Computer tools in education, 2019
№ 3: 90–95
http://cte.eltech.ru
doi:10.32603/2071-2340-2019-3-90-95

TEACHING STUDENTS TO USE THE GAUSS METHOD FOR INTEGER MATRICES WHEN IMPLEMENTED ON A COMPUTER

Kosovskaya T. M.[1], Doctor of science, Professor, ✉ kosovtm@gmail.com

[1]Saint Petersburg State University, 28 Universitetskiy pr., Stary Peterhof, 198504, Saint Petersburg, Russia

Abstract

The paper is written on the basis of a part of “Analysis of algorithms” course for students of the Computer science department of the Division of mathematics and mechanics of Saint Petersburg State University. The example of the computer implementation of the Gauss method illustrates the difference between the algebraic complexity (the number of arithmetic operations) of processing integers and the computational complexity which depends on the length of the input data. A formula which specifies the increase in the length of matrix coefficients, along with the implementation the Gauss method, is proved. The problems arising in the processing of large integers associated with “chopping” numbers are shown. To overcome the indicated problems, the possibility of using multi-valued integers is proposed. The upper bounds of the number of steps for processing the multi-valued integers is shown to coincide with such bounds for a multi-tape Turing machine.

Keywords: *Gauss method, computational complexity, computation with large integers.*

Citation: T. M. Kosovskaya, “Teaching Students to Use the Gauss Method for Integer Matrices when Implemented on a Computer”, *Computer tools in education*, no. 3, pp. 90–95, 2019; doi:10.32603/2071-2340-2019-3-90-95

1. INTRODUCRION

This article is written as a part of “Analysis of algorithms” lecture course for students of the Computer science department of the Division of mathematics and mechanics of Saint Petersburg State University. The idea of including the material presented here in these courses emerged a long time ago, after N. K. Kosovsky [1] noticed the fact that using the Gauss method with integers (only multiplication and addition/subtraction operations are allowed), the length of matrix elements increases exponentially with the number of iterations and, therefore, with the length of the input data. However, firstly, all textbooks write that the Gauss method is a polynomial algorithm, and secondly, with an exponential increase in the length of the recorded result, the algorithm cannot be polynomial. Appeal to specialists in algebra did not give a positive resolution of this problem: the complexity of an algorithm, in algebra, is considered to be the number of arithmetic operations performed, not paying attention to the increase in the length of the recorded result. The people who are engaged in applications are well aware of the overflows that arise and the “struggle” with them by developing approximate methods.

2. THE PROBLEMS EMERGING WITH COMPUTER IMPLEMENTATION OF GAUSS METHOD FOR INTEGER-VALUED MATRICES

Every mathematician knows the following properties of Gauss method for solving a system of linear equations (as well as for calculation of an inverse matrix): the method is faithful, the method is unstable. With the implementation to matrices with elements of type *real*, various methods of "combating" the instability of the method are developed. Such an instability is closely connected with the fact that the division operation in the computer is performed approximately (moreover, the lower digits of the number are chopped, not rounded).

First-year students learn to implement Gauss method without division (except, may be, division without a remainder) to integer-valued matrices. In such a case there is no a round-off error. It would seem that all the elements of the type *real* of each row of the array can be multiplied by the common denominator of its elements to get an integer matrix of an equivalent system. After that, calculations are made without using division.

What happens in most modern computers if the result of an arithmetic operation with numbers of type *integer* does not fit in a cell? The senior digits of a result are "chopped". That is, in reality, arithmetic operations with numbers of type *integer* or *longinteger* are performed in a computer modulo 2^{16} or 2^{32}, respectively.

Consider a very simple and illustrative example of solving a system of linear equations with integer coefficients on a computing device that runs with decimal integers of length 2 (that is, with integers from the segment $[-99, 99]$ modulo_+ 100). Upon getting a number which length exceeds two digits, the result is "chopped" at the expense of higher digits.[1]

Solve a system of linear equations with an extended matrix

$$\left(\begin{array}{cc|c} 99 & -73 & 26 \\ 84 & 15 & 99 \end{array}\right).$$

Obviously, the solution of this system are the numbers 1 and 1. When using the Gauss method without division, the following actions should be performed: $15\cdot 99 - 84\cdot(-73)$ and $99\cdot 99 - 84\cdot 26$.

In the first expression $15\cdot 99 = 1485 = 85\ (\text{mod}_+\ 100)$, $84\cdot 73 = 6132 = -68\ (\text{mod}_+\ 100)$, $85 + 32 = 117 = -83\ (\text{mod}_+\ 100)$.

In the second expression $99\cdot 99 = 9801 = 1\ (\text{mod}_+\ 100)$, $84\cdot 26 = 2184 = -16\ (\text{mod}_+ 100)$, $1 - (-16) = 17\ (\text{mod}_+\ 100)$.

As a result of applying the first iteration of the Gauss method, we obtain an extended matrix

$$\left(\begin{array}{cc|c} 99 & -73 & 26 \\ & -83 & 17 \end{array}\right).$$

Obviously, the numbers 1 and 1 are not a solution of this system.

3. ANALYSIS OF THE INCREASING COEFFICIENTS WITH THE DIRECT USE OF THE GAUSS METHOD FOR MATRICES WITH INTEGER COEFFICIENTS

Why, when using the Gauss method for matrices with integer coefficients, not apply the algorithm that students are taught in their first year? It's all about the high increase of the elements of the matrix in the process of applying the Gauss method. Let's see how fast they increase.

[1]In a real computer, this happens modulo 2^{16} for numbers of type *integer* or modulo 2^{32} for numbers of type *longinteger*.

It is well known (see, for example, [2]) that the Gauss method requires no more than polynomial (cubic) in the dimension of the matrix number of arithmetic operations. But when estimating the computational complexity of algorithms, the parameter is not the number of arguments of the problem, but the length of their recorded output. How many operations does a computer actually do?

Let an integer $n \times m$ matrix be given, and the length of each of its element does not exceed M ($\| a_{ij} \| \le M$, $i = 1, \dots m$, $j = 1, \dots n$).[2] After the first iteration, we have a matrix of the form

$$\begin{pmatrix} a_{11} & a_{12} \dots a_{1n} \\ 0 & \\ \vdots & B^1 \\ 0 & \end{pmatrix},$$

where elements of the matrix B^1 with indices $i = 2, \dots m$, $j = 2, \dots n$ are calculated via the formula

$$b^1_{ij} = a_{ij} a_{11} - a_{1j} a_{j1}.$$

Taking into account that upon multiplying integers, their lengths are added (perhaps minus one), and upon adding (subtracting) the length of the recorded output does not exceed the maximum of their lengths plus 1, we have

$$\| b^1_{ij} \| \le 2M + 1,$$

$i = 2, \dots m$, $j = 2, \dots n$.

After the k-th iteration we have a matrix of the form

$$\begin{array}{cccccc} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1n} \\ 0 & b^1_{22} & \dots & b^1_{2k} & \dots & b^1_{2n} \\ 0 & 0 & \dots & b^{k-1}_{kk} & \dots & b^{k-1}_{kn} \\ 0 & 0 & \vdots & 0 & B^k & \\ 0 & 0 & \dots & 0 & & \end{array},$$

where the elements of the matrix B^k with indices of the elements $i = k+1, \dots m$, $j = k+1, \dots n$ are calculated via the formula

$$b^k_{ij} = b^{k-1}_{ij} b^{k-1}_{kk} - b^{k-1}_{kj} b^{k-1}_{jk}. \tag{1}$$

Here, of course, it is necessary to stipulate cases where $b^{k-1}_{kk} = 0$, $b^{k-1}_{ki} = 0$ for all $i = k, \dots m$, but in the “worst” case (in terms of the number of operations performed and the increase of the lengths of the coefficients) we have

$$\| b^2_{ij} \| \le 2(2M+1) + 1 = 4M + 3,$$

$$\| b^3_{ij} \| \le 2(4M+3) + 1 = 8M + 7,$$

$$\| b^4_{ij} \| \le 2(8M+7) + 1 = 16M + 15.$$

By induction, taking into account the stipulated cases, it is easy to prove that after the direct passage of the Gauss method for a matrix of rank r

$$\| b^r_{ij} \| \le 2^r (M+1) - 1. \tag{2}$$

It is known [3] that no algorithm, for which the length of the intermediate data exponentially depends on the length of the input data, can run in a polynomial number of steps under the length of the input data. Does it turn out that the Gauss method is not polynomial?

[2]Hereinafter, the notation $\| a \|$ is used for the length of the recorded output of the word (or number) a.

4. ANALYSIS OF THE INCREASING COEFFICIENTS WITH THE DIRECT USE OF THE GAUSS METHOD FOR MATRICES WITH INTEGER COEFFICIENTS USING THE SYLVESTER THEOREM

The Sylvester theorem is formulated in [2]. According to this theorem, for all $k \geq 2$ each element b_{ij}^{k} ($i = k+1,\ldots,m,\ j = k+1,\ldots,n$) can be divided without a remainder by $b_{(k-1)(k-1)}^{k-2}$ (here $b_{11}^{0} = a_{11}$).

Thus, when calculating elements b_{ij}^{k} with $k \geq 2$, it is possible, while remaining in the framework of integers, to use the formula

$$b_{ij}^{k} = \frac{(b_{ij}^{k-1} b_{kk}^{k-1} - b_{kj}^{k-1} b_{jk}^{k-1})}{b_{(k-1)(k-1)}^{k-2}}. \tag{3}$$

In this case, the length of b_{ij}^{k} will decrease by at least the length of $b_{(k-1)(k-1)}^{k-2}$ minus one.

Let us see how the estimate (2) of the length of elements changes after the direct passage of Gauss method for a matrix of rank r.

For $k \geq 2$, $i = k+1,\ldots,m,\ j = k+1,\ldots,n$ we have

$$\| b_{ij}^{k} \| \leq 2 \| b_{ij}^{k-1} \| + 1 - (\| b_{(k-1)(k-1)}^{k-2} \| - 1) =$$

$$2 \| b_{ij}^{k-1} \| - \| b_{(k-1)(k-1)}^{k-2} \| + 2.$$

As $||a_{ij}|| \leq M$ we get

$$\| b_{ij}^{2} \| \leq 2(2M+1) - M + 2 = 3M + 4,$$

$$\| b_{ij}^{3} \| \leq 2(3M+4) - (2M+1) + 2 = 4M + 9,$$

$$\| b_{ij}^{4} \| \leq 2(4M+9) - (3M+4) + 2 = 5M + 16,$$

$$\| b_{ij}^{5} \| \leq 2(5M+16) - (4M+9) + 2 = 6M + 25.$$

By induction, taking into account the mentioned cases, it is easy to prove that after the direct passage of the Gauss method for a matrix of rank r

$$\| b_{ij}^{r} \| \leq (r+1)M + r^{2}. \tag{4}$$

The coefficient $r+1$ at M is essentially less than 2^{r}. Moreover, the estimate (4) of the computational complexity of the Gauss method is polynomial under the length of the input matrix, while the estimate (3) is exponential. So, for example, if $M = 16$ and the rank of the matrix $r = 10$, then the estimate (2) without using the Sylvester theorem gives guaranteed lengths of the coefficients not exceeding $2^{10} \cdot 16 - 1 = 16383 \approx 1024 \cdot 16$. Using this theorem, their guaranteed lengths do not exceed $11 \cdot 16 + 10^{2} = 276 = 17{,}25 \cdot 16$.

However, it is still a much longer number than that for a value of type *integer* (binary length does not exceed 16) or *longinteger* (binary length does not exceed 32).

5. THE USE OF MULTI-DIGIT NUMBERS

For processing numbers which lengths do not allow to write them using standard data types, it is possible to use the so-called multi-digit numbers that can be stored as lists or dynamic arrays. Computations with such numbers are described in detail, for example, in [4]. In this case, the numbers written in one cell are called macro digits.

In lectures given to students, the author proves estimates for the computational complexity of arithmetic operations on multi-digit numbers x and y of length n and m, respectively.

When adding (or subtracting) two multi-digit integers, the number of steps is $O(\max\{n, m\})$, where a step is the addition of two macro-digits.

Predicates of equality and inequalities of multi-digit integers are checked in $\min\{n, m\}$ steps, where by a step we mean a comparison of two macro-digits.

When multiplying two integers under the assumption that $m \le n$ (this condition is checked in 1 step, otherwise the numbers can be multiplied in another order), the number of steps is $O(nm)$, where by a step we mean multiplication or addition of two macro-digits.

The calculation of the partial quotient of dividing two integer multi-digit numbers may be done in $O(nm \cdot (|n-m|)$ steps, where by a step we mean multiplication, or addition (subtraction), or comparison of two macro-digits.

At the same time, students are taught that the estimates obtained coincide with the estimates of the number of steps of a multi-tape Turing machine that performs the corresponding operations.

6. CONCLUSION

The paper describes a theoretical justification for the need to use multi-digit numbers (that is, numbers of arbitrary length that cannot be written as a number of the type *integer* or *longinteger*) using the Gauss method for matrices with integer coefficients.

References

1. T. M. Kosovskaya and N. K. Kosovskii, “About polynomial algorithms for solution of diophantine systems of linear equations and comparisons,” in *Proc. of VIII International seminar Descrete mathematics and its applications*, Moscow, 2004, pp. 72–74 (in Russian).
2. A. Schrijver, *Theory of linear and integer programming*, Wiley, 1998.
3. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: Freeman, 1979.
4. S. M. Okulov, *Programming in algorithms*, Moscow: BINOM, 2007 (in Russian).

Received 05.07.2019, the final version — 30.07.2019.

Kosovskaya Tatiana Matveevna, doctor of physical and mathematical Sciences, associate Professor, Professor of the Department of Informatics of the faculty of mathematics and mechanics of Saint Petersburg state University, ✉ kosovtm@gmail.com

Компьютерные инструменты в образовании, 2019
№ 3: 90–95
УДК: 621.320
http://cte.eltech.ru
doi:10.32603/2071-2340-2019-3-90-95

Обучение студентов использованию метода Гаусса для целочисленных матриц при реализации на компьютере

Косовская Т.М.[1], доктор физико-математических наук, доцент, ✉ kosovtm@gmail.com
[1]Санкт-Петербургский государственный университет, Университетский пр., д. 28, Старый Петергоф, 198504, Санкт-Петербург, Россия

Аннотация

Статья написана на основе части курса "анализ алгоритмов" для студентов кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета. На примере компьютерной реализации метода Гаусса проиллюстрирована разница между алгебраической сложностью (числом арифметических операций) обработки целых чисел и вычислительной сложностью, зависящей от длины записи входных данных. Доказана формула, задающая увеличение длины матричных коэффициентов при реализации метода Гаусса. Показаны проблемы, возникающие при обработке больших целых чисел, связанные с ‘нарезкой" цифр. Для преодоления указанных проблем предлагается возможность использования многозначных целых чисел. Показано, что верхние границы числа шагов при обработке многозначных целых чисел совпадают с такими границами для многоленточной машины Тьюринга.

Ключевые слова: *метод Гаусса, вычислительная сложность, вычисления с большими целыми числами.*

Цитирование: Косовская Т.М. Обучение студентов использованию метода Гаусса для целочисленных матриц при реализации на компьютере // Компьютерные инструменты в образовании. 2019. № 3. С. 90–95. doi:10.32603/2071-2340-2019-3-90-95

Поступила в редакцию 05.07.2019, окончательный вариант — 30.07.2019.

Косовская Татьяна Матвеевна, доктор физико-математических наук, доцент, профессор кафедры информатики математико-механического факультета СПбГУ, ✉ kosovtm@gmail.com