ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 9, Issue, 02, pp.46130-46132, February, 2017

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

ON ALGORITHM FOR A CLASS OF LINEAR ALGEBRA

***Lijiang Zeng**

Research Centre of Zunyi Normal College, Zunyi 563099, GuiZhou, China

ARTICLE INFO

Article History:
Received 13th November, 2016
Received in revised form
25th December, 2016
Accepted 17th January, 2017
Published online 28th February, 2017

Key words:

Linear algebra, Vector space,
Canonical basis, Linear independence.

ABSTRACT

The linear algebra plays important roll to almost all kinds of algebraic system. And the algebraic system in various natural sciences has the extremely widespread application. In this article, we collected a large number of data, describes how to the linear algebra to the transformation of the matrix, and its many kinds of algorithms, and even some algorithm program is presented.

Citation: Lijiang Zeng, 2017. "On algorithm for a class of linear algebra", *International Journal of Current Research*, 9, (02), 46130-46132.

INTRODUCTION

In industry, agriculture, engineering and technology, physics and so on, in the natural sciences, we often used linear algebra (The pure literal rule and polynomial average time., 1985; Sherwin *et al.,* 2003; Matthieu Martel *et al.1998*; Serson *et al., 2001*; Dutertre, 2006; Cowen, 1993). So called linear algebra, is a vector space on the field K (Yunhai, 2002; Context-free languages and Turing machine computations, 1967), it is almost the foundation of all algebraic system, vector space and the algebraic system in various natural science has the extremely widespread application (Ferrer *et al.,* 2014; Drias and Sadeg, 2005; Gilman *et al.,* 1987; Matthieu Martel, 2009; Bolis *et al.,* 2009). In this article, we collected a large number of data, describes how to the linear algebra to the transformation of the matrix, and its many kinds of algorithms, and even some algorithm program is presented.

Generalities on linear algebra algorithms

Let K be a field. Linear algebra over K is the study of K-vector space and K linear maps between them. We will always assume that the vector space that we use are finite-dimensional (Liu *et al.,* 2015). Of course infinite-dimensional vector spaces arise naturally, for example the space $K[X]$ of polynomials in one variable over K. Usually however when one needs to perform linear algebra on these spaces it is almost always on finite-dimensional subspaces.

****Corresponding author: Lijiang Zeng,***
Research Centre of Zunyi Normal College, Zunyi 563099, GuiZhou, China.

As we know, K-vector space V is an abstract object, but in practice we will assume that V is given by a basis of n linearly independent vectors $v_1, v_2, \cdots, v_n$ in some K^m, where m is greater or equal, but not necessarily equal to n. This is of course highly non-canonical, but we can always reduce to that situation. Since K^m has by definition a canonical basis, we can consider V as being given by an $m \times n$ matrix $M(V)$, i.e. a matrix with m rows and n columns, such that the columns of $M(V)$ represent the coordinates in the canonical basis of K^m of the vectors v_i. If $n=m$ the linear independence of the v_i means of course that $M(V)$ is m invertible matrix, where the notation $M(V)$ is slightly improper since $M(V)$ is attached, not to the vector space V, but to the chosen basis v_i.

Linear algebra transformed into a matrix

We note that changing bases in V is equivalent to multiplying $M(V)$ on the right by an invertible $n \times n$ matrix. In particular, we may want matrix $M(V)$ to satisfy certain properties, for example being in upper triangular form. A liner map f between two vector space V and W of respective dimensions n and m will in practice be represented by an $m \times n$ matrix $M(f)$, $M(f)$ being the matrix of the map f with respect to the bases $M(V)$ and $M(W)$ of V and W respectively. In other words, the j-th column of $M(f)$ represents the coordinates of $f(v_i)$ in the basis is w_i where the v_j correspond to the columns of $M(V)$ and the w_i to the columns of $M(W)$. Note that in the above we use column-representation of vectors and not row-representation,

this is quite arbitrary but corresponds to traditional usage. Once a choice is made however one must consistently stick with it. Thus the objects with which we will have to work with in performing linear algebra operations are matrices and (row or column) vectors. This is only for practical purposes, but we keep in mind that it rarely corresponds to anything canonical. The internal representation of vectors is completely straightforward, i.e. as a linear array. For matrices essentially three equivalent kinds of representation are possible. The particular one which should be chosen depends on the language in which the algorithms will be implemented. For example it will not be the same in Fortran and in *C*.

Lattlces algorithms for linear algebra

We have seen that one representation is to consider matrices as (row) vectors of (column) vectors. In fact, we could also consider them as column vectors of row vectors but the former is preferable since we have chosen to represent vectors mainly in column-representation. A second method is to represent matrices as two-dimensional arrays. Finally, we can also represent matrices as one-dimensional arrays, by adding suitable macro-definitions so as to be able to access individual elements by row and column indices. Whatever representation is chosen, we must also choose the index numbering for rows and columns. Although many languages such as *C* take 0 the starting index for consistency with usual mathematical notation we will assume that the first index for vectors or for rows and columns of matrices is always taken to be equal to 1. This is not meant to suggest that one should use this in a particular implementation, it is simply for elegance of exposition. In any given implementation, it may be preferable to make the necessary trivial changes to use 0 as the starting index. Again, this is a language dependent issue.

Gaussian elimination and solving linear systems

In linear algebra algorithms the basic operation which is used is that of Gaussian elimination, sometimes also known Gaussian pivoting. This consists in replacing a column (resp. a row) *C* by some linear combination of all the columns (resp. rows), where the coefficient of C must be non-zero, so that (for example) some coefficient becomes equal to zero. Another operation is that of exchanging two columns (resp. rows). Together these two basic types of operations (which we will call elementary operations on columns or rows) will allow us to perform al1 the tasks that we will need in linear algebra. Note that they do not change the vector space spanned by the columns (resp. rows). Also, in matrix terms performing a series of elementary operations on columns (resp. rows) is equivalent to right (resp. left)multiplication by an invertible square matrix of the appropriate size. Conversely, one can show that an invertible square matrix is equal to a product of matrices corresponding to elementary operations. The linear algebra algorithms that we give are simply adaptations of these basic principles to the specific problems that we must solve, but the underlying strategy is always the same, i.e. reduce a matrix to some simpler form (i.e. with many zeros at suitable places) so that the problem can be solved very simply. The proofs of the algorithms are usual1y completely straightforward, hence will be given only when really necessary. We will systematically use the following notation: *M* is a matrix, M_j denotes its *j*-th column M_i its *i*-th row and $m_{i,j}$ the entry at row *i* and column *j*. If *B* is a (column or row) vector, b_i will denote its *i*-th coordinate. Perhaps the best way to see Gaussian elimination in action is in solving square linear systems of equations.

Algorithm 1. (Square Linear System)

Let *M* be an $n \times n$ matrix and *B* a column vector. This algorithm either outputs a message saying that *M* is not invertible or outputs a column vector *X* such that *MX=B*. We use an auxiliary column vector *C*.

1. [lnitialize] Set $j \leftarrow 0$.

2. [Finished?] Let $j \leftarrow j+1$. If $j > n$ go to step 6.

3. [Find non-zero entry] If $m_{i,j} = 0$ for all $i \geq j$, output a message saying that *M* is not invertible and terminate the algorithm. Otherwise, let $i \geq j$ be some index such that $m_{i,j} \neq 0$.

4.[Swap?] If $i > j$ for $l = j, \cdots, n$ exchange $m_{i,l}$ and $m_{j,l}$ and exchange b_i and b_j.

5.[Eliminate] (Here $m_{j,j} = 0$.) Set $d \leftarrow m_{j,j}^{-1}$ and for all $k > j$ set $c_k \leftarrow dm_{k,j}$. Then, for all $k > j$ and $l > j$ set (Note that we donot neèd to compute this for $l = j$ since' it is equal to zero.) Finally for $k > j$ set $b_k \leftarrow b_k \leftarrow c_k b_j$ and go to step 2.

6. [Solve triangular system] (Here *M* is an upper triangular matrix.) For $i = n, n-1, \cdots, 1$ (in that order) set $x_i \leftarrow (b_i - \sum_{i<j\leq n} m_{i,j}x_j)/m_{i,i}$ output $X = (x_i)_{1\leq i\leq n}$ and terminate the algorithm.

Note that steps 4 and 5 (the swap and elimination operations) are really row operations, but we have written them as working on entries since it is not necessary to take into account the first *j*-1 columns. Note also in step 5 that we start by computing the inverse of $m_{j,j}$ since in fields F_p division is usually much more time-consuming than multiplication.

Conclusion

According to above, the number of necessary multiplications/divisions in this algorithm is clearly asymptotic to $n^3/3$ in the general case. Note however that this does not represent the true complexity of the algorithm, which should be counted in bit operations. This of course depends on the base field. Inverting a square matrix *M* means solving the linear systems $MX = E_i$ where the E_i are the canonical basis vectors of K^n, hence one can achieve this by successive applications of Algorithm 1. Clearly it is a waste of to use Gaussian elimination on the matrix for each linear system. More generally this is true when we must solve several linear systems with the same matrix *M* but different right hand sides *B*. We should compute the inverse of *M* and then one solution of a linear system requires only a simple matrix times vector

multiplication requiring n^2 field multiplications. To obtain the inverse of M only a slight modification of Algorithm 1 is necessary.

REFERENCES

Bolis, A., C.D. Cantwell, D. Moxey, D. Serson, S.J. Sherwin, 2009. An adaptable parallel algorithm for the direct numerical simulation of incompressible turbulent flows using a Fourier spectral/ hp element method and MPI virtual topologies. *Computer Physics Communications,* (3), 102-105.

Context-free languages and Turing machine computations. Hartmanis *J. Proceedings of Symposia in Applied Mathematics*, 1967, 90-95.

Cowen, R. 1993. Some connections between set theory and computer science. *Computational logic and proof theory,* (1), 46-50

Drias, H., S. Sadeg, S. Yahi, 2005. Cooperative bees swarm for solving the maximum weighted satisfiability problem[C]. Computational Intelligence and Bioinspired Systems, 86-90.

Dutertre, B., L. De Moura, 2006. "A Fast Linear-Arithmetic Solver for DPLL(T),". *Computer Aided Verification,* (3), 56-61

Ferrer, E., D. Moxey, R. H. J. Willden, S. J. Sherwin, 2014. Stability of Projection Methods for Incompressible Flows Using High Order Pressure-Velocity Pairs of Same Degree: Continuous and Discontinuous Galerkin Formulations. *Communications in Computational Physics*, (3), 68-71

Gilman, P., C. Stramma, C. L. Cornillon *et al*. 1987. Processingand analysis of large volumes of satellite-derived thermalinfrared data. *Chinese Journal of Geophysics,* (3), 78-82.

Liu, L., M. Mirzangar, R.M. Kirby, R. Whitaker, D. H. House, 2015. Visualizing Time-Specific Hurricane Predictions, with Uncertainty, from Storm Path Ensembles. *Computer Graphics Forum,* (3), 67-70.

Matthieu Martel, 2009. Enhancing the implementation of mathematical formulas for fixed-point and floating-point arithmetics. *Formal Methods in System Design*, (3),78-81

Matthieu Martel, Amine Najahi, Guillaume Revy, 1998. Trade-offs of certified fixed-point code synthesis for linear algebra basic blocks. *Journal of Systems Architecture,* 89-92.

Semantics of round off error propagation in finite precision calculations[J]. Matthieu Martel. Higher-Order and Symbolic Computation, 2006(1), 88-93

Serson, D., J.R. Meneghini, S.J. Sherwin, 2001. Velocity-correction schemes for the incompressible Navier-Stokes equations in general coordinate systems. *Journal of Computational Physics*, 53-56.

Sherwin, S.J., V. Franke, J. Peiró, K. Parker, 2003. One-dimensional modeling of a vascular network in space-time variables. *Journal of Engineering Mathematics,* (3),78-82.

The pure literal rule and polynomial average time, 1985. Paul Walton Purdom, Cynthia A. Brown. *SIAM Journal on Computing,* 122-126

Yunhai, Z., Yong, M. 2002. Optimal algorithm for system reconstruction [C]. Power System Technology, 2002. Proceedings. Power Con 2002. International Conference on. 82-87.

****