

Original Research Article

Matrix Inverse as by-product of Determinant

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.

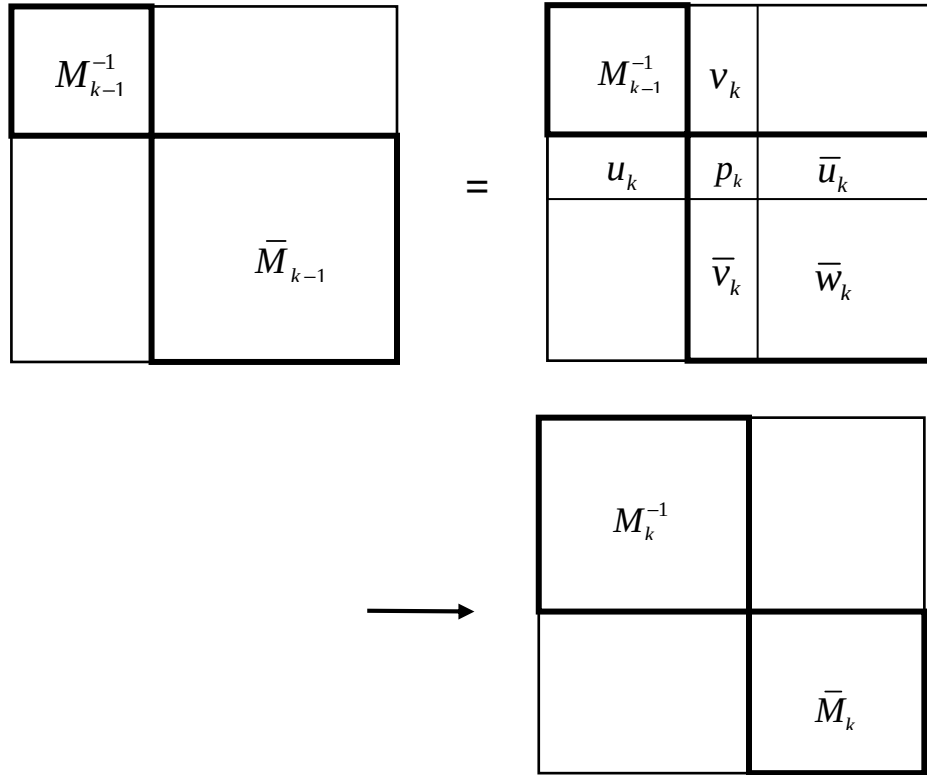
1. INTRODUCTION

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.

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, $k = 1, 2, \dots, N$, 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:

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,

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

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

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

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

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.

111
112
113
114

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

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

115

we have

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

116
117

(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

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

118

and

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

119
120

(end of proof)

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

124

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

125

126 and

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

127
128
129

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 Figure 2. In keeping with what was previously suggested, this extended iteration process may fail should any pivot block get out to be singular.

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

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

or

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

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

Fig. 2. Picture illustration of extended iteration

161

162 **3. COMPUTER ROUTINES**

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

165

166 **(1) Basic scheme**

167

```

168 function [detM,invM,p] = det_inv_o(M)
169 % The determinant the inverse of a given matrix are
170 % found by matrix order condensation and expansion.
171 % ---- Basic Scheme.
172 % F.C. Chang 09/11/2015.
173
174 N = size(M,1); nM = M; iM = [ ]; p = [ ];
175 for k = 1:N, n = N-k+1;
176 P = nM(1,1); iP = 1/P; p = [p,P];
177 nM = nM(2:n,2:n)-nM(2:n,1)*iP*nM(1,2:n);
178 iM = blkdiag(iM,0)+[-iM*M(1:k-1,k);1] ...
179 *iP*[-M(k,1:k-1)*iM,1]; % k,P,nM,iM,
180 end;
181 detM = prod(p); invM = iM; % p,detM,invM,

```

182

183 **(2) Optimal scheme**

184

```

185 function [detM,invM,p,s,rc] = det_inv_p(M)
186 % The determinant of a given matrix and an array of
187 % pivots are found by matrix order condensation.
188 % Then as by-product the inverse matrix is obtained
189 % by matrix order condensation. ---- Optimal Scheme.
190 % F.C. Chang 09/11/2015.
191
192 k = 0; N = size(M,1); mA = max(abs(M(:))); % N,M,
193 nM = M; mM = [ ]; iM = [ ]; % k,nM,mM,iM,
194 s = 1; r = [ ]; c = [ ]; p = [ ];
195 for k = 1:N, n = N-k+1;
196 [mp,rcm] = max(abs(nM(:)));
197 [ri,ci] = ind2sub([n,n],rcm); rci = [ri;ci];
198 rx = setdiff([1:n],ri); cx = setdiff([1:n],ci);
199 ro = setdiff([1:N],r); co = setdiff([1:N],c);
200 r = [r,ro(ri)]; c = [c,co(ci)]; rc = [r;c];
201 P = nM(rcm); iP = 1/P; p = [p,P];
202 if mp/mA < 1.e-10, disp('Given matrix is singular !');
203 detM = 0; invM = NaN; p; s; rc; return, end;
204 s = s*(-1)^(ri+ci); % k,rci,rc;P,iP,
205 nM = nM(rx,cx)-nM(rx,ci)*iP*nM(ri,cx);
206 mM = M(r,c);
207 iM = blkdiag(iM,0)+[-iM*mM(1:k-1,k);1] ...
208 *iP*[-mM(k,1:k-1)*iM,1]; % nM,mM,iM,
209 end;
210 detM = prod(p)*s; invM(c,r) = iM; % rc,s,p,detM,invM,

```

211

212

```

213
214 (3) Extended scheme
215
216 function [detM,invM,p,s,rc] = det_inv_LL(M)
217 % Find the determinant via order condensation and then
218 % as by-product the inverse via order expansion.
219 % Manually select rows/columns for pivot blocks.
220 % ----- Expanded Scheme.
221 % F C Chang 09/11/15
222
223 k = 0; N = size(M,1); nM = M; mM = [ ]; iM = [ ];
224 r = [ ]; c = [ ]; p = [ ]; s = 1; t(1) = 0; k,nM,mM,
225 for k = 1:N,
226 n = size(nM,1); m = size(mM,1); if n == 0, break,end;
227 disp('Select rows and columns from nM ');
228 rci = input('ri;ci = ');
229 ri = sort(rci(1,:)); ci = sort(rci(2,:));
230 rx = setdiff([1:n],ri); cx = setdiff([1:n],ci);
231 ro = setdiff([1:N],r); co = setdiff([1:N],c);
232 r = [r,ro(ri)]; c = [c,co(ci)];
233 rci = [ri;ci]; rcx = [rx;cx]; rc = [r;c];
234 s = s*(-1)^sum(ri+ci);
235 P = nM(ri,ci); L = size(P,1);
236 iP = inv(P); d = det(P); p = [p,d];
237 t(k+1) = t(k)+L; rt = [1:t(k)]; ct = [t(k)+1:t(k+1)];
238 nM = nM(rx,cx)-nM(rx,ci)*iP*nM(ri,cx);
239 mM = M(r,c);
240 iM = blkdiag(iM,zeros(L,L))+[-iM*mM(rt,ct);eye(L)] ...
241 *iP*[-mM(ct,rt)*iM,eye(L)]; k,P,d,iP,nM,mM,iM,
242 end;
243 detM = prod(p)*s; invM(c,r) = iM; rc,s,p,detM,invM,
244
245

```

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.

266

267 **4. NUMERICAL ILLUSTRATIONS**

268

269 **>> N=5, M=magic(5),**270 **N =**271 **5**272 **M =**273 **17 24 1 8 15**274 **23 5 7 14 16**275 **4 6 13 20 22**276 **10 12 19 21 3**277 **11 18 25 2 9**

278

279 **>> [detM,invM,p]=det_inv_o(M),**280 **detM =**281 **5070000**282 **invM =**283 **-0.0049 0.0512 -0.0354 0.0012 0.0034**284 **0.0431 -0.0373 -0.0046 0.0127 0.0015**285 **-0.0303 0.0031 0.0031 0.0031 0.0364**286 **0.0047 -0.0065 0.0108 0.0435 -0.0370**287 **0.0028 0.0050 0.0415 -0.0450 0.0111**288 **p =**289 **17.0000 -27.4706 12.8373 -9.3786 90.1734**

290

291 **>> [detM,invM,p,s,rc]=det_inv_p(M)**292 **detM =**293 **5070000**294 **invM =**295 **-0.0049 0.0512 -0.0354 0.0012 0.0034**296 **0.0431 -0.0373 -0.0046 0.0127 0.0015**297 **-0.0303 0.0031 0.0031 0.0031 0.0364**298 **0.0047 -0.0065 0.0108 0.0435 -0.0370**299 **0.0028 0.0050 0.0415 -0.0450 0.0111**300 **p =**301 **25.0000 23.2800 20.1031 -22.1667 19.5489**302 **s =**303 **-1**304 **rs =**305 **5 1 3 4 2**306 **3 2 4 5 1**

307

308

309 **>> [detM,invM,p,s,rs]=det_inv_LL(M)**310 **N =**311 **5**312 **M =**313 **17 24 1 8 15**314 **23 5 7 14 16**315 **4 6 13 20 22**316 **10 12 19 21 3**317 **11 18 25 2 9**318 **k =**319 **0**320 **nM =**321 **17 24 1 8 15**322 **23 5 7 14 16**323 **4 6 13 20 22**324 **10 12 19 21 3**

```

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

```

```

385         -0.0049      0.0512      -0.0354      0.0012      0.0034
386         0.0431      -0.0373      -0.0046      0.0127      0.0015
387        -0.0303      0.0031      0.0031      0.0031      0.0364
388         0.0047      -0.0065      0.0108      0.0435      -0.0370
389         0.0028      0.0050      0.0415      -0.0450      0.0111
390     p =
391         -160.0      31687.5
392     s =
393         -1
394     rc =
395         1      2      3      4      5
396         2      4      5      1      3

```

```

399     >> N=8, M=magic(N), [detM,invM,p,s,rs]=det_inv_p(M),
400     N =
401         8
402     M =
403         64      2      3      61      60      6      7      57
404         9      55      54      12      13      51      50      16
405        17      47      46      20      21      43      42      24
406        40      26      27      37      36      30      31      33
407        32      34      35      29      28      38      39      25
408        41      23      22      44      45      19      18      48
409        49      15      14      52      53      11      10      56
410         8      58      59      5      4      62      63      1

```

```

411
412     Given matrix is singular !
413     detM =
414         0
415     invM =
416         NaN
417     p =
418         64      62.125      12.817      -7.4226e-015      ---
419     s =
420         1
421     rs =
422         1      8      2      5      ---
423         1      7      8      2      ---

```

```

424
425
426     >> N=11, M=magic(N); [detM,invM,p]=det_inv_o(M); detM,p,
427     N =
428         11
429     detM =
430         NaN
431     p =
432         68.000      -2.2941      0      -Inf      NaN      NaN
433         NaN      NaN      NaN      NaN      NaN
434                                     ---→ No good !
435

```

```

436     >> N=11, M=magic(N); [detM,invM,p,s,rs]=det_inv_p(M); detM,p,
437     N =
438         11
439     detM =
440         -4.1038e+022
441     p =
442         121.0000      119.1074      109.3497      110.7177      111.5512      112.5070
443         114.6380      109.3049      107.1374      117.2489      119.0564
444                                     ---→ Good !

```

```

445
446
447 >> N=3, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M),
448     N =
449         3
450     detM =
451         1.2424 + 2.6826i
452     invM =
453         0.031601 + 0.11641i    0.0758 - 0.42246i    0.099138 - 0.13227i
454        -0.0066331 - 0.62526i   -0.25293 - 0.075866i    0.6305 - 0.20649i
455         0.029416 + 0.16779i   -0.040886 + 0.82601i   -0.11191 - 0.73749i
456     p =
457         0.4617 + 1.4524i    0.12257 + 1.5995i    -0.059778 - 1.2077i
458     s =
459         1
460     rs =
461         3         1         2
462         1         2         3
463
464
465 >> N=9; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
466 >>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
467     N =
468         9
469     elapsed_time =
470         0.04
471     erM =
472         2.8833e-015
473
474 >> N=99; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
475 >>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
476     N =
477        99
478     elapsed_time =
479         0.671
480     erM =
481         1.786e-012
482
483 >> N=555; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
484 >>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
485     N =
486       555
487     elapsed_time =
488       109.61
489     erM =
490       1.629e-011
491
492 >> N=999; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
493 >>     toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
494     N =
495      999
496     elapsed_time =
497       734.84
498     erM =
499       1.933e-010
500
501

```

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 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 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 $N \times N$ 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 $N \times N$ 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.

554 8. Chang F.C., "Determinant of matrix by order condensation," *British J. of Math. &*
555 *Comput. Science*, 2014; 4(13): 1843-1848.

556

557 APPENDIX

558

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

561

562 >> diary on

563

564 >> format short

565

566 >> M=magic(5),

567

568 M =

569	17	24	1	8	15
570	23	5	7	14	16
571	4	6	13	20	22
572	10	12	19	21	3
573	11	18	25	2	9

574

575

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

577

578 k =

579 1

580 P =

581 17

582 nM =

583	-27.4706	5.6471	3.1765	-4.2941
584	0.3529	12.7647	18.1176	18.4706
585	-2.1176	18.4118	16.2941	-5.8235
586	2.4706	24.3529	-3.1765	-0.7059

587 iM =

588 0.0588

589 -----

590 k =

591 2

592 P =

593 -27.4706

594 nM =

595	12.8373	18.1585	18.4154
596	17.9764	16.0493	-5.4925
597	24.8608	-2.8908	-1.0921

598 iM =

599	-0.0107	0.0514
600	0.0493	-0.0364

601 -----

602 k =

603 3

604 P =

605 12.8373

606 nM =

607	-9.3786	-31.2802
608	-38.0567	-36.7556

609 iM =

610	-0.0038	0.0510	-0.0272
611	0.0452	-0.0362	0.0160

```

612         -0.0197    0.0010    0.0779
613         -----
614         k =
615             4
616         P =
617         -9.3786
618         nM =
619         90.1734
620         iM =
621         -0.0058    0.0496   -0.0481    0.0149
622         0.0428   -0.0380   -0.0101    0.0187
623         -0.0393   -0.0133   -0.1333    0.1508
624         0.0139    0.0101    0.1493   -0.1066
625         -----
626         k =
627             5
628         P =
629         90.1734
630         nM =
631         []
632         iM =
633         -0.0049    0.0512   -0.0354    0.0012    0.0034
634         0.0431   -0.0373   -0.0046    0.0127    0.0015
635         -0.0303    0.0031    0.0031    0.0031    0.0364
636         0.0047   -0.0065    0.0108    0.0435   -0.0370
637         0.0028    0.0050    0.0415   -0.0450    0.0111
638         -----
639         detM =
640         5.0700e+006
641         invM =
642         -0.0049    0.0512   -0.0354    0.0012    0.0034
643         0.0431   -0.0373   -0.0046    0.0127    0.0015
644         -0.0303    0.0031    0.0031    0.0031    0.0364
645         0.0047   -0.0065    0.0108    0.0435   -0.0370
646         0.0028    0.0050    0.0415   -0.0450    0.0111
647         p =
648         17.0000   -27.4706    12.8373   -9.3786    90.1734
649
650
651     >> [detM,invM,p,s,rc]=det_inv_p(M),
652
653         N =
654             5
655         M =
656         17    24     1     8    15
657         23     5     7    14    16
658         4     6    13    20    22
659         10    12    19    21     3
660         11    18    25     2     9
661         -----
662         k =
663             0
664         nM =
665         17    24     1     8    15
666         23     5     7    14    16
667         4     6    13    20    22
668         10    12    19    21     3
669         11    18    25     2     9
670         mM =
671         [ ]

```

```

672 iM =
673     [ ]
674 -----
675 k =
676     1
677 rci =
678     5     3
679 P =
680     25
681 iP =
682     0.0400
683 nM =
684     16.5600    23.2800     7.9200    14.6400
685     19.9200    -0.0400    13.4400    13.4800
686     -1.7200    -3.3600    18.9600    17.3200
687     1.6400    -1.6800    19.4800    -3.8400
688 mM =
689     25
690 iM =
691     0.0400
692 -----
693 k =
694     2
695 rci =
696     1     2
697 P =
698     23.2800
699 iP =
700     0.0430
701 nM =
702     19.9485    13.4536    13.5052
703     0.6701    20.1031    19.4330
704     2.8351    20.0515    -2.7835
705 mM =
706     25     18
707     1     24
708 iM =
709     0.0412    -0.0309
710     -0.0017     0.0430
711 -----
712 k =
713     3
714 rci =
715     2     2
716 P =
717     20.1031
718 iP =
719     0.0497
720 nM =
721     19.5000     0.5000
722     2.1667    -22.1667
723 mM =
724     25     18     2
725     1     24     8
726     13     6     20
727 iM =
728     0.0369    -0.0297     0.0082
729     0.0072     0.0405    -0.0169
730     -0.0262     0.0072     0.0497
731 -----

```

```

732 k =
733     4
734 rci =
735     2     2
736 P =
737    -22.1667
738 iP =
739    -0.0451
740 nM =
741    19.5489
742 mM =
743     25     18     2     9
744     1     24     8    15
745     13     6    20    22
746     19    12    21     3
747 iM =
748     0.0362    -0.0300     0.0052     0.0030
749     0.0040     0.0395    -0.0304     0.0135
750    -0.0366     0.0040     0.0062     0.0436
751     0.0108     0.0032     0.0450    -0.0451
752 -----
753 k =
754     5
755 rci =
756     1     1
757 P =
758    19.5489
759 iP =
760     0.0512
761 nM =
762     [ ]
763 mM =
764     25     18     2     9     11
765     1     24     8    15    17
766     13     6    20    22     4
767     19    12    21     3    10
768     7     5    14    16    23
769 iM =
770     0.0364    -0.0303     0.0031     0.0031     0.0031
771     0.0015     0.0431    -0.0046     0.0127    -0.0373
772    -0.0370     0.0047     0.0108     0.0435    -0.0065
773     0.0111     0.0028     0.0415    -0.0450     0.0050
774     0.0034    -0.0049    -0.0354     0.0012     0.0512
775 -----
776 detM =
777    5.0700e+006
778 invM =
779    -0.0049     0.0512    -0.0354     0.0012     0.0034
780     0.0431    -0.0373    -0.0046     0.0127     0.0015
781    -0.0303     0.0031     0.0031     0.0031     0.0364
782     0.0047    -0.0065     0.0108     0.0435    -0.0370
783     0.0028     0.0050     0.0415    -0.0450     0.0111
784 p =
785    25.0000    23.2800    20.1031    -22.1667    19.5489
786 s =
787    -1
788 rc =
789     5     1     3     4     2
790     3     2     4     5     1
791

```

```

792
793
794 >> N=3, M=rand(N)+i*randn(N), [dM,iM]=det_inv_p(M),
795 >> W=iM, [dW,iW]=det_inv_p(W), erD=dM-1/dW, erM=M-iW,
796
797     N =
798         3
799     M =
800         0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
801         0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
802         0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
803     dM =
804         0.8840 + 1.9683i
805     iM =
806         0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
807         0.5856 + 0.5723i   -0.0536 - 0.1048i   -0.2642 - 0.1396i
808         -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
809
810     W =
811         0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
812         0.5856 + 0.5723i   -0.0536 - 0.1048i   -0.2642 - 0.1396i
813         -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
814     dW =
815         0.1899 - 0.4228i
816     iW =
817         0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
818         0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
819         0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
820
821     erD =
822         3.3307e-016 -2.2204e-016i
823     erM =
824         1.0e-015 *
825         -0.0555 + 0.0000i    0.0000 - 0.1110i   -0.0555 - 0.1388i
826         -0.1110 - 0.4441i    0.1110 + 0.0000i   -0.2220 + 0.0000i
827         0.0000 - 0.1110i    0.0555 - 0.0833i   -0.1110 + 0.0139i
828
829 >> diary off
830
831

```