

# 1 Original Research Article

## 2 **Matrix Inverse as by-product of Determinant**

### 6 **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 the inverse of this matrix is then readily computed 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 fail 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 finally developed to shorten the iteration process steps.

8  
9 *Keywords: Determinant; Matrix inversion; Matrix multiplication; Recursive algorithm; Matrix*  
10 *order expansion, Matrix order condensation.*

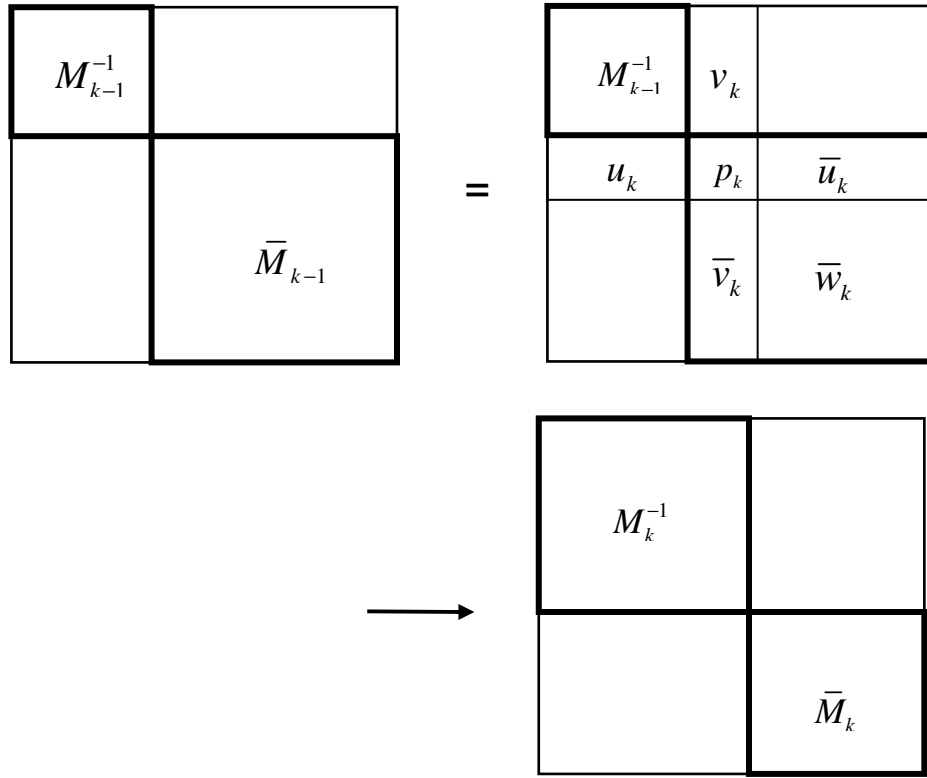
### 13 **1. INTRODUCTION**

14 For any given square matrix, sets of pivot elements (known as Schur complements) are  
15 evaluated via the basic iterative algorithm of matrix order condensation. It follows that the  
16 determinant is obtained as the product of all pivot elements, and as the by-product of the  
17 procedure, the inverse of the matrix is readily found via the iterative algorithm of matrix order  
18 expansion. The optimal iterative algorithm is then derived to give the smooth computational  
19 process. Finally, the extended iterative algorithm is developed to further reduce the iteration  
20 steps.

### 23 **2. FORMULATION FOR ALGORITHMS**

#### 24 **2.1 ALGORITHM FOR BASIC ITERATION**

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



$$\begin{aligned} \begin{bmatrix} \bar{M}_{k-1} \end{bmatrix} &= \begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{w}_k \end{bmatrix} \\ \begin{bmatrix} \bar{M}_k \end{bmatrix} &= \begin{bmatrix} \bar{w}_k - \bar{v}_k p_k^{-1} \bar{u}_k \end{bmatrix} \\ \begin{bmatrix} M_k \end{bmatrix} &= \begin{bmatrix} M_{k-1} & v_k \\ u_k & d_k \end{bmatrix} \quad d_k = p_k + u_k M_{k-1}^{-1} v_k \\ \begin{bmatrix} M_k^{-1} \end{bmatrix} &= \begin{bmatrix} M_{k-1}^{-1} + M_{k-1}^{-1} v_k p_k^{-1} u_k M_{k-1}^{-1} & -M_{k-1}^{-1} v_k p_k^{-1} \\ -p_k^{-1} u_k M_{k-1}^{-1} & p_k^{-1} \end{bmatrix} \end{aligned}$$

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

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

75 Assign  $[\bar{M}_0] = [M]$  at the beginning of iterative process. At the  $k$ -th step of the iterative  
 76 process, the condensed matrix  $[\bar{M}_k]$  of order  $(N - k)$ , located at the lower right corner, is  
 77 evaluated from its precursor  $[\bar{M}_{k-1}]$  of order  $(N - k - 1)$  as,  
 78

$$79 \quad [\bar{M}_{k-1}] = \begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{W}_k \end{bmatrix}.$$

$$80 \quad [\bar{M}_k] = [\bar{W}_k - \bar{v}_k p_k^{-1} \bar{u}_k].$$

81  
 82  
 83 where  $p_k$  is the desired pivot element. The determinant of  $[\bar{M}_{k-1}]$  is likewise produced  
 84 recursively on noting that  
 85

$$86 \quad \det[\bar{M}_{k-1}] = \det \begin{bmatrix} p_k & \bar{u}_k \\ \bar{v}_k & \bar{W}_k \end{bmatrix} = p_k \cdot \det[\bar{M}_k].$$

87

88

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

90

91 At the  $k$ -th step of the iterative process, the expanded matrix  $[M_k]$  of order  $k$ , located at the  
 92 upper left corner, is evaluated from its precursor  $[M_{k-1}]$  of order  $(k - 1)$  by annexing the two  
 93 sub-matrices  $v_k$  and  $u_k$ , and the single entry  $d_k$ ,  
 94

$$95 \quad [M_k] = \begin{bmatrix} M_{k-1} & v_k \\ u_k & d_k \end{bmatrix} = \begin{bmatrix} M_{k-1} & v_k \\ u_k & p_k + u_k M_{k-1}^{-1} v_k \end{bmatrix}$$

96

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

98

$$99 \quad [M_k]^{-1} = \begin{bmatrix} M_{k-1} & v_k \\ u_k & p_k + u_k M_{k-1}^{-1} v_k \end{bmatrix}^{-1} = \begin{bmatrix} M_{k-1}^{-1} + M_{k-1}^{-1} v_k p_k^{-1} u_k M_{k-1}^{-1} & -M_{k-1}^{-1} v_k p_k^{-1} \\ -p_k^{-1} u_k M_{k-1}^{-1} & p_k^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} M_{k-1}^{-1} & \\ & 0 \end{bmatrix} + \begin{bmatrix} -M_{k-1}^{-1} v_k \\ 1 \end{bmatrix} \begin{bmatrix} p_k^{-1} \end{bmatrix} \begin{bmatrix} -u_k M_{k-1}^{-1} & 1 \end{bmatrix}$$

100

101 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  
 102 readily obtained directly in the earlier condensation process.  
 103  
 104  
 105  
 106  
 107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

**Proof :**

(1). Condensation process:

Since

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

and

$$\begin{bmatrix} \bar{M}_k \end{bmatrix} = \begin{bmatrix} \bar{w}_k - \bar{v}_k p_k^{-1} \bar{u}_k \end{bmatrix}$$

110

we have

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

111

(2). Expansion process:

Since

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

we have

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

112

and

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

113

114

**(end of proof)**

115 The determinant and the inverse of the original matrix  $\begin{bmatrix} M \end{bmatrix} = \begin{bmatrix} \bar{M}_0 \end{bmatrix} = \begin{bmatrix} M_N \end{bmatrix}$  are therefore,

116 respectively, found after performing  $N$  steps of the matrix order condensation process and

117 the matrix order expansion process:

118

119

$$\det \begin{bmatrix} M \end{bmatrix} = p_1 \cdot p_2 \cdot \cdots \cdot p_N,$$

120

and

121

$$\cdot \begin{bmatrix} M \end{bmatrix}^{-1} = \begin{bmatrix} M_N^{-1} \end{bmatrix}$$

122

123

124

125

126

## 2.2 OPTIMAL AND EXTENDED ITERATIONS

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

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

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

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

148

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

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

149

150 or

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

151

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

152

153

154

155

156

### 3. COMPUTER ROUTINES

157

158

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

159

160 **(1) Basic scheme**

161

162 function [detM,invM,p] = det\_inv\_o(M)

163 % The determinant the inverse of a given matrix are

164 % found by matrix order condensation and expansion.

165 % ----- Basic Scheme.

166 % F.C. Chang 09/11/2015.

167

168 N = size(M,1); nM = M; iM = [ ]; p = [ ];

169 for k = 1:N, n = N-k+1;

170 P = nM(1,1); iP = 1/P; p = [p,P];

171 nM = nM(2:n,2:n)-nM(2:n,1)\*iP\*nM(1,2:n);

172 iM = blkdiag(iM,0)+[-iM\*M(1:k-1,k);1] ...

173 \*iP\*[-M(k,1:k-1)\*iM,1]; % k,P,nM,iM,

174 end;

175 detM = prod(p); invM = iM; % p,detM,invM,

176

177 **(2) Optimal scheme**

178

179 function [detM,invM,p,s,rc] = det\_inv\_p(M)

180 % The determinant of a given matrix and an array of

181 % pivots are found by matrix order condensation.

182 % Then as by-product the inverse matrix is obtained

183 % by matrix order condensation. ----- Optimal Scheme.

184 % F.C. Chang 09/11/2015.

185

186 k = 0; N = size(M,1); mA = max(abs(M(:))); % N,M,

187 nM = M; mM = [ ]; iM = [ ]; % k,nM,mM,iM,

188 s = 1; r = [ ]; c = [ ]; p = [ ];

189 for k = 1:N, n = N-k+1;

190 [mp,rcm] = max(abs(nM(:)));

191 [ri,ci] = ind2sub([n,n],rcm); rci = [ri;ci];

192 rx = setdiff([1:n],ri); cx = setdiff([1:n],ci);

193 ro = setdiff([1:N],r); co = setdiff([1:N],c);

194 r = [r,ro(ri)]; c = [c,co(ci)]; rc = [r;c];

195 P = nM(rcm); iP = 1/P; p = [p,P];

196 if mp/mA < 1.e-10, disp('Given matrix is singular !');

197 detM = 0; invM = NaN; p; s; rc; return, end;

198 s = s\*(-1)^(ri+ci); % k,rci,rc;P,iP,

199 nM = nM(rx,cx)-nM(rx,ci)\*iP\*nM(ri,cx);

200 mM = M(r,c);

201 iM = blkdiag(iM,0)+[-iM\*mM(1:k-1,k);1] ...

202 \*iP\*[-mM(k,1:k-1)\*iM,1]; % nM,mM,iM,

203 end;

204 detM = prod(p)\*s; invM(c,r) = iM; % rc,s,p,detM,invM,

205

206

207

208 **(3) Extended scheme**

209

210 function [detM,invM,p,s,rc] = det\_inv\_LL(M)

211 % Find the determinant via order condensation and then

```

212 %      as by-product the inverse via order expansion.
213 %      Manually select rows/columns for pivot blocks.
214 %      ----- Expanded Scheme.
215 %      F C Chang      09/11/15
216
217         k = 0; N = size(M,1); nM = M; mM = [ ]; iM = [ ];
218         r = [ ]; c = [ ]; p = [ ]; s = 1; t(1) = 0;      k,nM,mM,
219     for k = 1:N,
220         n = size(nM,1); m = size(mM,1); if n == 0, break,end;
221         disp('Select rows and columns from nM ');
222         rci = input(' [ri;ci] = ');
223         ri = sort(rci(1,:));      ci = sort(rci(2,:));
224         rx = setdiff([1:n],ri);   cx = setdiff([1:n],ci);
225         ro = setdiff([1:N],r);    co = setdiff([1:N],c);
226         r = [r,ro(ri)];           c = [c,co(ci)];
227         rci = [ri;ci];      rcx =[rx;cx];      rc = [r;c];
228         s = s*(-1)^sum(ri+ci);
229         P = nM(ri,ci);      L = size(P,1);
230         iP = inv(P);      d = det(P);      p = [p,d];
231         t(k+1) = t(k)+L; rt = [1:t(k)]; ct = [t(k)+1:t(k+1)];
232         nM = nM(rx,cx)-nM(rx,ci)*iP*nM(ri,cx);
233         mM = M(r,c);
234         iM = blkdiag(iM,zeros(L,L))+[-iM*mM(rt,ct);eye(L)] ...
235             *iP*[-mM(ct,rt)*iM,eye(L)];      k,P,d,iP,nM,mM,iM,
236     end;
237         detM = prod(p)*s; invM(c,r) = iM;      rc,s,p,detM,invM,
238
239
240
241

```

## 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 large. By removing any %'s at appropriate locations in these routine, the expected related intermediate data will appear in the processing output.

The valid outputs 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 and Appendix for more detail.

## 4. NUMERICAL ILLUSTRATIONS

```

262
263     >> N=5, M=magic(5),
264         N =

```

```

265         5
266     M =
267         17         24         1         8         15
268         23         5         7         14         16
269         4         6         13        20         22
270         10        12        19        21         3
271         11        18        25         2         9
272
273     >> [detM,invM,p]=det_inv_o(M) ,
274         detM =
275             5070000
276         invM =
277         -0.0049    0.0512   -0.0354    0.0012    0.0034
278         0.0431   -0.0373   -0.0046    0.0127    0.0015
279         -0.0303    0.0031    0.0031    0.0031    0.0364
280         0.0047   -0.0065    0.0108    0.0435   -0.0370
281         0.0028    0.0050    0.0415   -0.0450    0.0111
282     p =
283         17.0000   -27.4706    12.8373   -9.3786    90.1734
284
285     >> [detM,invM,p,s,rc]=det_inv_p(M)
286         detM =
287             5070000
288         invM =
289         -0.0049    0.0512   -0.0354    0.0012    0.0034
290         0.0431   -0.0373   -0.0046    0.0127    0.0015
291         -0.0303    0.0031    0.0031    0.0031    0.0364
292         0.0047   -0.0065    0.0108    0.0435   -0.0370
293         0.0028    0.0050    0.0415   -0.0450    0.0111
294     p =
295         25.0000    23.2800    20.1031   -22.1667    19.5489
296     s =
297         -1
298     rs =
299         5         1         3         4         2
300         3         2         4         5         1
301
302
303     >> [detM,invM,p,s,rs]=det_inv_LL(M)
304     N =
305         5
306     M =
307         17         24         1         8         15
308         23         5         7         14         16
309         4         6         13        20         22
310         10        12        19        21         3
311         11        18        25         2         9
312     k =
313         0
314     nM =
315         17         24         1         8         15
316         23         5         7         14         16
317         4         6         13        20         22
318         10        12        19        21         3
319         11        18        25         2         9
320     mM =
321         [ ]
322     k =
323         1
324     Select rows and columns from nM

```

```

325 [ri;ci] =
326 [1 2 3; 2 4 5]
327 P =
328      24      8      15
329      5      14      16
330      6      20      22
331 d =
332     -160
333 iP =
334      0.0750     -0.7750      0.5125
335      0.0875     -2.7375      1.9313
336     -0.1000      2.7000     -1.8500
337 nM =
338      1153.8     -107.66
339      -97.5      36.562
340 mM =
341      24      8      15
342      5      14      16
343      6      20      22
344 iM =
345      0.0750     -0.7750      0.5125
346      0.0875     -2.7375      1.9313
347     -0.1000      2.7000     -1.8500
348 k =
349      2
350 Select rows and columns from nM
351 [ri;ci] =
352 [1 2; 1 2]
353 P =
354      1153.8     -107.66
355      -97.5      36.562
356 d =
357      31687
358 iP =
359      0.0011538      0.0033974
360      0.0030769      0.03641
361 nM =
362 [ ]
363 mM =
364      24      8      15      17      1
365      5      14      16      23      7
366      6      20      22      4      13
367      12      21      3      10      19
368      18      2      9      11      25
369 iM =
370      0.0431     -0.0373     -0.0046      0.0127      0.0015
371      0.0047     -0.0065      0.0108      0.0435     -0.0370
372      0.0028      0.0050      0.0415     -0.0450      0.0111
373     -0.0049      0.0512     -0.0354      0.0012      0.0034
374     -0.0303      0.0031      0.0031      0.0031      0.0364
375 -----
376 detM =
377      5070000
378 invM =
379     -0.0049      0.0512     -0.0354      0.0012      0.0034
380      0.0431     -0.0373     -0.0046      0.0127      0.0015
381     -0.0303      0.0031      0.0031      0.0031      0.0364
382      0.0047     -0.0065      0.0108      0.0435     -0.0370
383      0.0028      0.0050      0.0415     -0.0450      0.0111
384 p =

```

```

385         -160.0      31687.5
386     s =
387         -1
388     rc =
389         1      2      3      4      5
390         2      4      5      1      3
391
392
393 >> N=8, M=magic(N), [detM,invM,p,s,rs]=det_inv_p(M),
394     N =
395         8
396     M =
397         64      2      3      61      60      6      7      57
398         9      55      54      12      13      51      50      16
399         17      47      46      20      21      43      42      24
400         40      26      27      37      36      30      31      33
401         32      34      35      29      28      38      39      25
402         41      23      22      44      45      19      18      48
403         49      15      14      52      53      11      10      56
404         8      58      59      5      4      62      63      1
405
406     Given matrix is singular !
407     detM =
408         0
409     invM =
410         NaN
411     p =
412         64      62.125  12.817  -7.4226e-015      ---
413     s =
414         1
415     rs =
416         1      8      2      5      ---
417         1      7      8      2      ---
418
419
420 >> N=11, M=magic(N); [detM,invM,p]=det_inv_o(M); detM,p,
421     N =
422         11
423     detM =
424         NaN
425     p =
426         68.000      -2.2941      0      -Inf      NaN      NaN
427         NaN      NaN      NaN      NaN      NaN      NaN
428                                     ----> No good !
429
430 >> N=11, M=magic(N); [detM,invM,p,s,rs]=det_inv_p(M); detM,p,
431     N =
432         11
433     detM =
434         -4.1038e+022
435     p =
436         121.0000  119.1074  109.3497  110.7177  111.5512  112.5070
437         114.6380  109.3049  107.1374  117.2489  119.0564
438                                     ----> Good !
439
440
441 >> N=3, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M),
442     N =
443         3
444     detM =

```

```

445         1.2424 + 2.6826i
446     invM =
447         0.031601 + 0.11641i    0.0758 - 0.42246i    0.099138 -0.13227i
448        -0.0066331 - 0.62526i   -0.25293 - 0.075866i   0.6305 - 0.20649i
449         0.029416 + 0.16779i   -0.040886 + 0.82601i   -0.11191 - 0.73749i
450     p =
451         0.4617 + 1.4524i    0.12257 + 1.5995i    -0.059778 - 1.2077i
452     s =
453         1
454     rs =
455         3          1          2
456         1          2          3
457
458
459 >> N=9; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
460 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
461     N =
462         9
463     elapsed_time =
464         0.04
465     erM =
466         2.8833e-015
467
468 >> N=99; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
469 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
470     N =
471         99
472     elapsed_time =
473         0.671
474     erM =
475         1.786e-012
476
477 >> N=555; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
478 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
479     N =
480         555
481     elapsed_time =
482         109.61
483     erM =
484         1.629e-011
485
486 >> N=999; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
487 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
488     N =
489         999
490     elapsed_time =
491         734.84
492     erM =
493         1.933e-010
494
495

```

## 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 all very compact, efficient, straightforward, and involves only the simple elementary arithmetical operations, such as addition, subtraction, multiplication, and division. It does not involve any high mathematics at all.

It is shown that for a given  $N \times N$  matrix, the number of multiplication/division operations needed to create a set of  $N$  pivot elements and their inverse are  $(N^3 - N)/3$ , that includes overall  $N$  division operations. It follows applying these results the number of multiplications required to compute the determinant and inversion of this given  $N \times N$  matrix are merely  $(N - 1)$  and  $(N^3 - N^2)$  respectively. Noted that  $N^3$  is the total number of multiplications needed to compute the product of any two  $N \times N$  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.

## REFERENCES

1. Sherman j. and Morrison W.J., "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 H.S., "Matrix inversion by the annihilation of rank," *J. Soc. Indust. Appl. Math.* 1959: 7(2); 149-151.
3. Asif A. and Moura J.M.F., "Block matrices with L-block-banded inverse: Inversion Algorithm," *IEEE Trans. on Signal Processing*, 2005: 53(2): 630-642.
4. Chang F.C., "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. and Wang X., "New recursive algorithm for matrix inversion," *J. Systems Engineering and Electronics*, 2008; 19(2): 381-384.
6. Aydin K. and Celik Kizilkan G., "Iterative inverse algorithm for perturbed matrix," *SDU Science Journal*, 2008; 3(1): 107-112.
7. Su C.T. and Chang F.C., "Quick evaluation of determinant," *Appl. Math. & Comput.* 1996; 75: 117-118.
8. Chang F.C., "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

```

556
557 >> format short
558
559 >> M=magic(5),
560
561 M =
562     17     24     1     8     15
563     23     5     7    14     16
564     4      6    13    20    22
565     10    12    19    21     3
566     11    18    25     2     9
567
568
569 >> [detM,invM,p]=det_inv_o(M);
570
571 k =
572     1
573 P =
574     17
575 nM =
576    -27.4706     5.6471     3.1765    -4.2941
577     0.3529    12.7647    18.1176    18.4706
578    -2.1176    18.4118    16.2941    -5.8235
579     2.4706    24.3529    -3.1765    -0.7059
580 iM =
581     0.0588
582 -----
583 k =
584     2
585 P =
586    -27.4706
587 nM =
588     12.8373     18.1585     18.4154
589     17.9764     16.0493    -5.4925
590     24.8608    -2.8908    -1.0921
591 iM =
592    -0.0107     0.0514
593     0.0493    -0.0364
594 -----
595 k =
596     3
597 P =
598     12.8373
599 nM =
600    -9.3786   -31.2802
601   -38.0567   -36.7556
602 iM =
603    -0.0038     0.0510    -0.0272
604     0.0452    -0.0362     0.0160
605    -0.0197     0.0010     0.0779
606 -----
607 k =
608     4
609 P =
610    -9.3786
611 nM =
612     90.1734
613 iM =
614    -0.0058     0.0496    -0.0481     0.0149
615     0.0428    -0.0380    -0.0101     0.0187

```

```

616         -0.0393    -0.0133    -0.1333     0.1508
617         0.0139     0.0101     0.1493    -0.1066
618         -----
619     k =
620         5
621     P =
622         90.1734
623     nM =
624         []
625     iM =
626         -0.0049     0.0512    -0.0354     0.0012     0.0034
627         0.0431    -0.0373    -0.0046     0.0127     0.0015
628         -0.0303     0.0031     0.0031     0.0031     0.0364
629         0.0047    -0.0065     0.0108     0.0435    -0.0370
630         0.0028     0.0050     0.0415    -0.0450     0.0111
631     -----
632     detM =
633         5.0700e+006
634     invM =
635         -0.0049     0.0512    -0.0354     0.0012     0.0034
636         0.0431    -0.0373    -0.0046     0.0127     0.0015
637         -0.0303     0.0031     0.0031     0.0031     0.0364
638         0.0047    -0.0065     0.0108     0.0435    -0.0370
639         0.0028     0.0050     0.0415    -0.0450     0.0111
640     P =
641         17.0000   -27.4706    12.8373    -9.3786    90.1734
642
643
644     >> [detM,invM,p,s,rc]=det_inv_p(M) ,
645
646     N =
647         5
648     M =
649         17     24     1     8     15
650         23     5     7    14    16
651         4      6    13    20    22
652         10    12    19    21     3
653         11    18    25     2     9
654     -----
655     k =
656         0
657     nM =
658         17     24     1     8     15
659         23     5     7    14    16
660         4      6    13    20    22
661         10    12    19    21     3
662         11    18    25     2     9
663     mM =
664         [ ]
665     iM =
666         [ ]
667     -----
668     k =
669         1
670     rci =
671         5     3
672     P =
673         25
674     iP =
675         0.0400

```

```

676      nM =
677      16.5600    23.2800    7.9200    14.6400
678      19.9200   -0.0400    13.4400    13.4800
679      -1.7200   -3.3600    18.9600    17.3200
680      1.6400   -1.6800    19.4800   -3.8400
681      mM =
682      25
683      iM =
684      0.0400
685      -----
686      k =
687      2
688      rci =
689      1      2
690      P =
691      23.2800
692      iP =
693      0.0430
694      nM =
695      19.9485    13.4536    13.5052
696      0.6701    20.1031    19.4330
697      2.8351    20.0515   -2.7835
698      mM =
699      25      18
700      1      24
701      iM =
702      0.0412   -0.0309
703      -0.0017    0.0430
704      -----
705      k =
706      3
707      rci =
708      2      2
709      P =
710      20.1031
711      iP =
712      0.0497
713      nM =
714      19.5000    0.5000
715      2.1667   -22.1667
716      mM =
717      25      18      2
718      1      24      8
719      13      6      20
720      iM =
721      0.0369   -0.0297    0.0082
722      0.0072    0.0405   -0.0169
723      -0.0262    0.0072    0.0497
724      -----
725      k =
726      4
727      rci =
728      2      2
729      P =
730      -22.1667
731      iP =
732      -0.0451
733      nM =
734      19.5489
735      mM =

```

```

736         25      18      2      9
737         1      24      8     15
738         13      6     20     22
739         19     12     21      3
740     iM =
741         0.0362    -0.0300     0.0052     0.0030
742         0.0040     0.0395    -0.0304     0.0135
743        -0.0366     0.0040     0.0062     0.0436
744         0.0108     0.0032     0.0450    -0.0451
745     -----
746     k =
747         5
748     rci =
749         1      1
750     P =
751        19.5489
752     iP =
753         0.0512
754     nM =
755         [ ]
756     mM =
757         25      18      2      9      11
758         1      24      8     15     17
759         13      6     20     22      4
760         19     12     21      3     10
761         7      5     14     16     23
762     iM =
763         0.0364    -0.0303     0.0031     0.0031     0.0031
764         0.0015     0.0431    -0.0046     0.0127    -0.0373
765        -0.0370     0.0047     0.0108     0.0435    -0.0065
766         0.0111     0.0028     0.0415    -0.0450     0.0050
767         0.0034    -0.0049    -0.0354     0.0012     0.0512
768     -----
769     detM =
770        5.0700e+006
771     invM =
772        -0.0049     0.0512    -0.0354     0.0012     0.0034
773         0.0431    -0.0373    -0.0046     0.0127     0.0015
774        -0.0303     0.0031     0.0031     0.0031     0.0364
775         0.0047    -0.0065     0.0108     0.0435    -0.0370
776         0.0028     0.0050     0.0415    -0.0450     0.0111
777     p =
778        25.0000     23.2800     20.1031    -22.1667     19.5489
779     s =
780        -1
781     rc =
782         5      1      3      4      2
783         3      2      4      5      1
784
785
786
787 >> N=3, M=rand(N)+i*randn(N), [dM,iM]=det_inv_p(M),
788 >>     W=iM, [dW,iW]=det_inv_p(W), erD=dM-1/dW, erM=M-iW,
789     N =
790         3
791     M =
792         0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
793         0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
794         0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
795     dM =

```

```

796         0.8840 + 1.9683i
797     iM =
798         0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
799         0.5856 + 0.5723i    -0.0536 - 0.1048i    -0.2642 - 0.1396i
800         -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
801         -----
802     W =
803         0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
804         0.5856 + 0.5723i    -0.0536 - 0.1048i    -0.2642 - 0.1396i
805         -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
806     dW =
807         0.1899 - 0.4228i
808     iW =
809         0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
810         0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
811         0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
812         -----
813     erD =
814         3.3307e-016 -2.2204e-016i
815     erM =
816         1.0e-015 *
817         -0.0555 + 0.0000i    0.0000 - 0.1110i    -0.0555 - 0.1388i
818         -0.1110 - 0.4441i    0.1110 + 0.0000i    -0.2220 + 0.0000i
819         0.0000 - 0.1110i    0.0555 - 0.0833i    -0.1110 + 0.0139i
820
821
822     >> diary off
823
824

```