SHIPLEY'S ALGORITHM FOR INVERTING MATRICES

Finbarr Holland

1. Introduction

This is a brief outline of a method for inverting matrices that was developed in the late fifties at the Tennessee Valley Authority. It was found to be particularly suitable for inverting matrices that describe power system impedances or admittances. The method was first reported on by Shipley and Coleman in [4]. The essentials of the method can also be found in the texts [1] and [3]; but I could find no mention of it in the mathematical literature.

I have been teaching it to a class of Third Year Electrical Engineering students at Cork for the last two years, and some readers may recall a talk I gave at the 1982 Limerick Algebra Conference in which I presented the key ideas behind the method and attempted - unsuccessfully, as it turned out - to demonstrate it on a personal computer.

The method in question is simple to apply, direct rather than iterative, easily programmable and takes full advantage of any symmetry present in the matrix under examination. Since otherwise it closely resembles the well-known Gauss Elimination method, the latter is a significant feature of the Shipley method, which is based on modifying successively the elements of the matrix according to a simple rule.

2. The Shipley Modification of a Square Matrix

Given an nxn matrix $A = [a_{ij}]$, with $a_{kk} \neq 0$, we define the kth Shipley modification $\sigma_k(A) = B$ by the rules:

$$b_{kk} = -1/a_{kk},$$

$$b_{ik} = -a_{ik}/a_{kk} = a_{ik}b_{kk}, \quad i \neq k$$

$$b_{kj} = -a_{kj}/a_{kk} = a_{kj}b_{kk}, \quad j \neq k$$

$$b_{ij} = a_{ij} - a_{ik}a_{kj}/a_{kk} = a_{ij} + a_{ik}b_{kj}, \quad i,j \neq k$$

Example For instance, if

$$M = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 0 & 2 \\ 3 & 2 & 4 \end{bmatrix},$$

then

$$\sigma_1(M) = \begin{bmatrix} -1 & -1 & -3 \\ -1 & -1 & -1 \\ -3 & -1 & -5 \end{bmatrix} = N, \quad \sigma_2(N) = \begin{bmatrix} 0 & -1 & -2 \\ -1 & 1 & -1 \\ -2 & -1 & -4 \end{bmatrix} = L,$$

and

$$\sigma_3(L) = \begin{bmatrix} 1 & -.5 & -.5 \\ -.5 & 1.25 & -.25 \\ -.5 & -.25 & .25 \end{bmatrix}.$$

3. Shipley's Algorithm

It is a simple matter to check that the last displayed matrix is the negative of the inverse of M, i.e.

$$\sigma_3(\sigma_2(\sigma_1(M))) = -M^{-1}$$

This example illustrates the essence of Shipley's algorithm.

THEOREM 1. Let A be an nxn matrix. If for some permutation π of the integers 1,2,.....,n the n-fold composition

$$\sigma_{\pi(n)} \circ \sigma_{\pi(n-1)} \circ \cdots\cdots \sigma_{\pi(2)} \circ \sigma_{\pi(1)}(A)$$

is defined, then it is equal to the negative of the inverse of A.

Perhaps the easiest way to be convinced of this is to consider the correspondence $Ax = y$, between the nx1 vectors x and y, as a system of n equations:

$$\sum_{j=1}^{n} a_{ij}x_j = y_i, \quad i = 1,2,\ldots,n$$

On the assumption that $a_{kk} \neq 0$ we can use the kth equation to express x_k in terms of y_k and x_j, $j = 1,2,\ldots,n$, $j \neq k$, and then substitute this value of x_k into the other equations. After some rearrangement of terms, this leads to an equivalent system, viz.,

$$\sum_{j<k} (a_{ij}-a_{ik}a_{kj}/a_{kk})x_j + (a_{ik}/a_{kk})y_k + \sum_{j>k} (a_{ij}-a_{ik}a_{kj}/a_{kk})x_j = y_i, \qquad i \neq k$$

$$\sum_{j<k} (-a_{kj}/a_{kk})x_j + (1/a_{kk})y_k + \sum_{j>k} (-a_{kj}/a_{kk})x_j = x_k.$$

Allowing for an adjustment of sign, this can be formulated as the matrix equation

$$\sigma_k(A)x' = y',$$

where x' is obtained from x by replacing x_k by $-y_k$ and y' is obtained from y by replacing y_k by x_k.

Under the conditions of the theorem we can carry out the manoevure just described n times, in the order determined by the permutation π, and in this way replace successively each component of the vector x by the negative of the corresponding component of the vector y; we end up with the equation

$$\sigma_{\pi(n)}(\sigma_{\pi(n-1)}(\ldots\ldots(\sigma_{\pi(2)}(\sigma_{\pi(1)}(A)))\ldots\ldots))(-y) = x.$$

This is clearly sufficient to demonstrate the truth of the theorem.

4. Criterion for the Algorithm to Work

Since the method is not completely general - for example it will not invert the simple 2x2 matrix

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- it is important to know when it works. To find out, we analyse the formation of the diagonal elements under different Shipley modifications.

Given $A = [a_{ij}]$, we can form $B = \sigma_1(A)$ if $a_{11} \neq 0$. We can then form $C = \sigma_2(B) = \sigma_2(\sigma_1(A))$ if $b_{22} \neq 0$, i.e. if

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0.$$

If $c_{33} \neq 0$, i.e. if

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \neq 0,$$

we can compute $\sigma_3(C) = \sigma_3(\sigma_2(\sigma_1(A)))$, etc. This leads to the following theorem, whose proof we can omit.

THEOREM 2. If for some permutation π of the integers 1,2,..,n, the principal minors of the matrix $[a_{\pi(i)\pi(j)}]$ are all non-zero, then

$$-A^{-1} = \sigma_{\pi(n)}(\sigma_{\pi(n-1)}(\ldots\ldots(\sigma_{\pi(1)}(A))\ldots\ldots)).$$

Thus, for example, if A is strictly positive-definite or strictly dominant diagonal, that is to say if

$$x^tAx > 0 \text{ for all } x \neq 0,$$

or $$\sum_{j=1}^{n} |a_{ij}| < 2|a_{ii}|, \quad i = 1,2,\ldots,n,$$

then we can apply the Shipley algorithm to evaluate the inverse of A, it being clear in both cases that the property of positive-definiteness or diagonal dominance is inherited by the principal minors of $[a_{\pi(i)\pi(j)}]$ for every permutation π.

5. A Noteworthy Feature of the Shipley Algorithm

An important property of the Shipley algorithm is that it preserves symmetry.

THEOREM 3. If $A = [a_{ij}]$ is symmetric and $a_{kk} \neq 0$, then $\sigma_k(A)$ is symmetric.

PROOF. Obvious.

The fact that symmetry is retained during the implementation of the algorithm reduces the arithmetic of computing the inverse of a symmetric matrix by approximately one half and the memory requirements of a computer by about the same amount.

6. The Complexity of the Shipley Algorithm

To perform one modification of a given matrix we must carry out one division, $(2(n-1) + (n-1)^2)$ multiplications and $(n-1)^2$ additions. Therefore to invert an nxn matrix we must perform about n^3 multiplications and the same number of additions. The order of complexity of the method is therefore $O(n^3)$. In this respect, then, there is no difference between it and the standard elimination procedure.

7. Computer Implementation of the Method

The method is a little unpleasant to operate by hand, but it can be easily programmed for a computer and can therefore be readily demonstrated in a classroom. The program below was designed by my son Ian for the Newbrain personal computer (on which a draft of this article was prepared). It is structured to take advantage of the savings involved in applying the algorithm to symmetric matrices. Also, for a given matrix $A = [a_{ij}]$, the program selects the sequence of operations $\sigma_{\pi(1)}, \sigma_{\pi(2)}, \ldots\ldots, \sigma_{\pi(n)}$, according to the following rule: $\pi(1)$ is chosen to be the index k corresponding to the largest non-vanishing $|a_{kk}|$; if $B = \sigma_{\pi(1)}(A)$, $\pi(2)$ is chosen to be the index $k \neq \pi(1)$ corresponding to the largest non-vanishing $|b_{kk}|$; and so on. By this means it is hoped to keep computational errors to a minimum.

8. Acknowledgements

It is a pleasure to record my thanks to my son Ian for the assistance he has given me with this project, to Tom Laffey who drew my attention to Schwein's theorem and its relatives [2] (which one needs to prove Theorem 2) and to Michael O'Callaghan, with whom I first discussed the Shipley Algorithm, for many useful observations.

References

1. Brown, Homer E., Solution of Large Networks by Matrix Methods, Wiley, 1975.
2. Muir, Thomas, A Treatise on the Theory of Determinants, Dover, 1933.
3. Shipley, R.B., Matrices and Power Systems, Wiley, 1976.
4. Shipley, R.B., and Coleman, D.W., A New Direct Matrix Inversion Method, Trans. AIEE C&E, Nov. 1959, 568-572.

```
  1  REM ** Program to invert a nxn matrix using the Shipley Algorithm **
  9  CLOSE£1:OPEN£1,5:CLEAR
 10  REM Open relevant screens
 20  OPEN£0,4 ,"125"
 21  d$="  "
 30  PUT 31: REM Clear screen
 40  REM Read in dimension of matrix
 50  PUT 22,10,10:?"WHAT IS THE DIMENSION OF THE MATRIX  ";
 60  INPUT n
 62  PUT 22,10,10:?"IS THE MATRIX SYMMETRIC?  Y/N              ";:GET£1,A$
 64  IF INSTR("YyNn",a$)=0 THEN GOTO 62
 70  DIM b(n,n),a(n,n)
 80  REM Read in the elements of the matrix A
 85  PUT 31
 90  PUT 22,20,1:?"Enter the elements of the matrix"
 91  IF INSTR("nN",a$)>0 THEN GOTO 125
 95  FOR x=1 TO n
100     FOR y=x TO n
105        PUT 22,y*9,x*2+2:INPUT("")a(x,y):a(y,x)=a(x,y)
110     NEXT y
115  NEXT x
120  GOTO 150
125  FOR x=1 TO n
130     FOR y=1 TO n
135        PUT 22,y*9,x*2+2:INPUT("") a(x,y)
140     NEXT y
145  NEXT x
150  REM *** THE SHIPLEY ALGORITHM ***
155  FOR t=1 TO n:REM t counts no of modifications
160  REM Find the largest unused diagonal element
170  k1=0
180  FOR x=1 TO n
190     IF INSTR(d$,STR$(x))>0 THEN GOTO 210
200     IF ABS(a(x,x)) > k1 THEN k=x:k1=ABS(a(x,x))
210  NEXT x
220  IF k1=0 THEN END:REM Singular MATRIX if k1=0
250  D$=D$+STR$(K)
300  REM *** THE ALGORITHM PROPER ***
309  REM kth column
310  FOR j=1 TO n
320    IF j=k THEN GOTO 340
330    b(k,j) = -a(k,j) / a(k,k)
340  NEXT J
399  REM kth row
400  FOR i=1 TO n
410     IF i=k THEN GOTO 430
420     b(i,k) = -a(i,k) / a(k,k)
430  NEXT i
500  b(k,k) = -1/a(k,k)
550  REM The modification of the remainder of the matrix
560  FOR i=1 TO n
570     IF i=k GOTO 620
580     FOR j=1 TO n
590        IF j=k GOTO 610
600        b(i,j) = a(i,j) - a(i,k)*a(k,j) / a(k,k)
610     NEXT j
```



```
620  NEXT i
650  REM COPY B INTO A
660  FOR i=1 TO n
670     FOR j=1 TO n
680        a(i,j) = b(i,j)
690     NEXT j
700  NEXT i
750  REM END OF ONE COMPLETE MODIFICATION
760  GOSUB 1910:REM Call routine to print out modified matrix
770  PUT 22,20,23:?"THE MATRIX AFTER ";T;" MODIFICATIONS"
780  PUT 22,20,24:?"press any key to continue";
790  GET£1,ky$
800  NEXT t
900  REM PRINT OUT THE INVERSE
910  PUT 31:PUT 22,20,1:?"The INVERSE is"
920  GOSUB 1910
1000 END
1001 REM THE END OF THE PROGRAM
1900 REM A routine to print out the contents matrix A
1910 REM
1910 FOR x=1 TO n
1920    FOR y=1 TO n
1930       a$+STR$(a(x,y))
1940       IF LEN(a$)=15 THEN a$=LEFT$(a$,4)+RIGHT$(a$,5)
1950       a$=LEFT$(a$+"       ",9)
1960       PUT 22,y*9,x*2+2:?a$
1970    NEXT y
1980 NEXT x
1990 RETURN
```

This program is in Newbrain BASIC.

Department of Mathematics,

University College,

Cork,

Ireland