British Journal of Mathematics & Computer Science
12(4): 1-21, 2016, Article no.BJMCS.21918
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Matrix Inverse as by-Product of Determinant

Feng Cheng Chang[1*]

[1]*Allwave Corporation, Torrance, California, USA.*

Article Information

DOI: 10.9734/BJMCS/2016/21918
Editor(s):
(1) Feyzi Basar, Department of Mathematics, Fatih University, Turkey.
Reviewers:
(1) Jianchao Bai, Xi'an Jiaotong University, China.
(2) G. Y. Sheu, Chang-Jung Christian University, Tainan, Taiwan.
(3) Grienggrai Rajchakit, Mae jo University, Thailand.
Complete Peer review History: http://sciencedomain.org/review-history/11983

Original Research Article

Received: 09 September 2015
Accepted: 06 October 2015
Published: 26 October 2015

Abstract

The determinant of a given square matrix is obtained as the product of pivot elements evaluated via the iterative matrix order condensation. It follows as the by-product that the inverse of this matrix is then evaluated via the iterative matrix order expansion. The fast and straightforward basic iterative procedure involves only simple elementary arithmetical operations without any high mathematical process. Remarkably, the revised optimal iterative process will compute without failing the inverse of any square matrix within minutes, be it real or complex, singular or nonsingular, and amazingly enough even for size as huge as 999x999. The manually extended iteration process is also developed to shorten the iteration process steps.

Keywords: Determinant; matrix inversion; matrix multiplication; recursive algorithm; matrix order expansion; matrix order condensation.

1 Introduction

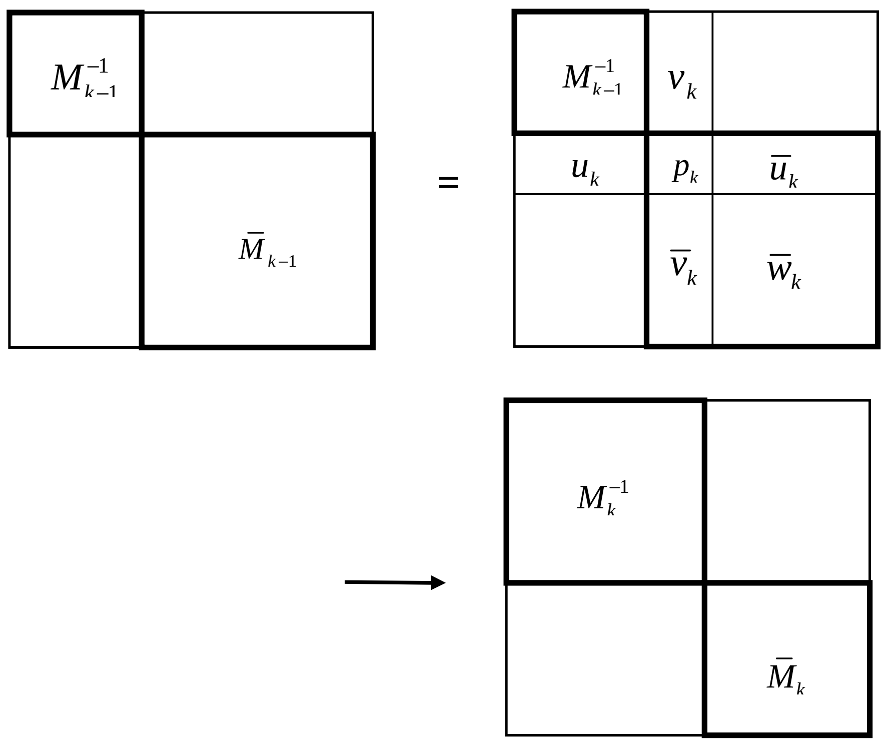
For any given square matrix, a set of pivot elements (known as Schur complements) is computed via the basic iterative algorithm of matrix order condensation. It follows that the determinant is obtained as the product of all pivot elements, and as the by-product of the procedure, the inverse of the matrix is evaluated via the iterative algorithm of matrix order expansion. The optimal iterative algorithm is then derived to give the smooth computational process. Finally, the extended iterative algorithm is developed to further reduce the iteration steps.

*Corresponding author: E-mail: fcchang007@yahoo.com;

2 Formulation for Algorithms

2.1 Algorithm for Basic Iteration

For a given square matrix $[M]$ of order *N*, its inverse $[M^{-1}]$ and determinant $\det[M]$ can be evaluated as follows by an iterative algorithm relying upon matrix order condensation and order expansion. Details of the basic iteration process from the (*k* -1)-th step to the *k*-th step, *k* = 1, 2, …. , *N*, are illustrated in Fig. 1.



$$[\bar{M}_{k-1}] = \begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{w}_k \end{bmatrix}$$

$$[\bar{M}_k] = \left[\bar{w}_k - \bar{v}_k p_k^{-1} \bar{u}_k \right]$$

$$[M_k] = \begin{bmatrix} M_{k-1} & v_k \\ u_k & d_k \end{bmatrix} \qquad d_k = p_k + u_k M_k^{-1} v_k$$

$$[M_k^{-1}] = \begin{bmatrix} M_{k-1}^{-1} + M_{k-1}^{-1} v_k p_k^{-1} u_k M_{k-1}^{-1} & -M_{k-1}^{-1} v_k p_k^{-1} \\ -p_k^{-1} u_k M_{k-1}^{-1} & p_k^{-1} \end{bmatrix}$$

Fig. 1. Iteration process from the (*k* -1)-th step to the *k*-th step

(1). ***Matrix order condensation*****:**

Assign $[\bar{M}_0]=[M]$ at the beginning of iterative process. At the k-th step of the iterative process, the condensed matrix $[\bar{M}_k]$ of order $(N-k)$, located at the lower right corner, is evaluated from its precursor $[\bar{M}_{k-1}]$ of order $(N-k-1)$ as,

$$\left[\bar{M}_{k-1}\right] = \begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{W}_k \end{bmatrix}.$$

$$\left[\bar{M}_k\right] = \left[\bar{W}_k - \bar{v}_k\, p_k^{-1}\,\bar{u}_k\right].$$

where p_k is the desired pivot element. The determinant of $[\bar{M}_{k-1}]$ is likewise produced recursively on noting that

$$\det[\bar{M}_{k-1}] = \det\begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{W}_k \end{bmatrix} = p_k \cdot \det[\bar{M}_k].$$

(2). ***Matrix order expansion*****:**

At the k-th step of the iterative process, the expanded matrix $[M_k]$ of order k, located at the upper left corner, is evaluated from its precursor $[M_{k-1}]$ of order $(k-1)$ by annexing the two sub-matrices v_k and u_k, and the single entry d_k,

$$[M_k] = \begin{bmatrix} M_{k-1} & v_k \\ u_k & d_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & v_k \\ u_k & p_k + u_k M_{k-1}^{-1} v_k \end{bmatrix}$$

The inverse of the matrix $[M_k]$ of order k is then determined as

$$[M_k]^{-1} = \begin{bmatrix} M_{k-1} & v_k \\ u_k & p_k + u_k M_{k-1}^{-1} v_k \end{bmatrix}^{-1} = \begin{bmatrix} M_{k-1}^{-1} + M_{k-1}^{-1} v_k p_k^{-1} u_k M_{k-1}^{-1} & -M_{k-1}^{-1} v_k p_k^{-1} \\ -p_k^{-1} u_k M_{k-1}^{-1} & p_k^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} M_{k-1}^{-1} & \\ & 0 \end{bmatrix} + \begin{bmatrix} -M_{k-1}^{-1} v_k \\ 1 \end{bmatrix} \left[\, p_k^{-1} \right] \left[-u_k M_{k-1}^{-1} \quad 1 \right]$$

wherein the pivot element p_k is related to entry d_k by $p_k = d_k - u_k M_{k-1}^{-1} v_k$, and can be readily obtained directly in the earlier condensation process.

***Proof* :**

(1). Condensation Process:

Since

$$\left[\bar{M}_{k-1}\right]=\begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{w}_k \end{bmatrix}=\begin{bmatrix} 1 & \\ \bar{v}_k p_k^{-1} & I_k \end{bmatrix}\begin{bmatrix} p_k & \\ & \bar{w}_k-\bar{v}_k p_k^{-1}\bar{u}_k \end{bmatrix}\begin{bmatrix} 1 & p_k^{-1}\bar{u}_k \\ & I_k \end{bmatrix}$$

and

$$\left[\bar{M}_k\right]=\left[\bar{w}_k-\bar{v}_k p_k^{-1}\bar{u}_k\right]$$

we have

$$\det\left[\bar{M}_{k-1}\right]=\det\left(\begin{bmatrix} 1 & \\ \bar{v}_k p_k^{-1} & I_k \end{bmatrix}\begin{bmatrix} p_k & \\ & \bar{w}_k-\bar{v}_k p_k^{-1}\bar{u}_k \end{bmatrix}\begin{bmatrix} 1 & p_k^{-1}\bar{u}_k \\ & I_k \end{bmatrix}\right)=\det\begin{bmatrix} p_k & \\ & \bar{M}_k \end{bmatrix}$$
$$= p_k\cdot\det\left[\bar{M}_k\right]$$

(2). Expansion Process:

Since

$$\left[M_k\right]=\begin{bmatrix} M_{k-1} & v_k \\ u_k & d_k \end{bmatrix}=\begin{bmatrix} M_{k-1} & v_k \\ u_k & p_k+u_k M_{k-1}^{-1}v_k \end{bmatrix}=\begin{bmatrix} I_k & \\ u_k M_{k-1}^{-1} & 1 \end{bmatrix}\begin{bmatrix} M_{k-1} & \\ & p_k \end{bmatrix}\begin{bmatrix} I_k & M_{k-1}^{-1}v_k \\ & 1 \end{bmatrix}$$

we have

$$\left[M_k\right]^{-1}=\left(\begin{bmatrix} I_k & \\ u_k M_{k-1}^{-1} & 1 \end{bmatrix}\begin{bmatrix} M_{k-1} & \\ & p_k \end{bmatrix}\begin{bmatrix} I_k & M_{k-1}^{-1}v_k \\ & 1 \end{bmatrix}\right)^{-1}=\begin{bmatrix} I_k & -M_{k-1}^{-1}v_k \\ & 1 \end{bmatrix}\begin{bmatrix} M_{k-1}^{-1} & \\ & p_k^{-1} \end{bmatrix}\begin{bmatrix} I_k & \\ -u_k M_{k-1}^{-1} & 1 \end{bmatrix}$$
$$=\begin{bmatrix} M_{k-1}^{-1}+M_{k-1}^{-1}v_k p_k^{-1}u_k M_0^{-1} & -M_{k-1}^{-1}v_k p_k^{-1} \\ -p_k^{-1}u_k M_{k-1}^{-1} & p_k^{-1} \end{bmatrix}$$

and

$$\det\left[M_k\right]=\det\left(\begin{bmatrix} I_k & \\ u_k M_{k-1}^{-1} & 1 \end{bmatrix}\begin{bmatrix} M_{k-1} & \\ & p_k \end{bmatrix}\begin{bmatrix} I_k & M_{k-1}^{-1}v_k \\ & 1 \end{bmatrix}\right)=\det\left[M_{k-1}\right]\cdot p_k$$

(*end of proof*)

The determinant and the inverse of the original matrix $[M]=[\bar{M}_0]=[M_N]$ are therefore, respectively, found after performing N steps of the matrix order condensation process and the matrix order expansion process:

$$\det[M] \quad = \quad p_1\cdot p_2\cdot \cdots \cdot p_N\,,$$

and

$$[M]^{-1}=[M_N^{-1}]$$

2.2 Optimal and Extended Iterations

It is noted that the basic iteration algorithm is quite straightforward, with pivot elements selected along the diagonal regardless of their magnitudes. However, this basic process could fail should the magnitudes of some among such pivots shrink to zero or else jump toward huge numbers, with the result that the determinant, obtained as their product, would eventually emerge with an erroneous value.

A modified optimal iteration algorithm can then be developed so as to resolve this potential problem. At every iteration step a pivot is picked as the element of maximum magnitude among all possible locations, this pivot element is then subsequently brought into diagonal position by having the rows and columns permuted accordingly. Past this row/column rearrangement step the modified algorithm is in all respects identical to its basic precursor. The desired inverse of the original matrix is therefore obtained once rows and columns are restored to their original locations.

In summary: (1) if the given matrix is non-singular then its determinant is found as the product of all its pivot elements; and (2) the matrix is said to be singular in the event that pivot elements shrink steeply toward zero.

Finally, it is interesting to note that in the basic iteration process the number of steps can be somewhat reduced by replacing each individual pivot element manually by a square pivot block (not necessary solid) of any size, provided only that this block has an inverse which can be easily computed. Picture illustration of extension process with related blocks is shown in Fig. 2. In keeping with what was previously suggested, this extended iteration process may fail should any pivot block get out to be singular.

$$\det\begin{bmatrix} | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \\ + & - & + & - & - & - & - & + & - \\ | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \\ + & - & + & - & - & - & - & + & - \\ + & - & + & - & - & - & - & + & - \\ | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \\ | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \\ | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \\ | & \cdot & | & \cdot & \cdot & \cdot & \cdot & | & \cdot \end{bmatrix} \qquad r = [\,2\ \ 4\ \ 5\,],\quad c = [\,1\ \ 3\ \ 8\,]$$

$$= \ (-1)^{(r+c)} \cdot \det\begin{bmatrix} + & + & + \\ + & + & + \\ + & + & + \end{bmatrix} \cdot \det\left(\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} - \begin{bmatrix} | & | & | \\ | & | & | \\ | & | & | \\ | & | & | \\ | & | & | \\ | & | & | \end{bmatrix} \begin{bmatrix} + & + & + \\ + & + & + \\ + & + & + \end{bmatrix}^{-1} \begin{bmatrix} - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{bmatrix} \right)$$

or

$$\det\begin{bmatrix} P & V \\ U & W \end{bmatrix} \ = \ s \cdot \det[P] \cdot \det\left([W] - [V]\,[P]^{-1}\,[U]\right)$$

$\uparrow$ here P, V, U, and W may not be all solid blocks.

Fig. 2. Picture illustration of extended iteration

3 Computer Routines

The MATLAB routines, derived from the basic iteration process and its optimal modification, as well as manually extended pivot process are presented below.

(1) ***Basic scheme***

```
function [detM,invM,p] = det_inv_o(M)
%   The determinant the inverse of a given matrix are
%     found by matrix order condensation and expansion.
%      ---- Basic Scheme.
%   F.C. Chang   09/11/2015.

       N = size(M,1);  nM = M; iM = [ ]; p = [ ];
   for  k = 1:N,   n = N-k+1;
       P = nM(1,1);  iP = 1/P;   p = [p,P];
      nM = nM(2:n,2:n)-nM(2:n,1)*iP*nM(1,2:n);
      iM = blkdiag(iM,0)+[-iM*M(1:k-1,k);1] ...
         *[iP]*[-M(k,1:k-1)*iM,1];    % k,P,nM,iM,
   end;
      detM = prod(p);  invM = iM;    % p,detM,invM,
```

(2) ***Optimal scheme***

```
function [detM,invM,p,s,rc] = det_inv_p(M)
%   The determinant of a given matrix and an array of
%      pivots are found by matrix order condensation.
%      Then as by-product the inverse matrix is obtained
%      by matrix order condensation. ---- Optimal Scheme.
%   F.C. Chang   09/11/2015.

      k = 0;  N = size(M,1); mA = max(abs(M(:)));   % N,M,
      nM = M;  mM = [ ];  iM = [ ];          % k,nM,mM,iM,
      s = 1;   r = [ ];  c = [ ];  p = [ ];
   for  k = 1:N,   n = N-k+1;
      [mp,rcm] = max(abs(nM(:)));
      [ri,ci] = ind2sub([n,n],rcm);   rci = [ri;ci];
      rx = setdiff([1:n],ri); cx = setdiff([1:n],ci);
      ro = setdiff([1:N],r);  co = setdiff([1:N],c);
      r = [r,ro(ri)];  c = [c,co(ci)];   rc = [r;c];
      P = nM(rcm);   iP = 1/P;   p = [p,P];
    if mp/mA < 1.e-10, disp('Given matrix is singular !');
      detM = 0; invM = NaN;  p; s; rc;  return, end;
      s = s*(-1)^(ri+ci);                 % k,rci,rc;P,iP,
      nM = nM(rx,cx)-nM(rx,ci)*iP*nM(ri,cx);
      mM = M(r,c);
      iM = blkdiag(iM,0)+[-iM*mM(1:k-1,k);1] ...
         *iP*[-mM(k,1:k-1)*iM,1];             % nM,mM,iM,
   end;
      detM = prod(p)*s; invM(c,r) = iM; % rc,s,p,detM,invM,
```

(3) ***Extended scheme***

```
function [detM,invM,p,s,rc] = det_inv_LL(M)
%    Find the determinant via order condensation and then
%      as by-product the inverse via order expansion.
%      Manually select rows/columns for pivot blocks.
%      ----- Expanded Scheme.
%    F C Chang     09/11/15
```

```
    k = 0; N = size(M,1); nM = M; mM = [ ]; iM = [ ];
    r = [ ]; c = [ ]; p = [ ]; s = 1; t(1) = 0;   k,nM,mM,
for  k = 1:N,
    n = size(nM,1); m = size(mM,1); if n == 0, break,end;
    disp('Select rows and columns from nM ');
    rci = input('[ri;ci]  =  ');
    ri = sort(rci(1,:));     ci = sort(rci(2,:));
    rx = setdiff([1:n],ri);  cx = setdiff([1:n],ci);
    ro = setdiff([1:N],r);   co = setdiff([1:N],c);
    r = [r,ro(ri)];          c = [c,co(ci)];
    rci = [ri;ci];    rcx =[rx;cx];     rc = [r;c];
    s = s*(-1)^sum(ri+ci);
    P = nM(ri,ci);    L = size(P,1);
    iP = inv(P);   d = det(P);    p = [p,d];
    t(k+1) = t(k)+L; rt = [1:t(k)]; ct = [t(k)+1:t(k+1)];
    nM = nM(rx,cx)-nM(rx,ci)*iP*nM(ri,cx);
    mM = M(r,c);
    iM = blkdiag(iM,zeros(L,L))+[-iM*mM(rt,ct);eye(L)] ...
       *iP*[-mM(ct,rt)*iM,eye(L)];      k,P,d,iP,nM,mM,iM,
end;
    detM = prod(p)*s; invM(c,r) = iM;     rc,s,p,detM,invM,
```

Remark:

Given a matrix M of order N. At the k-th step of iteration process, the condensed matrix M_k, the expanded matrix M_k, and its inverse M_k^{-1} are, respectively, denoted as *nM*, *mM*, and *iM* in the given MATLAB routines. The pivot element P_k and its inverse P_k^{-1} at the k-th step are, likewise, denoted as P and *iP*, respectively. Also, the overall rows/columns rearrangement is expressed as [r c].

Outputs of the routines give only the desired final results and skip all intermediate related data in order to save space in case that the given matrix order N is very huge. By removing any %'s at appropriate locations in these routine, the expected related intermediate data will appear in the processing output.

The validation of output results may be performed by checking if the multiplication of the computed inverse matrix and the given original matrix is equal to an identity matrix of the same size within permitted error.

Please refer to Numerical Illustrations Section and Appendix for more detail.

4 Numerical Illustrations

```
>> N=5, M=magic(5),
     N =
        5
     M =
       17    24     1     8    15
       23     5     7    14    16
        4     6    13    20    22
       10    12    19    21     3
       11    18    25     2     9

>> [detM,invM,p]=det_inv_o(M),
     detM =
       5070000
```

```
    invM =
      -0.0049    0.0512   -0.0354    0.0012    0.0034
       0.0431   -0.0373   -0.0046    0.0127    0.0015
      -0.0303    0.0031    0.0031    0.0031    0.0364
       0.0047   -0.0065    0.0108    0.0435   -0.0370
       0.0028    0.0050    0.0415   -0.0450    0.0111
    p =
      17.0000  -27.4706   12.8373   -9.3786   90.1734
```

>> **[detM,invM,p,s,rc]=det_inv_p(M)**

```
    detM =
       5070000
    invM =
      -0.0049    0.0512   -0.0354    0.0012    0.0034
       0.0431   -0.0373   -0.0046    0.0127    0.0015
      -0.0303    0.0031    0.0031    0.0031    0.0364
       0.0047   -0.0065    0.0108    0.0435   -0.0370
       0.0028    0.0050    0.0415   -0.0450    0.0111
    p =
      25.0000   23.2800   20.1031  -22.1667   19.5489
    s =
       -1
    rs =
        5     1     3     4     2
        3     2     4     5     1
```

>> **[detM,invM,p,s,rs]=det_inv_LL(M)**

```
    N =
        5
    M =
       17    24     1     8    15
       23     5     7    14    16
        4     6    13    20    22
       10    12    19    21     3
       11    18    25     2     9
    k =
        0
    nM =
       17    24     1     8    15
       23     5     7    14    16
        4     6    13    20    22
       10    12    19    21     3
       11    18    25     2     9
    mM =
         [ ]
    k =
        1
    Select rows and columns from nM
    [ri;ci]  =
         [1 2 3; 2 4 5]
    P =
          24     8    15
           5    14    16
           6    20    22
```

```
d =
      -160
iP =
     0.0750      -0.7750       0.5125
     0.0875      -2.7375       1.9313
    -0.1000       2.7000      -1.8500
nM =
      1153.8     -107.66
       -97.5      36.562
mM =
        24          8          15
         5         14          16
         6         20          22
iM =
     0.0750      -0.7750       0.5125
     0.0875      -2.7375       1.9313
    -0.1000       2.7000      -1.8500
k =
    2
Select rows and columns from nM
[ri;ci]  =
      [1 2; 1 2]
P =
      1153.8     -107.66
       -97.5      36.562
d =
       31687
iP =
   0.0011538    0.0033974
   0.0030769      0.03641
nM =
        [ ]
mM =
         24         8        15        17         1
          5        14        16        23         7
          6        20        22         4        13
         12        21         3        10        19
         18         2         9        11        25
iM =
     0.0431    -0.0373    -0.0046     0.0127     0.0015
     0.0047    -0.0065     0.0108     0.0435    -0.0370
     0.0028     0.0050     0.0415    -0.0450     0.0111
    -0.0049     0.0512    -0.0354     0.0012     0.0034
    -0.0303     0.0031     0.0031     0.0031     0.0364

              --------------
detM =
     5070000
invM =
    -0.0049     0.0512    -0.0354     0.0012     0.0034
     0.0431    -0.0373    -0.0046     0.0127     0.0015
    -0.0303     0.0031     0.0031     0.0031     0.0364
     0.0047    -0.0065     0.0108     0.0435    -0.0370
     0.0028     0.0050     0.0415    -0.0450     0.0111
```

```
p =
     -160.0      31687.5
s =
       -1
rc =
        1          2          3          4          5
        2          4          5          1          3
```

>> [detM,invM,p,s,rc]=det_inv_LL(M),

```
k =
    0
nM =
   17    24     1     8    15
   23     5     7    14    16
    4     6    13    20    22
   10    12    19    21     3
   11    18    25     2     9
mM =
    []
Select rows and columns from nM
 [ri;ci]  =  [1 5; 1 5]
k =
    1
P =
   17    15
   11     9
d =
  -12
iP =
           -0.75         1.25
     0.91667      -1.4167
nM =
           -42.5       -142.5         22.5
             65          650         -65
           -22.5       -182.5         42.5
mM =
   17    15
   11     9
iM =
           -0.75         1.25
     0.91667      -1.4167
Select rows and columns from nM
 [ri;ci]  =  [1 3; 1 3]
k =
    2
P =
           -42.5         22.5
           -22.5         42.5
d =
           -1300
iP =
   -0.032692     0.017308
   -0.017308     0.032692
nM =
             325
```

```
mM =
   17    15    24     8
   11     9    18     2
   23    16     5    14
   10     3    12    21
iM =
    -0.35288      0.42212     0.086538     0.036538
    0.41122     -0.48045    -0.036538    -0.086538
    -0.0022436      0.05609    -0.032692     0.017308
    0.11058     -0.16442    -0.017308     0.032692
Select rows and columns from nM
 [ri;ci]  =  [1; 1]
k =
    3
P =
             325
d =
             325
iP =
   0.0030769
nM =
    []
mM =
   17    15    24     8     1
   11     9    18     2    25
   23    16     5    14     7
   10     3    12    21    19
    4    22     6    20    13
iM =
  -0.0049359    0.0033974     0.051154    0.0011538    -0.035385
   0.0027564      0.01109        0.005       -0.045     0.041538
    0.043141    0.0014744    -0.037308     0.012692   -0.0046154
   0.0046795    -0.036987   -0.0065385     0.043462     0.010769
   -0.030256      0.03641    0.0030769    0.0030769    0.0030769
------------------------------------
detM =
   5070000
invM =
  -0.0049359     0.051154    -0.035385    0.0011538    0.0033974
    0.043141    -0.037308   -0.0046154     0.012692    0.0014744
   -0.030256    0.0030769    0.0030769    0.0030769      0.03641
   0.0046795   -0.0065385     0.010769     0.043462    -0.036987
   0.0027564        0.005     0.041538       -0.045      0.01109
p =
             -12    -1300      325
s =
    1
rc =
    1     5     2     4     3
    1     5     2     4     3
```

```
>> N=5, M=magic(N)/10e+9; [detM,invM,p,s,rc]=det_inv_p(M);
       detM,p, erM=norm(M*invM-eye(N)),
   N =
      5
   detM =
     5.07e-044
   p =
      2.5e-009   2.328e-009  2.0103e-009 -2.2167e-009  1.9549e-009
   erM =
    4.5777e-016
>> N=8, M=magic(N), [detM,invM,p,s,rs]=det_inv_p(M),
   N =
       8
   M =
       64    2    3   61   60    6    7   57
        9   55   54   12   13   51   50   16
       17   47   46   20   21   43   42   24
       40   26   27   37   36   30   31   33
       32   34   35   29   28   38   39   25
       41   23   22   44   45   19   18   48
       49   15   14   52   53   11   10   56
        8   58   59    5    4   62   63    1
 Given matrix is singular!
   detM =
       0
   invM =
      NaN
   p =
      64   62.125 12.817 -7.4226e-015     ---
   s =
       1
   rs =
       1     8     2     5      ---
       1     7     8     2      ---
>> N=11, M=magic(N); [detM,invM,p]=det_inv_o(M); detM,p,
   N =
       11
   detM =
      NaN
   p =
     68.000   -2.2941     0     -Inf     NaN     NaN
      NaN      NaN      NaN     NaN     NaN
```

---→ ***No good*!**

```
>> N=11, M=magic(N); [detM,invM,p,s,rs]=det_inv_p(M); detM,p,
   N =
      11
   detM =
     -4.1038e+022
p =
       121.0000  119.1074  109.3497  110.7177  111.5512  112.5070
       114.6380  109.3049  107.1374  117.2489  119.0564
```

---→ ***Good*!**

```
>> N=3, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M),
      N =
         3
      detM =
         1.2424  + 2.6826i
      invM =
        0.031601  + 0.11641i    0.0758   - 0.42246i    0.099138 -0.13227i
       -0.0066331 - 0.62526i   -0.25293  - 0.075866i   0.6305  - 0.20649i
        0.029416  + 0.16779i   -0.040886 + 0.82601i   -0.11191 - 0.73749i
      p =
        0.4617   + 1.4524i     0.12257  + 1.5995i    -0.059778 - 1.2077i
      s =
         1
      rs =
         3     1     2
         1     2     3

>> N=9; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
>>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
      N =
         9
      elapsed_time =
             0.04
      erM =
        2.8833e-015

>> N=99; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
>>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
      N =
        99
      elapsed_time =
            0.671
      erM =
        1.786e-012

>> N=555; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
>>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
      N =
        555
      elapsed_time =
         109.61
      erM =
        1.629e-011

>> N=999; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
>>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
      N =
        999

elapsed_time =
         734.84
      erM =
        1.933e-010
```

5 Conclusion

A simple approach has been developed for finding the inverse and determinant of any square matrix, real or complex at will. The process involves successive applications of an algorithm for matrix order condensation as well as order expansion. It is then optimized so as to accommodate the situation wherein the intermediate computations have begun to suggest that the given matrix may in fact be nearly singular. The manually extended iteration process is also developed to shorten the iteration steps, if the calculation of small size inverse matrices is feasible.

When compared to various other methods available in the literature [1-8], the iteration process schemes presented are very compact, efficient, straightforward, and involves only the simple elementary arithmetical operations, such as addition, subtraction, multiplication, and division. It dose not involve any high mathematics at all.

It is shown that for a given *N* x *N* matrix, the number of multiplication/division operations needed to create a set of *N* pivot elements and their reciprocals are $(\frac{1}{3}N^3+\frac{2}{3}N)$, which includes overall *N* division operations. It follows applying these computed results the number of multiplications required to compute the determinant and inversion of this given *N* x *N* matrix are $(N-1)$ and $(N^3-\frac{1}{2}N^2-\frac{3}{2}N)$, respectively. The overall operations for determinant and matrix inversion are thus $(\frac{4}{3}N^3-\frac{1}{2}N^2+\frac{1}{6}N-1)$. Noted that N^3 is the total number of multiplications needed to compute the product of any two *N* x *N* matrices!

Numerical illustrations confirm that the optimized iteration process, embodied in few lines of code utilizing only elementary arithmetical operations, computes the inverse of any square matrices, real or complex, singular or nonsingular, without fail within minutes, and, amazingly enough, even for a size as huge as 999x999.

Acknowledgements

The author would also like to acknowledge the comments and useful discussions with Dr. George Cheng, Dr. Jan Grzesik, Dr. Yong Zhu, Ms. Lala Zhu, Mr. Felix Wong and Mr. Jene Wu of Allwave Corporation.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Sherman J, Morrison WJ. Adjustment of inverse matrix corresponding to changes in a given column or a given row of the original matrix. Ann. Math. Stat. 1949;75:124.

[2] Wilf HS. Matrix inversion by the annihilation of rank. J. Soc. Indust. Appl. Math. 1959;7(2): 149-151.

[3] Asif A, Moura JMF. Block matrices with L-block-banded inverse: Inversion Algorithm. IEEE Trans. on Signal Processing. 2005;53(2):630-642.

[4] Chang FC. Inverse and determinant of a square matrix by order expansion and condensation. IEEE Antenna and Propagation Magazine. 2015;57(1):28-32.

[5] Jianshu C, Wang X. New recursive algorithm for matrix inversion. J. Systems Engineering and Electronics. 2008;19(2):381-384.

[6] Aydin K, Celik Kizilkan G. Iterative inverse algorithm for perturbed matrix. SDU Science Journal. 2008;3(1):107-112.

[7] Su CT, Chang FC. Quick evaluation of determinant. Appl. Math. & Comput. 1996;75:117-118.

[8] Chang FC. Determinant of matrix by order condensation. British J. of Math. & Comput. Science. 2014;4(13):1843-1848.

APPENDIX

Notes: Present the detail printout of intermediate steps in running the basic and optimal iteration process after removing '%'s in the appropriate locations of the MATLAB routines.

>> diary on

>> format short

>> M=magic(5),

```
M =
   17   24    1    8   15
   23    5    7   14   16
    4    6   13   20   22
   10   12   19   21    3
   11   18   25    2    9
```

>> [detM,invM,p]=det_inv_o(M);

```
k =
    1
P =
   17
nM =
  -27.4706    5.6471    3.1765   -4.2941
    0.3529   12.7647   18.1176   18.4706
   -2.1176   18.4118   16.2941   -5.8235
    2.4706   24.3529   -3.1765   -0.7059
iM =
    0.0588
----------
k =
    2
P =
  -27.4706
nM =
   12.8373   18.1585   18.4154
   17.9764   16.0493   -5.4925
   24.8608   -2.8908   -1.0921
iM =
   -0.0107    0.0514
    0.0493   -0.0364
----------
k =
    3
P =
   12.8373
nM =
   -9.3786  -31.2802
  -38.0567  -36.7556
```

```
iM =
   -0.0038    0.0510   -0.0272
    0.0452   -0.0362    0.0160
   -0.0197    0.0010    0.0779
----------
k =
     4
P =
   -9.3786
nM =
   90.1734
iM =
   -0.0058    0.0496   -0.0481    0.0149
    0.0428   -0.0380   -0.0101    0.0187
   -0.0393   -0.0133   -0.1333    0.1508
    0.0139    0.0101    0.1493   -0.1066
----------
k =
     5
P =
   90.1734
nM =
     []
iM =
   -0.0049    0.0512   -0.0354    0.0012    0.0034
    0.0431   -0.0373   -0.0046    0.0127    0.0015
   -0.0303    0.0031    0.0031    0.0031    0.0364
    0.0047   -0.0065    0.0108    0.0435   -0.0370
    0.0028    0.0050    0.0415   -0.0450    0.0111
--------------------
detM =
  5.0700e+006
invM =
   -0.0049    0.0512   -0.0354    0.0012    0.0034
    0.0431   -0.0373   -0.0046    0.0127    0.0015
   -0.0303    0.0031    0.0031    0.0031    0.0364
    0.0047   -0.0065    0.0108    0.0435   -0.0370
    0.0028    0.0050    0.0415   -0.0450    0.0111
p =
   17.0000  -27.4706   12.8373   -9.3786   90.1734
```

>> [detM,invM,p,s,rc]=det_inv_p(M),

```
N =
     5
M =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
```

```
    11    18    25     2     9
-------
k =
     0
nM =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
mM =
     [ ]
iM =
     [ ]
-------
k =
     1
rci =
     5     3
P =
    25
iP =
    0.0400
nM =
   16.5600   23.2800    7.9200   14.6400
   19.9200   -0.0400   13.4400   13.4800
   -1.7200   -3.3600   18.9600   17.3200
    1.6400   -1.6800   19.4800   -3.8400
mM =
    25
iM =
    0.0400
-------
k =
     2
rci =
     1     2
P =
   23.2800
iP =
    0.0430
nM =
   19.9485   13.4536   13.5052
    0.6701   20.1031   19.4330
    2.8351   20.0515   -2.7835
mM =
    25    18
     1    24
iM =
    0.0412   -0.0309
```

```
  -0.0017    0.0430
-------
k =
     3
rci =
     2     2
P =
   20.1031
iP =
    0.0497
nM =
   19.5000    0.5000
    2.1667  -22.1667
mM =
    25    18     2
     1    24     8
    13     6    20
iM =
    0.0369   -0.0297    0.0082
    0.0072    0.0405   -0.0169
   -0.0262    0.0072    0.0497
-------
k =
     4
rci =
     2     2
P =
  -22.1667
iP =
   -0.0451
nM =
   19.5489
mM =
    25    18     2     9
     1    24     8    15
    13     6    20    22
    19    12    21     3
iM =
    0.0362   -0.0300    0.0052    0.0030
    0.0040    0.0395   -0.0304    0.0135
   -0.0366    0.0040    0.0062    0.0436
    0.0108    0.0032    0.0450   -0.0451
-------
k =
     5
rci =
     1     1
P =
   19.5489
```

```
iP =
   0.0512
nM =
    [ ]
mM =
   25    18     2     9    11
    1    24     8    15    17
   13     6    20    22     4
   19    12    21     3    10
    7     5    14    16    23
iM =
   0.0364   -0.0303    0.0031    0.0031    0.0031
   0.0015    0.0431   -0.0046    0.0127   -0.0373
  -0.0370    0.0047    0.0108    0.0435   -0.0065
   0.0111    0.0028    0.0415   -0.0450    0.0050
   0.0034   -0.0049   -0.0354    0.0012    0.0512
------------------------------
detM =
  5.0700e+006
invM =
  -0.0049    0.0512   -0.0354    0.0012    0.0034
   0.0431   -0.0373   -0.0046    0.0127    0.0015
  -0.0303    0.0031    0.0031    0.0031    0.0364
   0.0047   -0.0065    0.0108    0.0435   -0.0370
   0.0028    0.0050    0.0415   -0.0450    0.0111
p =
  25.0000   23.2800   20.1031  -22.1667   19.5489
s =
   -1
rc =
    5     1     3     4     2
    3     2     4     5     1
```

>> N=3, M=rand(N)+i*randn(N), [dM,iM]=det_inv_p(M),
>>   W=iM, [dW,iW]=det_inv_p(W), erD=dM-1/dW, erM=M-iW,

```
N =
    3
M =
  0.4447 + 0.1746i   0.9218 - 0.5883i   0.4057 + 0.1139i
  0.6154 - 0.1867i   0.7382 + 2.1832i   0.9355 + 1.0668i
  0.7919 + 0.7258i   0.1763 - 0.1364i   0.9169 + 0.0593i
dM =
  0.8840 + 1.9683i
iM =
  0.8843 + 0.2767i   0.0395 + 0.4206i   0.0463 - 0.7104i
  0.5856 + 0.5723i  -0.0536 - 0.1048i  -0.2642 - 0.1396i
 -0.8013 - 0.9100i   0.2988 - 0.4017i   0.5939 + 0.5261i
          --------------------
```

```
W =
  0.8843 + 0.2767i   0.0395 + 0.4206i   0.0463 - 0.7104i
  0.5856 + 0.5723i  -0.0536 - 0.1048i  -0.2642 - 0.1396i
 -0.8013 - 0.9100i   0.2988 - 0.4017i   0.5939 + 0.5261i
dW =
  0.1899 - 0.4228i
iW =
  0.4447 + 0.1746i   0.9218 - 0.5883i   0.4057 + 0.1139i
  0.6154 - 0.1867i   0.7382 + 2.1832i   0.9355 + 1.0668i
  0.7919 + 0.7258i   0.1763 - 0.1364i   0.9169 + 0.0593i
        --------------------
erD =
 3.3307e-016 -2.2204e-016i
erM =
 1.0e-015 *
 -0.0555 + 0.0000i   0.0000 - 0.1110i  -0.0555 - 0.1388i
 -0.1110 - 0.4441i   0.1110 + 0.0000i  -0.2220 + 0.0000i
  0.0000 - 0.1110i   0.0555 - 0.0833i  -0.1110 + 0.0139i
```

>> diary off

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/11983