УДК 519.852:519.876

В.И. КУДИН, С.И. ЛЯШКО, В.В. ОНОЦКИЙ

О ТЕХНОЛОГИИ ДЛИННОЙ АРИФМЕТИКИ ПРИ ПОСТРОЕНИИ АЛГОРИТМОВ ИССЛЕДОВАНИЯ ЛИНЕЙНЫХ СИСТЕМ

Предложен новый алгоритм и компьютерная реализация метода базисных матриц в среде Visual C++ с использованием метода Штрассена умножения длинных чисел и быстрого преобразования Фурье. Приведены результаты вычислительного эксперимента, в котором тестовые модели систем генерировались на основе матриц Гильберта разной размерности.

Для анализа свойств линейных систем уравнений (СЛАУ) и неравенств (СЛАН) с квадратной матрицей ограничений разработан ряд точных методов [1–4]. Известно, что по своим структурным свойствам модели СЛАУ относятся к некорректным. Одним из проявлений некорректности является свойство плохой обусловленности [1–3]. Для тестирования известных схем решения данного класса задач разработан ряд библиотек моделей систем с плохо обусловленными матрицами ограничений. В частности, к таким матрицам принадлежит матрица Гильберта. В работе [4] приведены основные соотношения, использованные при построении метода базисных матриц (МБМ). На основе формул связи элементов метода в смежных решениях предложена итерационная процедура нахождения величины ранга матрицы ограничений. Получено условие единственности решений и разработан алгоритм исследования плохо обусловленных СЛАУ и СЛАН на основе технологии длинной арифметики.

В данной работе приведены новые результаты об использовании усовершенствованных точных вычислений, которые базируются на быстрых методах умножения и деления длинных чисел. В частности, разработана алгоритмическая схема проведения вычислений по схеме МБМ с использованием технологии длинной арифметики для моделей с рациональными элементами, которая реализована на языке C++ в виде программного комплекса.

Постановка задачи. Рассмотрим СЛАУ вида

$$Au = C, \tag{1}$$

где матрица A размерностью $(m \times m)$, $C = (c_1, c_2, \ldots, c_m)^T$, $u = (u_1, u_2, \ldots, u_m)^T$ – векторы размерностью m.

Исследуем свойства компьютерной реализации алгоритма метода базисных матриц [4] для модели (1) в среде Visual C++ с использованием умножения длинных чисел методом Штрассена, который базируется на быстром преобразовании Фурье.

Концепция представления целых чисел

В современных ЭВМ, как правило, используются стандартные типы целых чисел, размер которых не превышает 64 байта. Преодоление такого аппаратного ограничения можно программным путем – разработкой собственного типа данных. Примером является класс longrat [4]. В разработанной на языке C++ библиотеке реализованы типы длинных целых чисел longint2 и соответствующих рациональных чисел longrat3 с быстрыми операциями умножения, деления, построенных на современных алгоритмах. Для реализации длинных целых чисел произвольного размера и соответствующих рациональных чисел в C++ был использован *объектно-ориентированный подход* (ООП). Такой подход состоит в сопоставлении реальному объекту предметной области так называемого *объектного типа* или *класса* объектов, что является обобщением структурного типа [6].

Определение 1. *Класс* в C++ – это программная конструкция, которая состоит из данных (*элементов данных* или *полей*) и подпрограмм, которые оперируют над этими полями и описывают свойства соответствующего объекта предметной области, которая моделируется.

В данной работе целое число произвольного размера запрограммировано в виде класса longint2. Рациональное число как пара (числитель класса longint2, знаменатель класса longint2) запрограммированное в виде класса longrat3. Целое число хранится в динамическом массиве, максимальная длина которого 4096. Элементами вектора есть 16-битные беззнаковые числа стандартного типа C++ *unsigned __int*16 и есть, фактически, ”цифрами” в системе исчисления с основой $BASE = 2^{16}$. Так, например, целое число из диапазона от 2^{32} до 2^{64}–1 будет храниться в массиве размером 3.

Массив ”цифр”, знак числа и операции над числами такой структуры оформлены в виде класса longint2. В частности, в классе longint2 реализованы такие операции с целыми числами: сложение, вычитание, умножение, деление с остатком, сравнение, конвертация в строку символов типа char.

Объявление класса longint2

```
class longint2
{
      public:
      unsigned __int16 *s;
      unsigned int l;
      longint2();
      longint2(unsigned n);
      longint2(unsigned __int16 *p,int n);
      longint2(const char *s1);
      longint2(const char *s1,int n);
      longint2(const longint2 &a);
      longint2 operator=(const longint2 &a);
      ~longint2();
      char* text(void);
      friend longint2 operator+(longint2 &a,longint2 &b);
      friend longint2 operator-(longint2 &a,longint2 &b);
      friend longint2 operator*(longint2 &a,longint2 &b);
      friend longint2 operator/(longint2 &a,longint2 &b);
      friend  ostream&  operator<<(ostream  &os,const  longint2
      &a);
      friend istream& operator>>(istream &is,longint &a);
      friend    longint2    karatsuba2(longint2    &u1,longint2
      &u2,unsigned int n);
      friend longint2 tomakuka(longint2 &u1,longint2 &u2);
      friend longint2 fastmul2(longint2 &A, longint2 &B);
      friend __int8 divmod(longint2 &A, longint2 &B, longint2
      &Q, longint2 &R) ;
      void mult2(int n);
      void mult(int m);
      void div(int m);
      friend int longcmp2(longint2 &a,longint2 &b);
      unsigned int toint();
      void print();
      void operator+=(longint2 &b);
      void operator-=(longint2 &b);
      void operator*=(longint2 &b);
      friend void savebint2(FILE *f,longint2& a);
      friend void loadbint2(FILE *f,longint2& a);
      friend longint2 ncd(longint2 a,longint2 b);
```

Представление рациональных чисел

Рациональное число представляется как пара: числитель и знаменатель типа longint2 и оформлено в виде класса longrat3. В данном классе также реализованы операции сложения, вычитания, умножения, деления, сокращения, сравнения,

представления в виде десятичного числа заданной точности, конвертации в тип double, конвертации в строку символов типа char. Операции ‘+’, ’–’, ’*’, ’/’ построены с использованием соответствующих операций над числами типа longint2. Эти классы были откомпилированы в виде динамической библиотеки и протестированы для точного решения систем линейных алгебраических уравнений.

Объявление класса longrat3

```
class longrat3
{
public:
  __int8 sign;
  longint2 num;
  longint2 denom;
  longrat3():num("0"),denom("1"){sign=0;}
  longrat3(char *s);
  longrat3(double d);
  longrat3(char *s1,char *s2);
  longrat3(const                                          longrat3
&l1):num(l1.num),denom(l1.denom),sign(l1.sign){}
  longrat3& operator=(const longrat3 &a);
  friend ostream& operator<<(ostream &f,const longrat3 &a);
  void text(char *s);//return s as 'p/q';
  friend void savebrat3(FILE *f,longrat3& a);
  friend void loadbrat3(FILE *f,longrat3& a);
  friend  longrat3 operator+(longrat3 &a,longrat3 &b);
  friend  longrat3 operator-(longrat3 &a,longrat3 &b);
  friend  longrat3 operator*(longrat3 &a,longrat3 &b);
  friend  longrat3 operator/(longrat3 &a,longrat3 &b);
  void operator+=(longrat3 &a);
  void operator-=(longrat3 &a);
  void operator*=(longrat3 &a);
  void operator/=(longrat3 &a);
  char* decimal(unsigned __int32 size);
  operator double();
  friend __int8 longrabscmp(longrat3 &a,longrat3 &b);
  ~longrat3(){};
  void reduce(void);
};
```

Операции '+' и '–' реализованы по классическим алгоритмам сложения и вычитания; умножение и деление существенно улучшено по сравнению с традиционными алгоритмами умножения и деления в столбик и оптимизированы как по времени, так и по использованию ресурсов ЭВМ.

Ускорение операций умножения и деление длинных чисел

Арифметические операции с точными рациональными числами базируются на умножении длинных целых чисел. Для осуществления сокращения рациональных дробей необходимо деление длинного целого числа на длинное целое число.

Умножение длинных целых чисел с использованием метода Штрассена и дискретного преобразования Фурье

Произведение целых чисел $A = a_0 + a_1 BASE + ... + a_{n-1} BASE^{n-1}$ и $B = b_0 + {} + b_1 BASE + ... + b_{m-1}\ BASE^{m-1}$ в системе исчисления с основанием $BASE$ можно интерпретировать как произведение многочленов

$$\left(a_0 + a_1 x + ... + a_{n-1} x^{n-1}\right)\left(b_0 + b_1 x + ... + b_{m-1} x^{m-1}\right) = c_0 + c_1 x + ... + c_{n+m-1} x^{n+m-1},$$

где c_i – компоненты вектора – свертки векторов $(a_0, a_1, ..., a_n)$ и $(b_0, b_1, ..., b_m)$.

Определение 2. Сверткой векторов a и b называется вектор $c = a \otimes b$ с координатами $c_i = \sum\limits_{k+l=i} a_k b_l$.

Определение 3. Дискретное преобразование Фурье (ДПФ) вектора $\left(a_0, a_1, ..., a_{N-1}\right)$ определяется как комплексный вектор с координатами $\left(y_0, y_1, ..., y_{N-1}\right)$:

$$y_k = \sum_{j=0}^{N-1} a_j \omega^{kj},$$

где ω – главный комплексный корень N-й степени из единицы $\left(\omega = \cos\frac{2\pi}{N} + i \sin\frac{2\pi}{N}\right)$.

Замечание. Обратное дискретное преобразование Фурье $\left(\text{ДПФ}^{-1}\right)$ можно вычислить по формуле

$$a_k = \frac{1}{N} \sum_{j=0}^{N-1} y_j \omega^{-kj}.$$

Теорема о свертке [5]. Преобразование Фурье от свертки двух векторов есть скалярное произведение Фурье-образов этих векторов:

$$c = a \otimes b \ \Leftrightarrow\ F(c) = F(a) * F(b),$$

тогда $c = F^{-1}\left(F(a) * F(b)\right)$.

Вытекает, что для умножения длинных целых чисел A и B достаточно:

вычислить коэффициенты свертки c_i, $i = \overline{0, n+m-1}$;

сделать переносы, чтобы все коэффициенты были меньше $BASE$.

Теорема о свертке дает обоснование для построения эффективного алгоритма вычисления коэффициентов свертки на шаге 1 с помощью ДПФ:

1. Вычислить $F(a)$ и $F(b)$.

2. Скалярно перемножить полученные векторы.

3. Вычислить обратное ДПФ от скалярного произведения.

Умножение многочленов сводится к скалярному произведению соответствующих векторов. Предполагается, что на каждом шаге размеры векторов одинаковые и равные *N*. Сложность алгоритма умножения имеет порядок $O(N \log N)$. Наилучшее быстродействие достигается лишь для таких реализаций ДПФ, которые работают на векторах размером $N=2^k$, поэтому векторы в таких ситуациях необходимо дополнить нулями.

Деление

Пусть число $A=(a_0, a_1,\ldots,a_{n+m-1})$ необходимо разделить на $B=(b_0, b_1,\ldots, b_{n-1})$. Тогда алгоритм деления можно подать как последовательное выполнение делений $(n+1)$-разрядной части A на n-разрядный делитель B. При этом i-й шаг состоит из двух действий [6]:

определить i-ю цифру частицы $q[i]$;

вычесть смещенное произведение $B^* q[i]$ из A. При этом сдвиг B относительно A на каждом шаге уменьшается.

Определение $q[i]$ можно осуществлять таким образом. Пусть очередной шаг представляет собой деление некоторого $U = (u_0, u_1,\ldots,u_n)$ на $B = (b_0, b_1,\ldots, b_{n-1})$.

Если $b_{n-1} > BASE/2$, то имеют место следующие факты [6]:

1) если положить $qGuess = (u_n * BASE + u_{n-1}) / b_{n-1}$, то

$qGuess - 2 < q < qGuess$, где q – верный результат деления U на B;

2) если же дополнительно выполняется неравенство

$qGuess * b_{n-2} > BASE * r + u_{n-2}$, где $r = U - qGuess * B$ – остаток при нахождении $qGuess$ и $qGuess \neq BASE$, то $qGuess - 1 < q < qGuess$, причем вероятность события $qGuess = q+1$ приблизительно равно $2/BASE$.

Таким образом, если $b_{n-1} > BASE/2$, то необходимо вычислить $qGuess = (u_n * BASE + u_{n-1}) / b_{n-1}$ и уменьшать $qGuess$ на единицу до тех пор, пока не будет выполнено условие 2). Полученное значение будет или правильным частным q, или, с вероятностью $2/BASE$, на единицу большим числом. Если же $qGuess = q+1$, будем использовать вычитание, которое вместо отрицательного числа даст дополнение к следующей степени основы $BASE$.

Сложность алгоритма $O(nm)$.

Решение СЛАУ МБМ с использованием усовершенствованного класса рационального числа longrat3

В работе [4] приведены основные стадии алгоритмической схемы нахождения величины ранга, начальной базисной матрицы и решения невырожденной системы (1) с рациональными элементами, которые базируются на технологии точных вычислений. По результатам приведенного алгоритма можно конструировать аналитическое представление общего решения СЛАН [4].

При решении СЛАУ [4] размерностью $m = 50$ с матрицей Гильберта $a_{ij}^{(1)} = 1/(i + j - 1),\ i = \overline{1,m};\ j = \overline{1,m}$ и единичной правой частью с использованием точных вычислений было получено точное решение. При этом время выполнения на ЭВМ (с процессором Athlon 1700+ с частотой 1400 МГц, оперативной памятью 256 Мбайт) составлял 507.9 с. Минимальный ведущий элемент равен 5.56135311E-59. Для СЛАУ с матрицей Гильберта размерностью $m = 100$ и единичной правой частью время нахождения точного решения на ЭВМ составило 6788.29 с [4], что медленнее, чем метод Гаусса с выбором максимального элемента. На других СЛАУ МБМ проявил себя лучше. В частности, для плохо обусловленных СЛАУ с коэффициентами

$$a_{ij}^{(2)} = \begin{cases} m - i + 1, & \text{при } i > j, \\ m - i, & \text{при } i = j, \\ m - j + 1, & \text{при } i < j \end{cases}$$

и правыми частями

$$b_{ij} = \begin{cases} m - i, & \text{при } i \le 2, \\ m - i + 1, & \text{при } i > 2, \end{cases}$$

временные параметры работы МБМ оказалась на порядок лучше, чем метода Гаусса: 1656.38 с и 9767.88 с. Этот вывод подтвердился и для модели размерностью $m = 100$.

В результате экспериментов для СЛАУ с матрицей Гильберта (таблица) размерностью $m = 100$ и единичной правой частью $b_i = 1,\ i = \overline{1,m}$ (с использованием технологии метода Штрассена и быстрого преобразования Фурье) получены точные решения методом базисных матриц, которые совпадают также с результатами, приведенными в [4].

ТАБЛИЦА

Вектор ограничений	Размерность (m)	Метод базисных матриц	
		Минимальный ведущий элемент	Время выполнения, c
$(1,1,\ldots 1)^T$	50	5.56135311E-59	494.359
$(1,1,\ldots 1)^T$	60	0.1416498617E-70	538.922
$(1,1,\ldots 1)^T$	100	0.708461361E-119	1498.48
$(1,1,\ldots 1)^T$	120	0.8034028680E-143	3090.72

Применение нового класса позволило (см. таблицу): ускорить вычисления в 2,5 раза (СЛАУ с матрицей Гильберта размерностью 100 была решена МБМ в точных числах за 1498.48 секунд, тогда как со старым классом на той самой ЭВМ время выполнения составляло 3731.94 с для ЭВМ: AMD Athlon(tm) 64X2 Dual Core Processor 4200+, 2.21 Ггц, 1 ГБ ОП); решать плохо обусловленные СЛАУ большей размерности.

В.І. Кудін, С.І. Ляшко, В.В. Оноцький

ПРО ТЕХНОЛОГІЮ ДОВГОЇ АРИФМЕТИКИ
ПРИ ПОБУДОВІ АЛГОРИТМІВ ДОСЛІДЖЕННЯ ЛІНІЙНИХ СИСТЕМ

Досліджено властивості комп'ютерної реалізації алгоритму методу базисних матриць для системи лінійних алгебраїчних рівнянь у середовищі Visual C++ з використанням множення довгих чисел методом Штрассена, що базується на швидкому перетворенні Фур'є. Наведено результати комп'ютерної реалізації на матрицях Гільберта різної розмірності.

V.I. Kudin, S.I. Ljashko, V.V. Onotsky

ADVANCEMENT OF LONG ARITHMETICS TECHNOLOGY IN ALGORITHMS FOR LINEAR SYSTEM ANALYSIS

New algorithm and computer implementation with Microsoft Visual C++ environment of base matrice method using Schtrassen multiplication of long numbers and fast Fourier transformations are proposed. Computation experiment results for this method for test models with Gilbert matrices in various dimensions are presented.

1. *Форсайт Дж., Молер К.* Численное решение систем линейных алгебраических уравнений // Пер. с англ. – М.: Мир, 1969. – 168 с .
2. *Воеводин В.В.* Вычислительные основы линейной алгебры. –М.: Наука, 1977. – 303 с.
3. *Самарский А.А., Гулин А.В.* Численые методы. – М.: Наука, 1989. – 432 с.
4. *Застосування* технології довгої арифметики при побудові алгоритмів дослідження лінійних систем / В.І. Кудін, С.І. Ляшко, В.В. Оноцький та ін. // Журн. обчислювальної та прикладної математики. – 2007. – 1(94). – С. 61–69.
5. *Кнут Д.* Искусство программирования: 3-е изд. – 2001. – **2**. – С. 337–343.
6. *Кантор Илья.* Большие числа и операции с ними. http://algolist.manual.ru-2002

Получено 24.04.2009

Об авторах:

Кудин Владимир Иванович,
доктор технических наук, старший научный сотрудник
Киевского национального университета имени Тараса Шевченка,

Ляшко Сергей Иванович,
член-корреспондент НАН Украины, профессор, доктор физико-математических наук,
заведующий кафедрой Киевского национального университета имени Тараса Шевченка,

Оноцкий Вячеслав Валериевич,
ассистент Киевского национального университета имени Тараса Шевченка.