Quest Journals
Journal of Research in Applied Mathematics
Volume 7 ~ Issue 12 (2021) pp: 01-03
ISSN(Online) : 2394-0743 ISSN (Print): 2394-0735
www.questjournals.org

Research Paper

Inverse of Non-Singular Square Matrix

Liana Zhu
[1](*Computer Science Department, California State University, Los Angeles, United States)*
Corresponding Author: Liana Zhu

ABSTRACT: *Given any real or complex non-singular square matrix, the inverse of the matrix can be found efficiently.*
KEYWORDS: *matrix, inverse, inversion, nonsingular, square, algorithm, iterative, fast, time complexity, efficient*

***R**eceived 28 Nov, 2021; **R**evised 10 Dec, 2021; **A**ccepted 12 Dec, 2021 © The author(s) 2021.*
P**ublished with open access at **www.questjournals.org

I. INTRODUCTION

To reduce the computational complexity of matrix inversion, which is the majority of processing in many practical applications, efficient algorithms for calculating the inverse of an arbitrary nonsingular matrix are presented. Based on the algorithms presented by F. C. Chang [1-3] and some related research [4-5], this compact algorithm may improve the time complexity and provide an even greater computational savings. The following are the algorithm, source code and some typical example with simple error checking.

II. ALGORITHM

The presented algorithm consists of the following steps:

Step 1 Initialization:
Let M be a non-singular n × n matrix. Let I be an n × n identity matrix.

```
n = length(M); I = eye(n); N = [1:n]; M = M-I; W = I;
```

Step 2 Iterative operations:

```
for k = 1:n;   K = [1:k];
    W(N,K)  = W*(I(N,K)-M(N,k)*W(k,K)/(1+W(k,N)*M(N,k)));
end;
```

Step 3 Output:
Print out the result of the inverse of M, which is W.

III. MATLAB SOURCE CODE

```
function W = inv_fcz(M)
%    Compute the inverse of non-singular square matrix
%    Error checking:  W*M = I

   n = length(M); N = [1:n]; I = eye(n); M = M-I; W = I;

for k = 1:n;   K = [1:k];
   W(N,K)  = W*(I(N,K)-M(N,k)*W(k,K)/(1+W(k,N)*M(N,k)));
end;
```

IV. TYPICAL EXAMPLES

```
» format short
» n=7, M=magic(n), W=inv_fcz(M), ck=W*M,

 n =
     7
 M =
    30    39    48     1    10    19    28
    38    47     7     9    18    27    29
    46     6     8    17    26    35    37
     5    14    16    25    34    36    45
    13    15    24    33    42    44     4
    21    23    32    41    43     3    12
    22    31    40    49     2    11    20
 W =
    0.0008    0.0008    0.0212   -0.0195   -0.0021    0.0041    0.0004
   -0.0021    0.0241   -0.0195    0.0012    0.0004    0.0008    0.0008
    0.0212   -0.0191    0.0004   -0.0021    0.0037    0.0008    0.0008
   -0.0170    0.0008    0.0008    0.0008    0.0008    0.0008    0.0187
    0.0008    0.0008   -0.0021    0.0037    0.0012    0.0207   -0.0195
    0.0008    0.0008    0.0012    0.0004    0.0212   -0.0224    0.0037
    0.0012   -0.0025    0.0037    0.0212   -0.0195    0.0008    0.0008
ck =
    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000
»
»
»
»
» n=4, M=magic(n)+i*hilb(n), W=inv_fcz(M), ck=W*M,

 n =
     4
 M =
  16.0000 + 1.0000i   2.0000 + 0.5000i   3.0000 + 0.3333i  13.0000 + 0.2500i
   5.0000 + 0.5000i  11.0000 + 0.3333i  10.0000 + 0.2500i   8.0000 + 0.2000i
   9.0000 + 0.3333i   7.0000 + 0.2500i   6.0000 + 0.2000i  12.0000 + 0.1667i
   4.0000 + 0.2500i  14.0000 + 0.2000i  15.0000 + 0.1667i   1.0000 + 0.1429i
 W =
   0.0285 - 0.5739i  -0.0849 - 1.7212i   0.0130 + 1.7216i   0.0336 + 0.5738i
  -0.1084 - 1.7210i   0.4432 - 5.1654i  -0.1881 + 5.1648i  -0.2349 + 1.7220i
   0.0868 + 1.7212i  -0.4125 + 5.1657i   0.1869 - 5.1658i   0.2859 - 1.7224i
  -0.0166 + 0.5740i  -0.0340 + 1.7217i   0.1353 - 1.7225i  -0.0160 - 0.5743i
ck =
   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
»
»
»
»
» n=150, M=randn(n)+i*rand(n); W=inv_fcz(M); erck=norm(W*M-eye(n)),

 n =
    150
 erck =
    2.4075e-011
»
```

V. CONCLUSION

The stability and speed of computation to get the desired matrix inverse in an algorithm is related to the total number of elementary multiplication/division operations. For a given square matrix of order n, the total number of EMDO used for the presented compact code is computed to be $(n^4+n^3+2n^2+2n)/2$.

It is also worth noting that the line inside the for loop of this function may also be replaced by

```
W(K,N)=(I(K,N)-W(K,k)*M(k,N)/(1+M(k,N)*W(N,k)))*W.
```

MATLAB simulations show that these algorithms are valid.

Acknowledgement is given for the useful discussions and comments provided by Dr. F.C. Chang, Allwave Corporation, California.

REFERENCES

[1]. F. C. Chang, "Computation of the Inverse of Square Matrix," IEEE Antennas and Propagation Magazine, vol.63, no.4, pp.128-130, Aug. 2021.
[2]. F. C. Chang, "Inversion of a perturbed matrix," Appl.Math.Letters, vol.19, pp.163-173, 2006.
[3]. F. C. Chang, "Inverse and Determinant of a Square Matrix by Order Expansion and Condensation," IEEE Antennas and Propagation Magazine, vol.57, no.1, pp.28-32, Feb. 2015.
[4]. J. Cao & X. Wang, "New Recursive Algorithm for Matrix Inversion," J. Systems Engineering and Elctronics, vol.19(2), pp.381-384, 2008.
[5]. K. Aydin & G. Celik Kizikan, "Iterative Algorithm for Perturbed Matrix," Fen Dergisi (E-Dergi), vol, 3(1), pp.107-112, 2008.