

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.

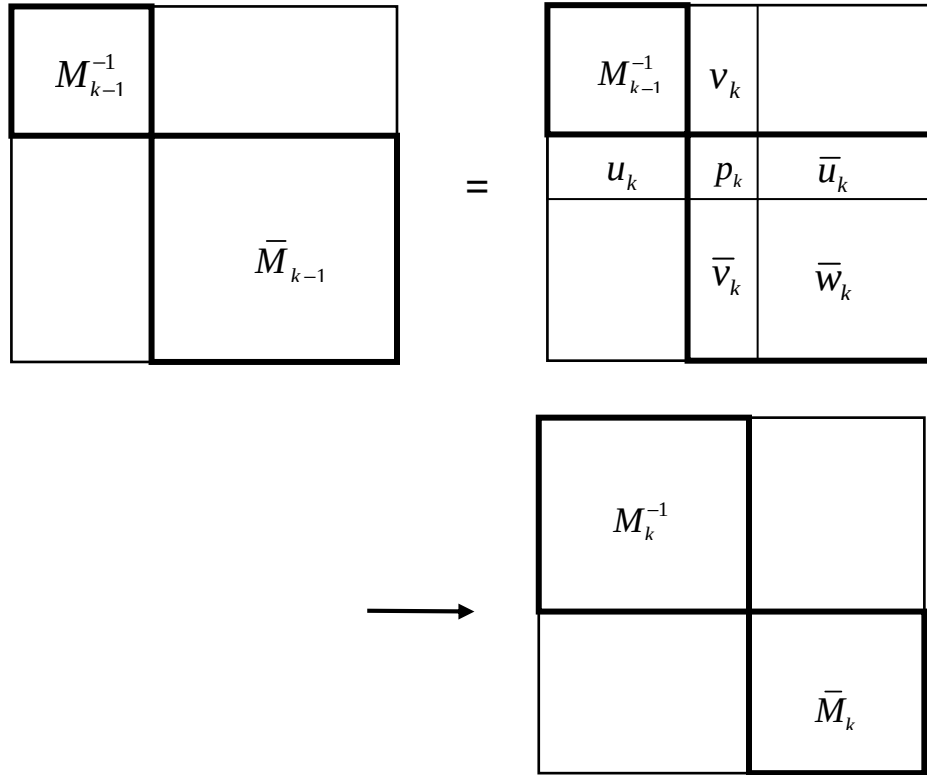
1. INTRODUCTION

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.

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

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

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

280

281 **>> [detM,invM,p]=det_inv_o(M),**282 **detM =**283 **5070000**284 **invM =**285 **-0.0049 0.0512 -0.0354 0.0012 0.0034**286 **0.0431 -0.0373 -0.0046 0.0127 0.0015**287 **-0.0303 0.0031 0.0031 0.0031 0.0364**288 **0.0047 -0.0065 0.0108 0.0435 -0.0370**289 **0.0028 0.0050 0.0415 -0.0450 0.0111**290 **p =**291 **17.0000 -27.4706 12.8373 -9.3786 90.1734**

292

293

294 **>> [detM,invM,p,s,rc]=det_inv_p(M)**295 **detM =**296 **5070000**297 **invM =**298 **-0.0049 0.0512 -0.0354 0.0012 0.0034**299 **0.0431 -0.0373 -0.0046 0.0127 0.0015**300 **-0.0303 0.0031 0.0031 0.0031 0.0364**301 **0.0047 -0.0065 0.0108 0.0435 -0.0370**302 **0.0028 0.0050 0.0415 -0.0450 0.0111**303 **p =**304 **25.0000 23.2800 20.1031 -22.1667 19.5489**305 **s =**306 **-1**307 **rs =**308 **5 1 3 4 2**309 **3 2 4 5 1**

310

311

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

```

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

```

```

385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444

```

detM =					
invM =	5070000				
	-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

```

```

445 d =
446     -1300
447 iP =
448     -0.032692     0.017308
449     -0.017308     0.032692
450 nM =
451     325
452 mM =
453     17     15     24     8
454     11     9     18     2
455     23     16     5     14
456     10     3     12     21
457 iM =
458     -0.35288     0.42212     0.086538     0.036538
459     0.41122     -0.48045     -0.036538     -0.086538
460     -0.0022436     0.05609     -0.032692     0.017308
461     0.11058     -0.16442     -0.017308     0.032692
462 Select rows and columns from nM
463 [ri;ci] = [1; 1]
464 k =
465     3
466 P =
467     325
468 d =
469     325
470 iP =
471     0.0030769
472 nM =
473     []
474 mM =
475     17     15     24     8     1
476     11     9     18     2     25
477     23     16     5     14     7
478     10     3     12     21     19
479     4     22     6     20     13
480 iM =
481     -0.0049359     0.0033974     0.051154     0.0011538     -0.035385
482     0.0027564     0.01109     0.005     -0.045     0.041538
483     0.043141     0.0014744     -0.037308     0.012692     -0.0046154
484     0.0046795     -0.036987     -0.0065385     0.043462     0.010769
485     -0.030256     0.03641     0.0030769     0.0030769     0.0030769
486 -----
487 detM =
488     5070000
489 invM =
490     -0.0049359     0.051154     -0.035385     0.0011538     0.0033974
491     0.043141     -0.037308     -0.0046154     0.012692     0.0014744
492     -0.030256     0.0030769     0.0030769     0.0030769     0.03641
493     0.0046795     -0.0065385     0.010769     0.043462     -0.036987
494     0.0027564     0.005     0.041538     -0.045     0.01109
495 p =
496     -12     -1300     325
497 s =
498     1
499 rc =
500     1     5     2     4     3
501     1     5     2     4     3
502
503
504

```

```

505 >> N=5, M=magic(N)/10e+9; [detM,invM,p,s,rc]=det_inv_p(M);
506     detM,p, erM=norm(M*invM-eye(N)),
507     N =
508         5
509     detM =
510         5.07e-044
511     p =
512         2.5e-009    2.328e-009    2.0103e-009   -2.2167e-009    1.9549e-009
513     erM =
514         4.5777e-016
515
516
517 >> N=8, M=magic(N), [detM,invM,p,s,rs]=det_inv_p(M),
518     N =
519         8
520     M =
521         64     2     3     61     60     6     7     57
522         9     55     54     12     13     51     50     16
523        17     47     46     20     21     43     42     24
524        40     26     27     37     36     30     31     33
525        32     34     35     29     28     38     39     25
526        41     23     22     44     45     19     18     48
527        49     15     14     52     53     11     10     56
528         8     58     59     5     4     62     63     1
529
530     Given matrix is singular !
531     detM =
532         0
533     invM =
534         NaN
535     p =
536         64    62.125    12.817   -7.4226e-015   ---
537     s =
538         1
539     rs =
540         1     8     2     5   ---
541         1     7     8     2   ---
542
543
544
545 >> N=11, M=magic(N); [detM,invM,p]=det_inv_o(M); detM,p,
546     N =
547         11
548     detM =
549         NaN
550     p =
551         68.000    -2.2941     0    -Inf     NaN     NaN
552         NaN     NaN     NaN     NaN     NaN     NaN
553     ---→ No good !
554
555
556 >> N=11, M=magic(N); [detM,invM,p,s,rs]=det_inv_p(M); detM,p,
557     N =
558         11
559     detM =
560        -4.1038e+022
561     p =
562        121.0000    119.1074    109.3497    110.7177    111.5512    112.5070
563        114.6380    109.3049    107.1374    117.2489    119.0564
564     ---→ Good !

```

```

565
566
567 >> N=3, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M),
568     N =
569         3
570     detM =
571         1.2424 + 2.6826i
572     invM =
573         0.031601 + 0.11641i    0.0758 - 0.42246i    0.099138 -0.13227i
574        -0.0066331 - 0.62526i   -0.25293 - 0.075866i    0.6305 - 0.20649i
575         0.029416 + 0.16779i   -0.040886 + 0.82601i   -0.11191 - 0.73749i
576     p =
577         0.4617 + 1.4524i    0.12257 + 1.5995i    -0.059778 - 1.2077i
578     s =
579         1
580     rs =
581         3         1         2
582         1         2         3
583
584
585 >> N=9; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
586 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
587     N =
588         9
589     elapsed_time =
590         0.04
591     erM =
592         2.8833e-015
593
594 >> N=99; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
595 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
596     N =
597        99
598     elapsed_time =
599         0.671
600     erM =
601         1.786e-012
602
603 >> N=555; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
604 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
605     N =
606       555
607     elapsed_time =
608       109.61
609     erM =
610       1.629e-011
611
612 >> N=999; tic, M=rand(N)+i*randn(N); [detM,invM,p,s,rs]=det_inv_p(M);
613 >> toc, N,detM;invM;p;s;rs; erM=norm(invM*M-eye(N)),
614     N =
615      999
616     elapsed_time =
617       734.84
618     erM =
619       1.933e-010
620
621

```

622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669

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 $N \times N$ 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.

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.

670

671 REFERENCES

- 672 1. Sherman j. and Morrison W.J., "Adjustment of inverse matrix corresponding to changes
673 in a given column or a given row of the original matrix," *Ann. Math. Stat.* 1949: 75; 124.
674
- 675 2. Wilf H.S., "Matrix inversion by the annihilation of rank," *J. Soc. Indust. Appl. Math.* 1959:
676 7(2); 149-151.
677
- 678 3. Asif A. and Moura J.M.F., "Block matrices with L-block-banded inverse: Inversion
679 Algorithm," *IEEE Trans. on Signal Processing*, 2005: 53(2): 630-642.
680
- 681 4. Chang F.C., "Inverse and determinant of a square matrix by order expansion and
682 condensation," *IEEE Antenna and Propagation Magazine*, 2015; 57(1): 28-32.
683
- 684 5. Jianshu C. and Wang X., "New recursive algorithm for matrix inversion," *J. Systems
685 Engineering and Electronics*, 2008; 19(2): 381-384.
686
- 687 6. Aydin K. and Celik Kizilkan G., "Iterative inverse algorithm for perturbed matrix," *SDU
688 Science Journal*, 2008; 3(1): 107-112.
689
- 690 7. Su C.T. and Chang F.C., "Quick evaluation of determinant," *Appl. Math. & Comput.*
691 1996; 75: 117-118.
692
- 693 8. Chang F.C., "Determinant of matrix by order condensation," *British J. of Math. &
694 Comput. Science*, 2014; 4(13): 1843-1848.
695
696

697

698

699

700 APPENDIX

701

702 **Notes:** Present the detail printout of intermediate steps in running the basic and optimal
703 iteration process after removing '%s' in the appropriate locations of the MATLAB routines.
704

705 **>> diary on**

706

707 **>> format short**

708

709 **>> M=magic(5),**

710

711 M =

712 17 24 1 8 15

713 23 5 7 14 16

714 4 6 13 20 22

715 10 12 19 21 3

716 11 18 25 2 9

717

```

718
719 >> [detM,invM,p]=det_inv_o(M);
720
721 k =
722     1
723 P =
724     17
725 nM =
726    -27.4706     5.6471     3.1765    -4.2941
727     0.3529    12.7647    18.1176    18.4706
728    -2.1176    18.4118    16.2941    -5.8235
729     2.4706    24.3529    -3.1765    -0.7059
730 iM =
731     0.0588
732 -----
733 k =
734     2
735 P =
736    -27.4706
737 nM =
738     12.8373     18.1585     18.4154
739     17.9764     16.0493    -5.4925
740     24.8608    -2.8908    -1.0921
741 iM =
742    -0.0107     0.0514
743     0.0493    -0.0364
744 -----
745 k =
746     3
747 P =
748     12.8373
749 nM =
750    -9.3786   -31.2802
751   -38.0567   -36.7556
752 iM =
753    -0.0038     0.0510    -0.0272
754     0.0452    -0.0362     0.0160
755    -0.0197     0.0010     0.0779
756 -----
757 k =
758     4
759 P =
760    -9.3786
761 nM =
762     90.1734
763 iM =
764    -0.0058     0.0496    -0.0481     0.0149
765     0.0428    -0.0380    -0.0101     0.0187
766    -0.0393    -0.0133    -0.1333     0.1508
767     0.0139     0.0101     0.1493    -0.1066
768 -----
769 k =
770     5
771 P =
772     90.1734
773 nM =
774     []
775 iM =
776    -0.0049     0.0512    -0.0354     0.0012     0.0034
777     0.0431    -0.0373    -0.0046     0.0127     0.0015

```

```

778         -0.0303    0.0031    0.0031    0.0031    0.0364
779         0.0047   -0.0065    0.0108    0.0435   -0.0370
780         0.0028    0.0050    0.0415   -0.0450    0.0111
781     -----
782     detM =
783         5.0700e+006
784     invM =
785         -0.0049    0.0512   -0.0354    0.0012    0.0034
786         0.0431   -0.0373   -0.0046    0.0127    0.0015
787         -0.0303    0.0031    0.0031    0.0031    0.0364
788         0.0047   -0.0065    0.0108    0.0435   -0.0370
789         0.0028    0.0050    0.0415   -0.0450    0.0111
790     p =
791         17.0000   -27.4706    12.8373   -9.3786    90.1734
792
793
794     >> [detM,invM,p,s,rc]=det_inv_p(M),
795
796     N =
797         5
798     M =
799         17    24     1     8    15
800         23     5     7    14    16
801         4     6    13    20    22
802         10    12    19    21     3
803         11    18    25     2     9
804     -----
805     k =
806         0
807     nM =
808         17    24     1     8    15
809         23     5     7    14    16
810         4     6    13    20    22
811         10    12    19    21     3
812         11    18    25     2     9
813     mM =
814         [ ]
815     iM =
816         [ ]
817     -----
818     k =
819         1
820     rci =
821         5     3
822     P =
823         25
824     iP =
825         0.0400
826     nM =
827         16.5600    23.2800     7.9200    14.6400
828         19.9200   -0.0400    13.4400    13.4800
829         -1.7200   -3.3600    18.9600    17.3200
830         1.6400   -1.6800    19.4800    -3.8400
831     mM =
832         25
833     iM =
834         0.0400
835     -----
836     k =
837         2

```

```

838 rci =
839     1     2
840 P =
841 23.2800
842 iP =
843 0.0430
844 nM =
845 19.9485    13.4536    13.5052
846 0.6701    20.1031    19.4330
847 2.8351    20.0515    -2.7835
848 mM =
849 25     18
850 1     24
851 iM =
852 0.0412    -0.0309
853 -0.0017    0.0430
854 -----
855 k =
856 3
857 rci =
858     2     2
859 P =
860 20.1031
861 iP =
862 0.0497
863 nM =
864 19.5000    0.5000
865 2.1667   -22.1667
866 mM =
867 25     18     2
868 1     24     8
869 13     6     20
870 iM =
871 0.0369    -0.0297    0.0082
872 0.0072    0.0405   -0.0169
873 -0.0262    0.0072    0.0497
874 -----
875 k =
876 4
877 rci =
878     2     2
879 P =
880 -22.1667
881 iP =
882 -0.0451
883 nM =
884 19.5489
885 mM =
886 25     18     2     9
887 1     24     8     15
888 13     6     20     22
889 19     12     21     3
890 iM =
891 0.0362   -0.0300    0.0052    0.0030
892 0.0040    0.0395   -0.0304    0.0135
893 -0.0366    0.0040    0.0062    0.0436
894 0.0108    0.0032    0.0450   -0.0451
895 -----
896 k =
897 5

```

```

898 rci =
899     1     1
900 P =
901    19.5489
902 iP =
903     0.0512
904 nM =
905     [ ]
906 mM =
907     25     18     2     9     11
908     1     24     8     15     17
909     13     6     20     22     4
910     19     12     21     3     10
911     7     5     14     16     23
912 iM =
913     0.0364    -0.0303     0.0031     0.0031     0.0031
914     0.0015     0.0431    -0.0046     0.0127    -0.0373
915    -0.0370     0.0047     0.0108     0.0435    -0.0065
916     0.0111     0.0028     0.0415    -0.0450     0.0050
917     0.0034    -0.0049    -0.0354     0.0012     0.0512
918 -----
919 detM =
920    5.0700e+006
921 invM =
922    -0.0049     0.0512    -0.0354     0.0012     0.0034
923     0.0431    -0.0373    -0.0046     0.0127     0.0015
924    -0.0303     0.0031     0.0031     0.0031     0.0364
925     0.0047    -0.0065     0.0108     0.0435    -0.0370
926     0.0028     0.0050     0.0415    -0.0450     0.0111
927 p =
928    25.0000    23.2800    20.1031   -22.1667    19.5489
929 s =
930    -1
931 rc =
932     5     1     3     4     2
933     3     2     4     5     1

```

```

937 >> N=3, M=rand(N)+i*randn(N), [dM,iM]=det_inv_p(M),
938 >> W=iM, [dW,iW]=det_inv_p(W), erD=dM-1/dW, erM=M-iW,
939 N =
940     3
941 M =
942    0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
943    0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
944    0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
945 dM =
946    0.8840 + 1.9683i
947 iM =
948    0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
949    0.5856 + 0.5723i   -0.0536 - 0.1048i   -0.2642 - 0.1396i
950   -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
951 -----
952 W =
953    0.8843 + 0.2767i    0.0395 + 0.4206i    0.0463 - 0.7104i
954    0.5856 + 0.5723i   -0.0536 - 0.1048i   -0.2642 - 0.1396i
955   -0.8013 - 0.9100i    0.2988 - 0.4017i    0.5939 + 0.5261i
956 dW =
957    0.1899 - 0.4228i

```

```

958     iw =
959         0.4447 + 0.1746i    0.9218 - 0.5883i    0.4057 + 0.1139i
960         0.6154 - 0.1867i    0.7382 + 2.1832i    0.9355 + 1.0668i
961         0.7919 + 0.7258i    0.1763 - 0.1364i    0.9169 + 0.0593i
962         -----
963     erD =
964         3.3307e-016 -2.2204e-016i
965     erM =
966         1.0e-015 *
967         -0.0555 + 0.0000i    0.0000 - 0.1110i    -0.0555 - 0.1388i
968         -0.1110 - 0.4441i    0.1110 + 0.0000i    -0.2220 + 0.0000i
969         0.0000 - 0.1110i    0.0555 - 0.0833i    -0.1110 + 0.0139i
970
971
972     >> diary off
973
974

```