Journal of Applied Mathematics & Bioinformatics, vol.3, no.1, 2013, 193-203
ISSN: 1792-6602 (print), 1792-6939 (online)
Scienpress Ltd, 2013

Numerical Linear Algebra methods in Data Encoding and Decoding

Dimitrios Triantafyllou[1]

Abstract

In this paper, Numerical Linear Algebra techniques for data encoding and decoding are presented. The algorithms are based on matrix triangularization leading to an efficient coding procedure. Rounding off errors can cause serious problems in the process of coding and thus the decoding may be led to inaccurate results. Techniques for improving the stability of the whole process are proposed. The complexity and the error analysis of the proposed methods are discussed. Examples illustrating the algorithms are presented.

Mathematics Subject Classification: 68P25, 94A60, 65F05, 65G50, 15A23
Keywords: LU factorization; matrix triangularization; encryption; decryption

[1] University of Military Education, Hellenic Army Academy, 16673, Vari, Greece, e-mail: dtriant@math.uoa.gr

Article Info: *Received* : November 1, 2012. *Revised* : January 30, 2013
Published online : March 30, 2013

1 Introduction

The application of Numerical Linear Algebra procedures in data encryption has attracted the scientists in recent years. Among matrix factorization, LU is the most popular. Except from the solution of Linear Systems, severa; other applications using LU factorization arise as the computation of the Greatest Common Divisor of several polynomials [1, 9], image deblurring [7, 10] in Copmuter Graphics and many other.

In [6], an alternative approach for the generation of new cryptographic functions, applying the LU factorization to Vandermonde matrices is proposed. An approach integrating data encryption and distribution is presented in [2] and the authors in [8, 12] take full advantage of the characteristics of the sparse parity check matrix, such as cyclicity and equality of row weight.

In this paper an application of matrix triangularization in encoding and decoding data is presented. More precisely, a known method which uses non-orthogonal transformations is developed and an improved version of the proposed algorithm in [2] is presented. Non-orthogonal transformations are faster than the orthogonal ones. Orthogonal transformations are more stable but require more floating point operations. Rounding off errors during numerical computations in floating point arithmetic can lead to wrong data decoding when Numerical Linear Algebra methods are used. A scope of this paper is to improve the stability of the algorithms in order to conclude to efficient and reliable procedures which encode in a safe way the initial data and guarantee a solid decoding.

In Section 2, the required theoretical background is presented and techniques improving the stability of the methods are proposed. In Section 3, the application of the described methods in data encryption and decryption is presented. In Section 4, useful conclusions are presented.

2 Triangularization of a matrix

There are various methods for triangularizing a matrix. These methods can be separated in two categories. The first one uses non-orthogonal transformations and the second orthogonal. The algorithms can be implemented either numerically or symbolically. The numerical evaluation of the computations in floating point arithmetic makes the procedure faster but rounding off errors such as catastrophic cancellation of significant digits can cause serious problems to the stability of the methods. On the other hand, the symbolical evaluation of the computations guarantees that the final result will be the real one, with no errors, but the increase of the required time is significant. Below, the LU factorization with pivoting using non-orthogonal orthogonal transformations is presented.

2.1 LU factorization with pivoting

The LU factorization of a matrix A results through row operations (Gaussian elimination) to a lower triangular matrix L and an upper one U such that $A = L \cdot U$. There are two decompositions achieving the previous scheme: Doolittle's results to a unit lower triangular matrix L and an upper U and Crout's to a unit upper triangular matrix U and a lower L. In order to enforce the stability of the algorithm, partial or complete pivoting can be used. More precisely, in partial pivoting, in $k-$th step of the procedure, the row with the largest $a_{ii}, i = k, k+1, \ldots, m$, element is interchange with the $k-th$ one, in order to keep the multipliers less or equal to 1. In this manner, the whole procedure is stable in practise. Another issue that must be discussed is the inner tolerance of the procedure. Due to rounding off errors, it is possible some negligible quantities to appear. These quantities should be zeroed. An inner tolerance ϵ_t is used for this purpose. We zero any element which is less in absolute value than ϵ_t. The choice of the most suitable ϵ_t is not easy. Different ϵ_t's can lead to different results. Bellow, the LU factorization with partial pivoting of an $m \times n$ matrix is presented.

Algorithm LU Factorization with row pivoting

```
for $k = 1 : min\{m-1, n\}$
    Find $r : |a_{r,k}| = max_{k \leq i \leq m}\{|a_{i,k}|\}$
    Interchange rows $k$ and $r$
    $m_{ik} = s_{ik}/s_{kk},\ i = k+1 : m$
    $a_{ij} = a_{ij} - m_{ik}a_{kj},\ i = k+1 : m, j = k+1 : n$
```

Set $a_{i,j} = 0$ if $|a_{i,j}| \leq \epsilon_t$, $i = k : m+n, j = k : m+n$

Row interchanges can be saved in a vector p, where p_i is the number of row which is the pivot one in step i of the procedure. If P_i is the permutation matrix in step i and $P = P_{n-1} \cdot \ldots \cdot P_2 \cdot P_1$, then the Gaussian elimination with partial pivoting yields LU factorization of a permuted matrix A as it is shown in the following scheme.

$$P \cdot A = L \cdot U.$$

In order to improve the stability, the LU factorization with complete pivoting can be used. The difference with partial pivoting is that in every step includes not only rows but also column interchanges. The column interchanges are saved again in a vector q and if Q_i is the column permutation matrix in step i and $Q = Q_1 \cdot Q_2 \ldots Q_{n-1}$ then it holds

$$P \cdot A \cdot Q = L \cdot U.$$

2.2 Numerical Complexity

The required floating point operations of LU factorization of an $m \times n$ matrix is $O(n^2(m - \frac{n}{3}))$.

2.3 Error Analysis

The LU factorization, is the exact factorization of the slightly disturbed initial matrix A:

$$A + E = L \cdot U, \quad ||E||_\infty \leq n^2 \rho u ||A||_\infty,$$

where ρ is the growth factor (in case of row pivoting) and u the unit round off. The theoretical bound of the norm of the error matrix is unfortunately large because of the growth factor. More precisely,

$$\rho \leq 2^{n-1}$$

in Gaussian elimination with partial and

$$\rho \leq (n \cdot 2^1 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \ldots \cdot n^{1/(n-1)})^{1/2}$$

in complete pivoting respectively [3, 11]. Wilkinson in [11] and Higham and Higham in [5] have constructed matrices for which the upper bound of the of the growth factor is attained in partial pivoting. In practise, these examples for which the LU factorization with partial pivoting fails are rare. They can not be found in real applications, since the elements of the initial matrix have to satisfy some relationships in order to fail the algorithm (we have to construct such matrices). It follows that the LU factorization with partial pivoting can be considered stable in practise and it is the most popular algorithms for triangularizing a matrix. The bound for the growth factor in complete pivoting is a slowly growing function of n, it is not attained in practice and thus Gaussian elimination with complete pivoting is a stable procedure.

Although the LU factorization with partial pivoting is stable in practice, if the entries of A vary widely, then there is a chance the accuracy of the procedure to be affected. More precisely, it is possible during the Gaussian elimination, a small number to be added to a large one. In this case, due to catastrophic cancellation of significant digits in exponential alignment, the results of the computations could lead to failure as it is shown below.

Example 2.1. Let $A = \begin{bmatrix} 10 & 10^6 \\ 1 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} 10^6 \\ 2 \end{bmatrix}$. Then, applying Gaussian elimination with row pivoting in order to solve the system $A \cdot x = b$ in a 4 digit arithmetic the result is $x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ which differs significant from the real solution which approximately is $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. In order to treat in an efficient way such circumstances, a suitable scaling of the entries of the initial matrix can be chosen. More specifically, using an invertible diagonal matrix D such that the entries of every row of $D^{-1} \cdot A$ to be of the same order, we can solve equivalently the system $D^{-1} \cdot A \cdot x = D^{-1} \cdot b$. Similarly, we can scale rows and columns of A by multiplying it with D_1^{-1} and D_2 from the left and the right respectively. In this case, the largest element in magnitude of $D_1^{-1} \cdot A \cdot D_2$ is between $1/\beta$ and 1, where β is the base of the number system [3]. Thus, we have to solve the system

$$\begin{cases} D_1^{-1} \cdot A \cdot D_2 \cdot y = D_1^{-1} \cdot b \\ \\ y = D_2^{-1} \cdot x \end{cases}$$

The previous procedure is known as equilibration [4].

3 Application to data encryption and decryption

Let A be an $n \times n$ matrix containing the initial information which can be eg. a text or an image. The text can be easily transformed to numbers. An image can be represented by non negative integers as follows.

Let I be a $2D$ image of dimensions $n \times n$. The previous image can be represented by a $2D$ matrix of size $n \times n$. The (i, j)-th element of the matrix I corresponds to the color values of the position of the (i, j)-th pixel of the image. Thus

$$I = \begin{bmatrix} I_{1,1} & I_{1,2} & I_{1,3} & \dots & I_{1,n} \\ I_{2,1} & I_{2,2} & I_{2,3} & \dots & I_{2,n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ I_{n,1} & I_{n,2} & I_{n,3} & \dots & I_{n,n} \end{bmatrix}$$

If the image is in grayscale, only one such matrix is needed for its representation. The element $I_{i,j}$ denotes the grayscale shade value of the (i, j)-th pixel of the image (intensity information). The value 0 corresponds to black (weakest intensity) and 1 corresponds to white (strongest intensity). In case that the initial image is coloured then three such matrices are needed, one for Red, one for Green and one for Blue (RGB).

The scheme of the LU factorization evaluates a secret variance and simultaneously, an encryption of the initial data, since a part of the encrypted information is saved in the lower triangular matrix L and another one in the

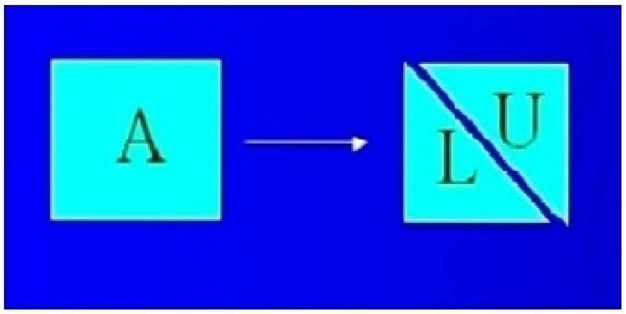

Figure 1: LU Decomposition

upper triangular U (see Figure 1). The extraction of the initial data is impossible, since it is an NP-hard problem to reconstruct the initial matrix A knowing only L or only U [2].

Although the proposed algorithm provides high security in data encoding, the use of floating point arithmetic can cause some serious problems in the evaluation of the procedure as it is discussed in Section 2. The use of partial or complete pivoting can offer even more security improving the stability of the algorithm without increasing significantly the required complexity and storage capacity. The scaling of the initial data, as described in Section 2, can also protect the proposed method from some more extreme circumstances, where the entries of the initial matrix A, containing the initial data, vary widely. This variation would be intended in data encoding for security reasons but simultaneously makes the evaluation of the algorithms more liable to rounding off errors and can lead the whole process to failure (eg. false decoding).

Another issue is the use of the inner tolerance ϵ_t in the encoding procedure. As it is mentioned in Section 2, different ϵ_t's can lead to different LU factorizations. The proper selection of the quantity ϵ_t is not always an easy task and it depends many times on the initial data. A wrong choice of ϵ_t can result to such matrices L and U that lead the decoding process to failure.

Data recovery can be achieved using the formula for matrix multiplication: $A = L \cdot U$ (without pivoting, not stable), $P \cdot A = L \cdot U$ (with partial pivoting) or $P \cdot A \cdot Q = L \cdot U$ (with complete pivoting), where P and Q are the identity matrix with interchanged rows/columns. In order to decrease the required storage capacity, L can be saved in the lower triangular of A and U in the upper one (the ones in diagonal is not needed to be saved) and the row/column interchanges can be saved in a vector v/u instead of a two-dimensional matrix P/Q respectively. In this manner, the required storage capacity is the same with that of the initial data. The multiplications are evaluated taking advantage of the special form of L and U reducing the required complexity.

Example 3.1. Let $A = \begin{bmatrix} 10 & 1 & 2 & 10^{20} \\ 2 & 5 & 1 & 1 \\ 2 & 1 & 3 & 1 \\ 2 & 10^{12} & 1 & 2 \end{bmatrix}$ be the matrix containing the initial data which have to be encoded. Applying the LU factorization with partial pivoting to A we get

$$\begin{aligned} L &= \begin{bmatrix} 1.000000000000000 & 0 & 0 & 0 \\ 0.200000000000000 & 1.000000000000000 & 0 & 0 \\ 0.200000000000000 & 0.000000000000800 & 1.000000000000000 & 0 \\ 0.200000000000000 & 0.000000000004800 & 0.230769230768166 & 1.000000000000000 \end{bmatrix}, \\ U &= \begin{bmatrix} 10 & 1 & 2 & 10^{20} \\ 0 & 9.999999999998001 \cdot 10^{11} & 0.600000000000000 & -2.000000000000000 \cdot 10^{19} \\ 0 & 0 & 2.599999999999520 & -1.999999999998400 \cdot 10^{19} \\ 0 & 0 & 0 & -1.538461538454438 \cdot 10^{19} \end{bmatrix}, \end{aligned}$$

and the permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

which can be stored as the vector $p = [1 \;\; 4 \;\; 3 \;\; 2]$ for saving storage capacity. As we can observe, the entries of some rows of A vary widely. As it is shown below, this variation caused serious problems during the numerical evaluation of LU factorization with partial pivoting, leading to an inaccurate final result. The product of $P \cdot L \cdot U$, which is the decoding, should be equal with the matrix A, which includes the initial data. But,

$$A - P \cdot L \cdot U = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

which means that the encoded message was wrong decoded.

In order to encode and decode efficiently the previous data, we use scaling before the partial pivoting. More precisely, we multiply matrix A from the left with D^1_{-1} and from right with D_2, where

$$\begin{aligned} D_1 &= \begin{bmatrix} \|A(1,:)\|_F & 0 & 0 & 0 \\ 0 & \|A(2,:)\|_F & 0 & 0 \\ 0 & 0 & \|A(3,:)\|_F & 0 \\ 0 & 0 & 0 & \|A(4,:)\|_F \end{bmatrix} \\ &= \begin{bmatrix} 10^{20} & 0 & 0 & 0 \\ 5.567764362830022 & 0 & 0 & 0 \\ 0 & 0 & 3.872983346207417 & 0 \\ 0 & 0 & 0 & 10^{12} \end{bmatrix}, \\ D_2 &= \begin{bmatrix} \frac{1}{\|A(1,:)\|_F} & 0 & 0 & 0 \\ 0 & \frac{1}{\|A(2,:)\|_F} & 0 & 0 \\ 0 & 0 & \frac{1}{\|A(3,:)\|_F} & 0 \\ 0 & 0 & 0 & \frac{1}{\|A(4,:)\|_F} \end{bmatrix} \\ &= \begin{bmatrix} 0.094491118252307 & 0 & 0 & 0 \\ 0 & 0.000000000001000 & 0 & 0 \\ 0 & 0 & 0.258198889747161 & 0 \\ 0 & 0 & 0 & 10^{-20} \end{bmatrix}, \end{aligned}$$

where $A(i,:)$ denotes the i-th row of matrix A, $A(:,j)$ denotes the j-th column of matrix A and $\|.\|_F$ denotes the Frobenius norm.

Applying the LU factorization with partial pivoting to $D_1^{-1} \cdot A \cdot D_2$ we get

$$\begin{aligned} L &= \begin{bmatrix} 1.000000000000000 & 0 & 0 & 0 \\ 0.000000000003873 & 1.000000000000000 & 0 & 0 \\ 0.695608343640252 & 0.718421208107818 & 1.000000000000000 & 0 \\ 0.000000000000000 & -0.000000000000000 & 0.000000000000000 & 1.000000000000000 \end{bmatrix}, \\ U &= \begin{bmatrix} 0.048795003647427 & 0.000000000000258 & 0.200000000000000 & 0.258198889747161 \cdot 10^{-20} \\ 0 & 0.000000000001000 & -0.000000000000516 & 0.000000000001000 \cdot 10^{-20} \\ 0 & 0 & -0.092747779151663 & -0.000000000000718 \cdot 10^{-20} \\ 0 & 0 & 0 & 1.000000000000000 \cdot 10^{-20} \end{bmatrix}, \end{aligned}$$

and the permutation matrix

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Now,

$$D_1^{-1} \cdot A \cdot -P \cdot L \cdot U = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -0.252 \cdot 10^{-28} & 0 & -0.505 \cdot 10^{-28} & 0 \\ 0 & 0 & 0.139 \cdot 10^{-16} & 0 \\ 0 & -8.211 \cdot 10^{-48} & -3.009 \cdot 10^{-36} & 0 \end{bmatrix}$$

which means that the error in the decoding is of the order of the epsilon of the machine.

4 Conclusions

In this paper we presented an application of the LU factorization with partial or complete pivoting for data encoding and decoding. The LU factorization accomplishes a dispersal and an encryption of initial data. The extraction of initial data, even if the matrix L or U is available, is an NP-hard problem. Rounding off errors and not proper selection of inner tolerance ϵ_t during the numerical evaluation of data encoding using LU factorization with partial or complete pivoting can lead to inaccurate decoding. The use of scaling (row or/and column) in the initial data improves significant the behaviour of the procedure resulting to an efficient and reliable method.

References

[1] S. Barnet, Greatest Common Divisor from Generalized Sylvester Resultant Matrices, *Linear and Multilinear Algebra*, **8**, (1980), 271–279.

[2] Sung Jin Choi and Hee Yong Youn, A Novel Data Encryption and Distribution Approach for High Security and Availability Using LU Decomposition, *LNCS*, **3046**, (2004), 637–646.

[3] B.N. Datta, *Numerical Linear Algebra and Applications*, Second Edition, SIAM, United States of America, 2010.

[4] G.E. Forsythe and C.B. Moller, *Computer Solutions of Linear Algebraic Systems*, Englewood Cliffs, New Jersey, Prentice Hall, 1967.

[5] N.J. Higham and D.J. Higham, Large growth factors in Gaussian elimination with pivoting, *SIAM J. Matrix Anal. Appl.*, **10**(2), (1989), 155–164.

[6] E.C. Laskari, G.C. Meletiou, D.K. Tasoulis and M.N. Vrahatis, Transformations of two cryptographic problems in terms of matrices, *ACM SIGSAM Bulletin*, **39**(4), (2005), 127–130.

[7] S. Unnikrishna Pillai and Ben Liang, Blind Image Deconvolution Using a Robust GCD Approach, *IEEE Transactions on Image Processing*, **8**, (1999), 295–301.

[8] Jia-ning Su, Zhou Jiang, Ke Liu, Xiao-yang Zeng and Hao Min, An efficient low complexity LDPC encoder based on LU factorization with pivoting, *ASIC and Syst. State Key Lab, Fudan Univ., Shanghai , ASIC*, (2005), 107–110.

[9] M. Mitrouli and D. Triantafyllou, On rank and null space computation of the generalized Sylvester matrix, *Numerical Algorithms*, **54**, (2010), 297–324.

[10] A. Danelakis, M. Mitrouli and D. Triantafyllou, Blind image deconvolution using a banded matrix method, *Numerical Algorithms*, to appear.

[11] J.C. Wilkinson, *The algebraic Eigenvalue Problem*, Oxford University Press Inc., New York, USA, 1968.

[12] Peng Zhang, Rui Lv, Gang Yang and Jinlun Chen, An Improved LU Decomposition Encoding Method of LDPC Codes, *Proceedings of the Management and Service Science, MASS '09, Wuhan*, (2009), 1–4.