ПРОГРАММИРОВАНИЕ

УДК 004.457

ПОДХОД К РЕШЕНИЮ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ С ИНТЕРВАЛЬНОЙ НЕОПРЕДЕЛЕННОСТЬЮ В ИСХОДНЫХ ДАННЫХ

А.В. Панюков, В.А. Голодов

Рассматривается система линейных алгебраических уравнений с интервальной матрицей коэффициентов и интервальной правой частью. Для данных систем вводится понятие псевдорешения. Доказано существование псевдорешения для любых интервальных систем линейных уравнений, предложен способ поиска псевдорешения с помощью решения соответствующей задачи линейного программирования. Вследствие вырожденности полученной задачи для ее решения необходимо использовать вычисления, обеспечивающие точность, намного превышающую возможности стандартных типов данных языков программирования. Симплекс-метод в сочетании с безошибочными дробно-рациональными вычислениями дает решение задачи. Для реализации используется крупнозернистый параллелизм в распределенных системах на основе MPI. Для реализации безошибочных дробно-рациональных вычислений на GPU используется CUDA C.

Ключевые слова: интервальная система линейных уравнений, псевдорешение интервальной системы, линейное программирование, точные вычисления.

Введение

Система линейных алгебраических уравнений – это фундаментальный объект, который встречается при решении многих задач. Часто оказывается, что коэффициенты рассматриваемой системы не могут быть заданы точно, но известны интервалы, которым они принадлежат. В условиях такой интервальной неопределенности коэффициентов необходимо уточнение определения решения. В дальнейшем изложении будем использовать стандартную нотацию, принятую в интервальном анализе [1].

В работах [1–5] систематизированы подходы к учету интервальной неопределенности и дана их классификация. В соответствии с данной классификацией, AE-решением системы линейных алгебраических уравнений $\mathbf{A}x = \mathbf{b}$, в которой элементы матриц $\mathbf{A}$ и $\mathbf{b}$ представляют интервалы $\mathbf{a_{ij}} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]$, $\mathbf{b_j} = \left[\underline{b}_j, \overline{b}_j\right]$, $i, j = 1, 2, \ldots, n$, называют элементы допускового множества

$$\Theta_{tol}(\mathbf{A}, \mathbf{b}) = \left\{ x : \ (\forall i, j = 1, 2, \ldots, n)\, (\forall a_{ij} \in \mathbf{a_{ij}}) \left(\sum_{j=1}^{n} a_{ij} x_j \in \mathbf{b_i} \right) \right\},$$

EE-решением рассматриваемой системы уравнений называют точки объединенного множества

$$\Theta_{uni}(\mathbf{A}, \mathbf{b}) = \left\{ x : \ (\forall i, j = 1, 2, \ldots, n)\, (\exists a_{ij} \in \mathbf{a_{ij}}) \left(\sum_{j=1}^{n} a_{ij} x_j \in \ \mathbf{b_i} \right) \right\},$$

В работах [6, 7] доказано, что поиск EE-решения интервальной системы линейных уравнений является NP-трудной задачей. С другой стороны, в соответствии с теоремой Рона [8] любая точка допустимого множества AE-решений допускает представление в виде $x = x^+ - x^-$, где x^+, x^- являются решением системы неравенств

$$\sum_{j=1}^{n} \left[\underline{a}_{ij} x_j^+ - \overline{a}_{ij} x_j^-\right] \geq \underline{b}_i, \quad \sum_{j=1}^{n} \left[\overline{a}_{ij} x_j^+ - \underline{a}_{ij} x_j^-\right] \leq \overline{b}_i, \ i = 1, 2, \ldots, n, \quad x^+, x^- \geq 0.$$

Следовательно, задача поиска AE-решений имеет полиномиальную сложность.

Методы оценки AE-решений для случаев $\Theta_{tol}(\mathbf{A}, \mathbf{b}) \neq \emptyset$ рассмотрены в работах [1–5] и др. Основным методом исследования допускового множества решений, развиваемым в Новосибирске [3–5], является «метод распознающего функционала». В нем для принятия решения о разрешимости или неразрешимости задачи (т.е. о пустоте/непустоте множества решений) необходимо поработать с некоторым специальным (негладким и вогнутым) функционалом, который назван «распознающим». При этом максимизация распознающего функционала, которую практически можно выполнять, например, с помощью различных методов негладкой оптимизации, разработанных в Институте кибернетики НАН Украины [9], дает достаточно содержательную информацию для возможной коррекции задачи. Разработанные С.П. Шарым и П.И. Стецюком программы для исследования разрешимости интервальной линейной задачи о допусках (пустоты-непустоты допускового множества решений), имеются в свободном доступе на сайте [10]. Программы реализованы в INTLAB'e – интервальном расширении MATLAB'a, а также в Int4Sci – интервальном расширении Scilab'a.

Во многих практических задачах система неравенств (1) оказывается плохо обусловленной или вообще несовместной. В этом случае по аналогии с работами [11, 12] разумным представляется введение понятия «псевдорешения».

Целью данной работы является изложение анонсированного в работе [13] понятия «псевдорешение» для систем уравнений с интервальной неопределенностью и способы построения инструментальных программных средств их поиска.

1. Псевдорешение системы интервальных уравнений

Пусть дана система линейных алгебраических уравнений $\mathbf{A}x = \mathbf{b}$, в которой элементы матриц $\mathbf{A}$ и $\mathbf{b}$ представляют интервалы $\mathbf{a_{ij}} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]$, $\mathbf{b_j} = \left[\underline{b}_j, \overline{b}_j\right]$, $i, j = 1, 2, \ldots, n$.

Для заданной системы уравнений построим параметризованное семейство систем уравнений $\mathbf{A}x = \mathbf{b}(z)$ с модифицированной правой частью $\mathbf{b}(z) = \left[\underline{b} - z\left|\underline{b}\right|, \overline{b} + z\left|\overline{b}\right|\right], z \geq 0$.

Пусть $z^* = \inf\{z : \Theta_{tol}(\mathbf{A}, \mathbf{b}(z)) \neq \emptyset\}$. Псевдорешением исходной системы $\mathbf{A}x = \mathbf{b}$ будем называть внутренние точки допустимого множества $\Theta_{tol}(\mathbf{A}, \mathbf{b}(z^*))$.

Корректность введенного определения подтверждает

Теорема 1. *Для любой системы интервальных уравнений $\mathbf{A}x = \mathbf{b}$ при всех $z > 1$ множество $\Theta_{tol}(\mathbf{A}, \mathbf{b}(z)) \neq \emptyset$.*

Доказательство. В соответствии с теоремой Рона условие $\Theta_{tol}(\mathbf{A}, \mathbf{b}(z)) \neq \emptyset$ эквивалентно совместности системы линейных неравенств

$$\sum_{j=1}^{n} \left[\underline{a}_{ij} x_j^+ - \overline{a}_{ij} x_j^-\right] \quad \geq \quad \underline{b}_i - z\left|\underline{b}_i\right|, \quad i = 1, 2, \ldots, n., \tag{1}$$

$$\sum_{j=1}^{n}\left[\overline{a}_{ij}x_j^+ - \underline{a}_{ij}x_j^-\right] \leq \overline{b}_i + z\left|\overline{b}_i\right|, \quad i=1,2,\ldots,n, \tag{2}$$

$$x^+, x^- \geq 0. \tag{3}$$

Полагая в (1) – (3) $x^+ = x^- = 0$, получим

$$0 \geq \underline{b}_i - z|\underline{b}_i|, \quad 0 \leq \overline{b}_i + z|\overline{b}_i|, \quad i=1,2,\ldots,n.$$

Таким образом, для всех $z \geq 1$ имеет место включение $0 \in \Theta_{tol}(\mathbf{A}, \mathbf{b}(z))$. Теорема доказана. □

2. Способ поиска псевдорешения

Способ нахождения псевдорешения системы уравнений $\mathbf{A}x = \mathbf{b}$ дает

Теорема 2. *Существует решение* $x^{+^*},\ x^{-^*} \in \mathbf{R}^n,\ z^* \in \mathbf{R}$ *задачи линейного программирования*

$$z \to \min_{x^+,\ x^-,\ z}, \tag{4}$$

$$\sum_{j=1}^{n}(\underline{a}_{ij}x_j^+ - \overline{a}_{ij}x_j^-) \geq \underline{b}_i - z|\underline{b}_i|, \quad i=1,2,\ldots,n, \tag{5}$$

$$\sum_{j=1}^{n}(\overline{a}_{ij}x_j^+ - \underline{a}_{ij}x_j^-) \leq \overline{b}_i + z|\overline{b}_i|,\ i=1,2,\ldots,n, \tag{6}$$

$$x_j^+, x_j^-, z \geq 0, \quad j=1,2,\ldots,n, \tag{7}$$

при этом $x^* = x^{+^*} - x^{-^*}$ *является псевдорешением системы* $\mathbf{A}x = \mathbf{b}$.

Доказательство. Сначала докажем существование оптимального решения $x^{+^*},\ x^{-^*} \in \mathbf{R}^n$, $z^* \in \mathbf{R}$ задачи линейного пограммирования (4)–(7). Из теоремы 1 и теоремы Рона следует, что множество допустимых решений рассматриваемой задачи не пусто. Задача, двойственная рассматриваемой, имеет вид

$$\sum_{i=1}^{n}\underline{b}_i y_{1i} - \sum_{i=1}^{n}\overline{b}_i y_{2i} \to \max_{y_{1i}, y_{2i}}, \tag{8}$$

$$\sum_{i=1}^{n}\underline{a}_{ji}y_{1i} - \sum_{i=1}^{n}\overline{a}_{ji}y_{2i} \leq 0, \quad j=1,2,\ldots,n, \tag{9}$$

$$-\sum_{i=1}^{n}\overline{a}_{ji}y_{1i} + \sum_{i=1}^{n}\underline{a}_{ji}y_{2i} \leq 0,\ j=1,2,\ldots,n, \tag{10}$$

$$\sum_{i=1}^{n}|\underline{b}_i|y_{1i} + \sum_{i=1}^{n}|\overline{b}_i|y_{2i} \leq 1, \quad i=1,2,\ldots,n, \tag{11}$$

$$y_{1i}, y_{2i} \geq 0, \quad i=1,2,\ldots,n. \tag{12}$$

Легко заметить, что решение $y_{1i} = y_{2i} = 0, \quad i=1,2,\ldots,n$ является допустимым решением задачи (8)–(12). Таким образом, показано существование допустимых решений как у прямой, так и двойственной задач линейного программирования (задачи (4)–(7) и (8)–(12)). Из теоремы двойственности в линейном программировании следует существование у этих задач оптимальных решений.

Пусть x^{+^*}, x^{-^*}, z^* оптимальное решение задачи (4)–(7). Из теоремы Рона следует, что $x^* = x^{+^*} - x^{-^*}$ является допустимым решением системы $\mathbf{A}x = \mathbf{b}(z^*)$. Из оптимальности z^* следует, что x^* является псевдорешением интервальной системы линейных уравнений $\mathbf{A}x = \mathbf{b}$. Теорема доказана. □

Таким образом, введенное понятие ≪псевдорешение≫ интервальной системы линейных уравнений является вполне конструктивным и позволяет давать результаты при решении интервальных систем в том случае, когда пусковое множество $\Theta_{tol}(\mathbf{A}, \mathbf{b})$ допустимых решений пусто.

Однако, следует обратить внимание на высокую степень вырожденност задач (4)–(7) и (8)–(12)), что будет приводить при использовании приближенных вычислений к зацикливанию симплекс-метода. Избежать зацикливания можно за счет использования вычислений без округления [14, 15]. В этом случае на каждой итерации симплекс-метода количество требуемых бит памяти не превосходит величины $4lm^4 + O(lm^3)$, где m – минимальная из размерностей задачи, l – число бит, достаточных для представления одного элемента матрицы исходных данных, при этом эффективность распараллеливания (т.е. отношение ускорения к числу процессоров) составляет в асимптотике величину, близкую к 100 % [16].

3. Техника реализации

Итак, для успешной реализации изложенного способа поиска псевдорешения на компьютере понадобится несколько составляющих. С одной стороны необходимо обеспечить достаточную точность вычислений для преодоления зацикливания симплекс-метода. Подобная техника описана в [17]. Далее, для обеспечения эффективности всего алгоритма в целом, следует обеспечить достаточное быстродействие точных вычислений, а для использования параллельной архитектуры современных процессоров, произвести эффективную декомпозицию задачи линейного программирования на потоки.

3.1. Обеспечение необходимой точности

В рамках предыдущих исследований были созданы классы `overlong` и `rational`, реализованные в объектно-ориентированной парадигме на языке C++ как библиотека классов `Exact Computation` [14]. Данные классы позволяют производить безошибочные дробно-рациональные вычисления и имеют следующие характеристики. Объектами класса `rational` являются обыкновенные дроби p/q, где p, q - объекты класса `overlong`. Класс overlong предназначен для расширения логических возможностей целочисленных вычислений на компьютере. Объем памяти, занимаемый такими объектами, определяется значениями представляемых чисел, их диапазон ограничен только объемом адресуемой памяти. Диапазон чисел, представляемых объектами класса `overlong`, расширен до $(2^{-32*65535}, 2^{32*65535})$, а минимальный шаг дискретизации чисел, представляемых объектами класса `rational`, может достигать $(2^{-2097150})$. Для объектов классов `overlong` и `rational` определены все операторы, операции и бинарные отношения, используемые для стандартных числовых типов данных. Таким образом, классы `overlong` и `rational` дают потенциальную возможность использовать в программах пользователя безошибочное выполнение основных арифметических операций над полем рациональных чисел.

На сегодняшний день возможность использования безошибочных вычислений представляет известная библиотека GMP(The GNU Multiple Precision Arithmetic Library) [15]. Библиотека распространяется под лицензией GNU LGPL, актуальная версия библиотеки GMP 5.1.1 доступна для загрузки с официального сайта проекта. Программный код оптимизирован под большинство существующих процессорных архитектур, однако она *не предо-*

ставляет своим объектам возможность их использования в распределенных вычислениях. Для более полноценного использования современных процессорных архитектур классы `overlong` и `rational` хранят и оперируют числами по основанию 2^{32}, код операций оптимизирован для системы счисления по основанию степень двойки. Оптимизации применяются также при работе с памятью, поскольку в C++ нет автоматического сборщика мусора, то лишние перевыделения памяти приводят ее фрагментации и снижению быстродействия приложения в целом, краткое описание современных реализаций классов дано в [18]. Для облегчения сопровождения и модификации классов операции с памятью инкапсулированы в отдельный класс `MemHandle`, а выполнение базовых арифметических операций с данными полностью производится в рамках класса `ArifRealization` (см. фрагмент листинга 1). Тем

```
class overlongNM {
 private:
  static ArifRealization realization;
 private:
 MemHandle mhandle;
 ...
 public: inline int32 size() const {return leng;}//leng of number
 public: inline int32 sign() const {return sgn;} //sign of number
 ...
 //addition
 template<typename Type>friend const overlongNM operator+
          (const overlongNM &num,Type v)
           {overlongNM rez(num); return (rez+=v);}
 friend const overlongNM operator+
        (const overlongNM&,const overlongNM&);
 ...
}
```

Листинг 1. Фрагмент класса `overlong`

самым объект класса `overlong` содержит в себе объект типа `MemHandle`, и все действия с памятью происходят через интерфейс `MemHandle`. Все арифметические операции с данными осуществляются вызовом соответствующих методов класса `ArifRealization`. Пример реализации метода `add` класса `overlong` представлен в листинге 2. Эта техника позволяет пол-

```
void overlongNM::add(const overlongNM& alpha,const overlongNM& beta){
 d_t carry;
 const overlongNM& a=(alpha.size()>=beta.size())? alpha:beta;
 const overlongNM& b=(alpha.size()>=beta.size())? beta:alpha;
 int32 LA=a.size(),LB=b.size(),sg=alpha.sgn,newleng;//LA>=LB
 //вызов базовой арифметической операции
 ArifRealization::add(a.mhandle.getptr(),LA,b.mhandle.getptr(),LB,
                      mhandle.providetmpptr(LA,1),newleng,carry);
 mhandle.settmpasptr();
 if(carry) mhandle.safesetvalue(LA,carry);
 leng=newleng;
 sgn=sg;
}
```

Листинг 2. Организация операции сложения

ностью абстрагироваться от реального места хранения данных (за это полностью отвечает

реализация класса `MemHandle`) и способа реализации арифметических операций (конкретная методика выполнения операций над разрядами инкапсулирована в `ArifRealization`).

3.2. Обеспечение необходимой производительности

Вышеописанная техника разбиения класса `overlong`, на три части «интерфейс-память-арифметика», позволяет гибко использовать возможности вычислительной системы. Поскольку производительность алгоритма симплекс метода обеспечивается, в том числе, за счет эффективности реализаций классов `overlong` и `rational`, при их написании учтена возможность использования современных гетерогенных вычислительных сред с GPU ускорителями Nvidia (возможность задействовать GPU от компании AMD и встроенные графических ускорителей, например, Intel HD Graphics дает язык OpenCL, однако эксперименты показали большую трудоемкость процесса кодирования при малом приросте эффективности). Алгоритмы параллельного выполнения базовых арифметических операций, а также некоторые аспекты их реализации в гетерогенной среде описаны в [18]. Хранение операндов организуется с учетом устройства, на котором производятся вычисления. Так, при наличии в системе GPU Nvidia, все данные (разряды) чисел можно хранить непосредственно на GPU, там же выполнять арифметические операции над числами, это снижает до минимума количество пересылок данных по PCI шине, остается лишь скопировать в основную память результаты вычислений. Для систем без GPU, вычисления проводятся на процессоре, данные хранятся непосредственно в оперативной памяти системы.

3.3. Мелкозернистый параллелизм

Параллельные вычисления на GPU требуют переработки алгоритмов базовых арифметических операций с учетом специфической архитектуры устройства. Ниже приводятся листинги некоторых операций для GPU Nvidia на расширениии языка C от Nvidia Corporation (CUDA C).

3.3.1. Сложение

Операция сложения длинных чисел на GPU осуществляется в несколько этапов: параллельное сложение разрядов, синхронизация, параллельное распространение переносов из разрядов. Особенность архитектуры GPU, а именно, выполнение нитей *блоками* и отсутствие синхронизации между блоками требует сохранения «пограничных» переносов во временом массиве (за это отвечает строка номер 16 в листинге 3 и последующего распространения (*DNumAdd_part2*) (листинг 4). Установка параметров и запуск кода на графическом ускорителе представлены на листинге 5. Этот код выполняется на стороне CPU, или, так называемой, стороне Host.

3.3.2. Умножение

Операция умножения одна из наиболее затратных по времени. Для выполнения операции параллельного умножения используется быстрая разделяемая между нитямя блока *__shared__* память. В реализации существенно используется особенности архитектуры GPU Nvidia, а именно, полностью синхронное выполнение инструкций в рамках одного *warp*. Это избавляет от необходимости выполнять синхронизацию между нитями. Исходный код для выполнения на GPU (*kernel*) представлен на листинге 6. Окончательное формирование результата происходит последовательным проходом по массиву **rez[]** и преобразованием 64-битного числа **rez[j]** в непосредственно разряд ответа и разряд переноса. Эти действия

```
__global__ void DNumAdd_part1
(d_t *A,int32 LA,d_t *B,int32 LB, d_t *C,d_t *bGCarry,int32 *f){
 int32 gId=blockDim.x*blockIdx.x + threadIdx.x; int64 tmp=0;
 if (gId >= LA) return; //bound check
 if(gId>=LB) C[gId]=A[gId];
 else{tmp=(int64)A[gId]+(int64)B[gId]; C[gId]=tmp&MAX_DIGIT;}
 __syncthreads();//carry propagation in the block
 int32 lId=threadIdx.x+1,i=gId+1,gS=blockDim.x;
 for(tmp>>=BIT_IN_DIGIT;tmp && i<LA;lId++,i++){
  if(lId==gS){bGCarry[blockIdx.x]=tmp; *f=1; return;}
  tmp+=(int64)C[i]; C[i]=tmp&MAX_DIGIT;  tmp>>=BIT_IN_DIGIT;
 }if(i==LA && tmp) {bGCarry[Lcarry]=1;}
}
```

Листинг 3. Поразрядное сложение

```
__global__ void DNumAdd_part2
(d_t *C, d_t *bGCarry, int32 gS, uint32 Lcarry,uint32 LA){
 //carry propagation between blocks
 int gId = blockDim.x*blockIdx.x + threadIdx.x,i=(gId+1)*gS;
 if (gId >= Lcarry) return;
 uint64 tmp=(uint64)bGCarry[gId];
 for(;tmp && i<LA;i++){
  tmp+=(uint64)C[i];
  C[i]=tmp&MAX_DIGIT;
  tmp>>=BIT_IN_DIGIT;
 }
 if (i==LA && tmp) {bGCarry[Lcarry]=1;}
}
```

Листинг 4. Параллельное распространение переносов

также выполняются на GPU, но в однопоточном режиме.

3.3.3. Деление

Поскольку стандартный алгоритм деления столбиком является абсолютно последовательным, для реализации в вычислительной системе с массовым параллелизмом, какой является GPU, эффективным является использование итеративных методов деления, выражающих результат через операцию умножения.

3.4. Крупнозернистый параллелизм

Применяемая техника параллельной реализации симплекс метода детально описана в [19]. В данном случае для разбиения задачи на потоки использовался механизм многопоточного програмирования **std::thread**, предоставляемый стандартом C++0x11.

4. Вычислительный эксперимент

Вычислительный эксперимент проводился на компьютере с процессором Intel Core i7-950 3.06 ГГц, 6 Гб ОЗУ, GPU Nvidia 460(1гб GDDR5), под управлением ОС Win 7 x64, в качестве компилятора был выбран 64-разрядный Visual C++ 2011. В качестве модельной

```
void ArifRealization::add(const d_t *A,int32 LA,
     const d_t *B,int32 LB,d_t *C, int32 &NL,d_t &Carry){
 int tPerBlock = 128,bPerGrid = (LA + tPerBlock - 1) / tPerBlock;
 d_t *bGCarry_d=NULL,*ansCarry_h=new d_t[1];
 int32 *cF_d=NULL,*cF_h=new int32[1];
 cudaMalloc((void**)&bGCarry_d, sizeof(d_t)*(bPerGrid+1));
 cudaMemset((void*)bGCarry_d, 0, sizeof(d_t)*(bPerGrid+1));
 cudaMalloc((void**)&cF_d, sizeof(int32));
 cudaMemset((void*)cF_d, 0, sizeof(int32));
 DNumAdd_part1 <<< bPerGrid, tPerBlock >>>
  (const_cast<d_t*>(A), LA, const_cast<d_t*>(B), LB, C, bGCarry_d,CF_d);
 cudaMemcpy(cF_h,cF_d,sizeof(int32), cudaMemcpyDeviceToHost);
 if(*cF_h){
  int gS=tPerBlock,LCarry=bPerGrid;
  bPerGrid = (bPerGrid + tPerBlock - 1) / tPerBlock;
  DNumAdd_part2 <<< bPerGrid, tPerBlock>>>
  (d_buffC, bGC_d, gS, LCarry, LA);
 }
 cudaMemcpy(ansCarry_h,&bGCarry_d[bPerGrid],
            sizeof(d_t), cudaMemcpyDeviceToHost);
 NL= (carry = *ansCarry_h)? LA+1: LA;
 delete[] ansCarry_h; cudaFree(bGCarry_d);  cudaFree(cF_d);
}
```

Листинг 5. Вызов функции для исполнения на GPU

```
__global__ void DNumMult(d_t *A,int32 LA,d_t *B,int32 LB,d_t *rez)
{
 int32 lId=threadIdx.x,gId=blockDim.x*blockIdx.x + lId;
 if(gId>=LB) return;
 int32 cBS=(LA+blockDim.x-1)/blockDim.x;
 __shared__ uint64 sha[],shrez[];
 for(int i=lId*cBS;i<(lId+1)*cBS && i<LA;i++){
  sha[i]=A[i]; shrez[i]=0;
 }
 shrez[LA+lId]=0;
 uint64 digit=(uint64)B[gId], t=0UL;
 for(int i=0;i<LA;i++){
  t+=sha[i]*digit;
  shrez[i+lId]+=t&MAX_DIGIT;
  t>>=BIT_IN_DIGIT;
 }
 shrez[LA+lId]+=t&MAX_DIGIT;
 cBS=(LA+blockDim.x+blockDim.x-1)/blockDim.x;
 for(int i=lId*cBS;i<(lId+1)*cBS && i<LA+blockDim.x;i++){
  AtomicAdd(rez[i+gId],shrez[i]);
 }
}
```

Листинг 6. Умножение на GPU

задачи использована система с матрицами:

$$\mathbf{A} = \left[\frac{i*(1-\delta)}{i+j-1}, \frac{i*(1+\delta)}{i+j-1}\right]_{n\times n}; \mathbf{b} = [1, 1/2, \ldots, 1/(n-1), 1/n]^T.$$

Зависимость минимального расширения правой части (параметр z^*), соответствующего псевдорешению, при фиксированном значении $n = 20$ приведена в таблице 1.

Таблица 1

Минимальное расширение правой части системы

δ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
z^*	0,81	0,389	0,1	0,025	0,0062	0,0017

В таблице 2 приведены результаты времени, затраченного на решение задачи для различных размерностей модельной интервальной системы.

Таблица 2

Время работы

Размерность матрицы (n)	10	20	50	100
Время работы	0,46 с	7,73 с	7,39 м	15,1 ч

5. Заключение

В работе полностью рассмотрена задача нахождения псевдорешения интервальной системы линейных алгебраических уравнений, дано определение понятия, показано существование псевдорешения для любой интервальной системы. Дано конструктивное доказательство существования псевдорешения как решения соответствующей задачи линейного программирования. Приведены подробности техники программной реализации симплекс-метода решения задачи линейного программирования, соответствующей псевдорешению интервальной системы, а также некоторые особенности реализации классов `overlong` и `rational`, обеспечивающих необходимую точность вычислений и, тем самым, позволяющих избежать зацикливания симплекс-метода в случае сильной вырожденности сформулированной задачи. Даны результаты численного эксперимента для различных смоделированных исходных данных. Дальнейшая работа направлена на получение более эффективных реализаций за счет оптимизации параллельной версии программы.

Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, соглашение 14.B37.21.0395.

Литература

1. Standardized Notation in Interval Analysis / R.B. Kearfott, M.T. Nakao, A. Neumaie, S.M. Rump, S.P. Shary, P. van Hentenryck // Proc. XIII Baikal International School-seminar: Optimization methods and their applications, Irkutsk, Baikal, July 2–8, 2005. Vol. 4 Interval analysis. – Irkutsk: Institute of Energy Systems SB RAS, 2005. – P. 106–113.
2. Neumaier, A. Interval Methods for Systems of Equations / A. Neumaier. – Cambridge: Cambridge University Press, 1990.
3. Shary, S.P. Solving the Linear Interval Tolerance Problem / S.P. Shary // Mathematics and Computers in Simulation. – 1996. – V. 39. – P. 53–85.
4. Shary, S.P. A New Technique in Systems Analysis under Interval Uncertainty and Ambiguity / S.P. Shary // Reliable Computing. – 2002. – V. 8, № 5. – P. 321–418.
5. Шарый, С.П. Решение интервальной линейной задачи о допусках / С.П. Шарый // Автоматика и телемеханика. – 2004. – №10. – С. 147–162.

6. Lakeyev, A.V. Optimal Solution of Interval Linear Systems is Intractable (NP-Hard) / A.V. Lakeyev, V. Kreinovich // Interval Computations. – 1993. – № 1. – P. 6–14.
7. Lakeyev, A.V. NP-Hard Classes of Linear Algebraic Systems with Uncertainties / A.V. Lakeyev, V. Kreinovich // Reliable Computing – 1997. – № 3. – P. 51–81.
8. Rohn, J. Inner Solutions of Linear Interval Systems / J. Rohn // Interval Mathematics. – 1985. – P. 157–158.
9. Стецюк, П.И. Субградиентные методы с преобразованием пространства для минимизации овражных выпуклых функций / Стецюк П.И. // Современные проблемы прикладной математики и механики: теория, эксперимент и практика: конф., посвящ. 90-летию со дня рождения академика Н.Н. Яненко, Новосибирск. 30 мая – 4 июня, 2011. – http://conf.nsc.ru/niknik-90/ru/reportview/37828 (дата обращения: 11 февраля 2013).
10. Интервальный анализ и его приложения. – URL: http://www.nsc.ru/interval (дата обращения: 11 февраля 2013).
11. Иванов, В.К. О линейных некорректных задачах / В.К. Иванов // ДАН СССР. – 1962. – Т. 145, № 2. – С. 270–272.
12. Тихонов, А.Н. Методы решения некорректных задач / А.Н. Тихонов, В.Я. Арсенин. – М.: Наука, 1979. – 285 с.
13. Панюков, А.В. Техника программной реализации алгоритма решения системы линейных алгебраических уравнений с интервальной неопределенностью в исходных данных / А.В. Панюков, В.А. Голодов // «Параллельные вычисления и задачи управления» PACO'2012. – Шестая междунар. конф., Москва, 24–26 окт. 2012 г.: тр. в 3 т. – М., 2012. – Т. 2. – С. 155–166.
14. Панюков, А.В. Библиотека классов «Exact Computation»/ Свидетельство о государственной регистрации программы для ЭВМ № 2009612777 от 29 мая 2009 г. / А.В. Панюков, М.И. Германенко, В.В. Горбик // Программы для ЭВМ, базы данных, топологии интегральных микросхем: офиц. бюл. Рос. агентства по патентам и товарным знакам. – 2009. – № 3. – С. 251.
15. The GNU MP Bignum Library. – URL: http://gmplib.org/ (дата обращения: 11 февраля 2013).
16. Панюков, А.В. Применение массивно-параллельных вычислений для решения задач линейного программирования с абсолютной точностью / А.В. Панюков, В.В. Горбик // Автоматика и телемеханика. – 2012. – №2. – С. 73–88.
17. Схрейвер, А. Теория линейного и целочисленного программирования: в 2-х т. / пер. с англ. – М.: Мир, 1991. – Т. 1. – 360 с.
18. Голодов, В.А. Распределенные символьные дробно-рациональные вычисления на процессорах x86 и x64 / В.А. Голодов // Параллельные вычислительные технологии (ПаВТ'2012)[Электронный ресурс]: тр. междунар. науч. конф. (Новосибирск, 26 марта – 30 марта 2012 г.). – Челябинск: Издательский центр ЮУрГУ, 2012. – С. 719.
19. Панюков, А.В. Параллельные реализации симплекс-метода для безошибочного решения задач линейного программирования / Панюков А.В., В.В. Горбик // Вестник ЮУрГУ. Серия: Математическое моделирование и программирование. – № 25 (242), вып. 9. – 2011. – С. 107–118.

Анатолий Васильевич Панюков, доктор физико-математических наук, профессор, кафедра «Экономико-математические методы и статистика», Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), anatoly.panyukov@gmail.com.

Валентин Александрович Голодов, аспирант, кафедра «Экономико-математические методы и статистика», Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), avaksa@gmail.com.

Bulletin of the South Ural State University.
Series «Mathematical Modelling, Programming & Computer Software»,
2013, vol. 6, no. 2, pp. 108–119.

MSC 65G40

Approach to Solve the Set of Linear Algebraic Equations with Interval Uncertainty of Data Given

A.V. Panyukov, South Ural State University, Chelyabinsk, Russian Federation, anatoly.panyukov@gmail.com
V.A. Golodov, South Ural State University, Chelyabinsk, Russian Federation, avaksa@gmail.com

The set of linear algebraic equations with interval matrixes of coefficients and interval right part is considered in the paper. The pseudosolution for such systems is introduced. The existence of pseudosolution for all interval sets of algebraic linear equations is proved in the paper, the way for pseudosolution analysis is shown on the basis of the solution the corresponding linear programming problem. It is necessary to use computation guaranteeing sufficient accuracy over standard data types of programming languages because of obtained problem degeneracy. Simplex method coupled with accurate rational-fractional computation gives effective solution to the problem. Coarse-grained parallelism for distributed computer systems with MPI is the instrument of realization. CUDA C software engineering is applied for accurate rational-fractional calculations.

Keywords: interval set of linear equations, pseudosolution of interval equation set, linear programming, exact comtutations.

References

1. Kearfott R.B., Nakao M.T., Neumaier A., Rump S.M., Shary S.P., van Hentenryck P. Standardized Notation in Interval Analysis. *Proc. XIII Baikal International School-seminar: Optimization methods and their applications, Irkutsk, Baikal, July 2–8, 2005. Vol. 4 Interval analysis.* Irkutsk, Institute of Energy Systems SB RAS, 2005, pp. 106–113.
2. Neumaier A. *Interval Methods for Systems of Equations.* Cambridge University Press, 1990.
3. Shary S.P. Solving the Linear Interval Tolerance Problem. *Mathematics and Computers in Simulation*, 1996, vol. 39, pp. 53–85.
4. Shary S.P. A New Technique in Systems Analysis Under Interval Uncertainty and Ambiguity. *Reliable Computing*, 2002, vol. 8, no. 5, pp. 321–418.
5. Shary S.P. An Interval Linear Tolerance Problem. *Automation and Remote Control,* 2004, vol. 65, no. 10, pp. 1653–1666.
6. Lakeyev A.V., Kreinovich V. Optimal Solution of Interval Linear Systems Is Intractable (np-Hard). *Interval Computations*, 1993, no. 1, pp. 6–14.
7. Lakeyev A.V., Kreinovich V. Np-Hard Classes of Linear Algebraic Systems with Uncertainties. *Reliable Computing*, 1997, no. 3, pp. 51–81.
8. Rohn J. Inner Solutions of Linear Interval Systems. *Interval Mathematics and Springer Verlag*, 1985, pp. 157–158.

9. Stecjuk P.I. Subgradient Methods with Space Trasform for Ravine Functions [Subgradientnye metody s preobrazovaniem prostranstva dlja minimizacii ovrazhnyh vypuklyh funkcij]. *Mezhdunarodnaja konferencija Covremennye problemy prikladnoj matematiki i mehaniki: teorija, jeksperiment i praktika, posvjashhennaja 90-letiju so dnja rozhdenija akademika N.N. Janenko.* [International Conference «Recent Developments in Applied Mathematics and Mechanics: Theory, Experiment and Practice. Devoted to the 90th Anniversary of Academician N.N.Yanenko»], 2011. Available at: http://conf.nsc.ru/niknik-90/ru/reportview/37828 (accessed 25 December 2013).
10. *Interval'nyj analiz i ego prilozhenija* [Interval Analysis and Application]. Available at: http://www.nsc.ru/interval (accessed 25 December 2013).
11. Ivanov V.K. About Linear Ill-Conditioned Problems [O linejnyh nekorrektnyh zadachah]. *DAN USSR,* 1962, vol. 145, no. 2, pp. 270–272.
12. Tihonov A.N., Arsenin V.Ja. *Metody reshenija nekorrektnyh zadach* [Ill-Posed Problems-Solving Procedures]. Moscow, Nauka, 1979. 285 p.
13. Panyukov A.V., Golodov V.A. Tehnika programmnoj realizacii algoritma reshenija sistemy linejnyh algebraicheskih uravnenij s interval'noj neopredelennost'ju v ishodnyh dannyh [Software Engendering for Algorithm of Solving a Linear Equation Set under Interval Uncertainty.] *Parallel Computing and Control Problems (PACO'2012). Sixth international conference, Moscow, Russia, October 24–26, 2012,* vol. 2, pp. 155–166.
14. *The GNU MP Bignum Library.* Available at: http://gmplib.org/(accessed 25 December 2013).
15. Panyukov A.V., Gorbik V.V. Using Massively Parallel Computations for Absolutely Precise Solution of the Linear Programming Problems. *Automation and Remote Control,* 2012, vol. 73, no. 2, pp. 276–290.
16. Schrijver A. *Theory of Linear and Integer Programming.* Wiley, 1998. 484 p.
17. Golodov V.A. Distibuted Symbolic Rational Calculations with the x86 and x64 Processors [Raspredelennye simvol'nye drobno-racional'nye vychislenija na processorah x86 i x64]. *Parallel'nye vychislitel'nye tehnologii (PaVT'2012): trudy mezhdunarodnoj nauchnoj konferencii* [Proc. Parallel Computational Technologies (PCT)]. Chelyabinsk: SUSU, 2012, pp. 719.
18. Panyukov A.V., Gorbik V.V. The Parallel Symplex-Method Achievements for Errorless Solving of Linear programming problems. *Bulletin of the South Ural State university. Series «Mathematical Modelling, Programming & Computer software»,* 2011, no. 25 (242), issue 9, pp. 107–118. (in Russian)

Поступила в редакцию 4 декабря 2012 г.