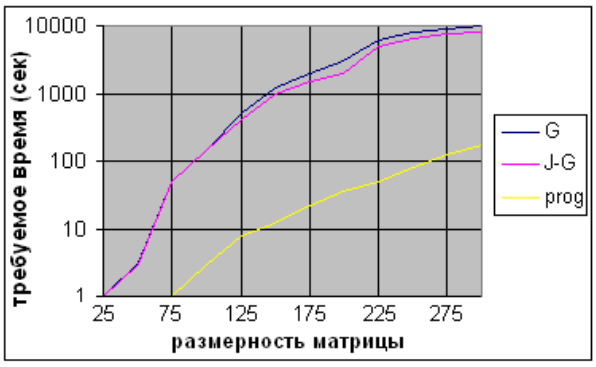

Рис. 3. Время (секунды) решения системы методами *Гаусса*, *Жордана-Гаусса* и *прогонки*

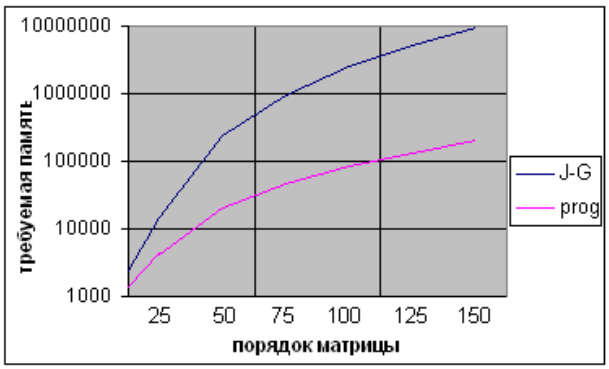

Рис. 4. Память (число слов памяти), требуемая для решения системы

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. Программное обеспечение параллельных вычислений 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.

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***.

Наиболее предпочтительной является декомпозиция исходных данных по строкам, т.е. разрезание матрицы системы на горизонтальные полосы, содержащие примерно равное число строк (рис. 5). 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*. Далее в ведущем процессе производится нормировка ведущей строки и ее рассылка 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 $n \times n$ 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 параллельный вычислений составит [3]:

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

Процесс 1	a_{11}	a_{12}	$\cdots$	a_{1N}	b_1
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	$a_{\left(\frac{N}{n}\right)1}$	$a_{\left(\frac{N}{n}\right)2}$	$\cdots$	$a_{\left(\frac{N}{n}\right)N}$	$b_{\left(\frac{N}{n}\right)}$
Процесс 2	$a_{\left(\frac{N}{n+1}\right)1}$	$a_{\left(\frac{N}{n+1}\right)2}$	$\cdots$	$a_{\left(\frac{N}{n+1}\right)N}$	$b_{\left(\frac{N}{n+1}\right)}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	$a_{\left(\frac{2N}{n}\right)1}$	$a_{\left(\frac{2N}{n}\right)2}$	$\cdots$	$a_{\left(\frac{2N}{n}\right)N}$	$b_{\left(\frac{2N}{n}\right)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Процесс n	$a_{\left(\frac{(n-1)N+n}{n}\right)1}$	$a_{\left(\frac{(n-1)N+n}{n}\right)2}$	$\cdots$	$a_{\left(\frac{(n-1)N+n}{n}\right)N}$	$b_{\left(\frac{(n-1)N+n}{n}\right)}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	a_{N1}	a_{N2}	$\cdots$	a_{NN}	b_N

Рис. 5. Декомпозиция данных

Если исходными данными является $n_1 \times n_2$ 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]:

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

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. 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. 6. Проведенные вычислительные эксперименты подтвердили теоретически рассчитанные коэффициенты ускорения (при достаточно больших 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.

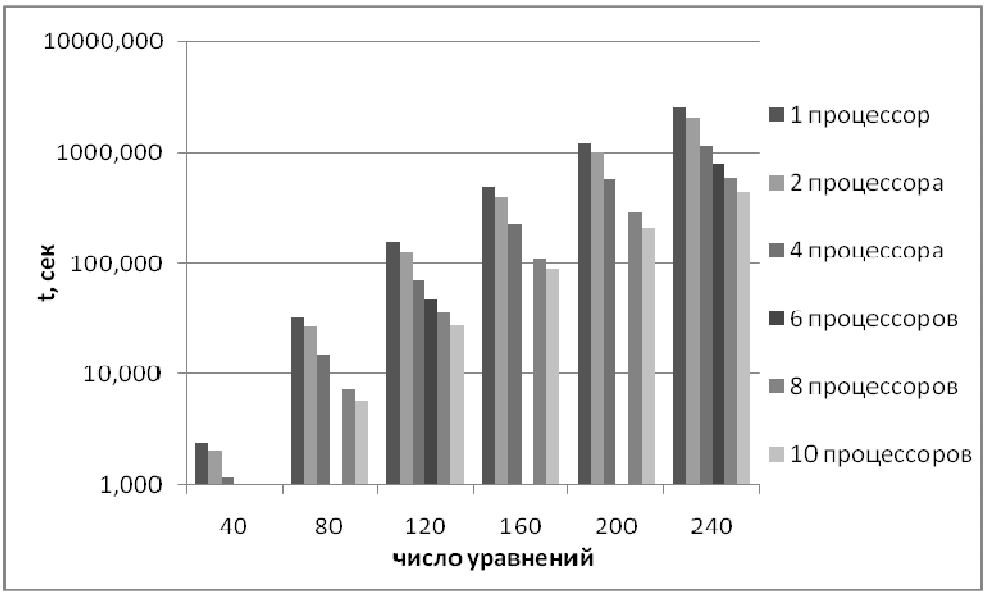

Рис. 6. Время решения систем уравнений на разных количествах процессоров

6. Заключение

1. Разработанные классы ***overlong***, ***rational*** и ***matrix*** позволяют решать плохо обусловленные невырожденные системы линейных алгебраических уравнений точно за время не более $O(l^{5/2})$. 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 $O(l^5)$. 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, а также возможность значительного увеличения скорости при использовании безошибочных вычислений дает использование параллельных вычислительных технологий. Проведенные практические исследования показали, что при увеличении числа процессоров, эффективность использования вычислительных ресурсов не уменьшается.

Литература

1. Вержбицкий, В.М. Численные методы (линейная алгебра и нелинейные уравнения): Учеб. Пособие для вузов / В.М. Вержбицкий – М.: Высш. шк., 2000. – 266 с.: ил.
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. – Том 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 C++: Введение в компьютерную алгебру с использованием объектно-ориентрованного программирования / К.Ш. Тан, В.-Х. Стиб, Й. Харди – М.: Мир, 2001. – 662 с.
6. Официальный сайт библиотеки GMP (http://www.gmplib.org).

7. Панюков А. В., Силаев М. М., Германенко М. И. Класс rational. // Программы для ЭВМ. Базы данных. Топологии интегральных микросхем. - Официальный бюллетень Российского агентства по патентам и товарным знакам. №4(29), 1999 г. М.: ФИПС. - 1999. - Рег. № 990607. - С. 97.
8. Панюков А. В., Силаев М. М. Класс overlong. // Программы для ЭВМ. Базы данных. Топологии интегральных микросхем. - Официальный бюллетень Российского агентства по патентам и товарным знакам. №4(29), 1999 г. М.: ФИПС. - 1999. - Рег. № 990489. - С. 17.
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. – 724 с.
11. 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. Германенко, М. И. Аналитическое и экспериментальное исследование сложности безошибочного решения систем линейных алгебраических уравнений / Германенко М.И. // Тезисы второго Международного конгресса студентов, молодых ученых и специалистов «Молодежь и наука – третье тысячелетие» / YSTM`02 (Москва, 15-19 апреля 2002 г.) -М.: НТА «АПФН», 2002 г. - Часть 2. – С. 11-12.
14. Шпаковский, Г.И. Программирование для многопроцессорных систем в стандарте MPI / Г.И. Шпаковский, Н.В. Серикова – Минск: БГУ, 2002.