INVERSA – NxN Matrix Inversion

© 1980 Valentín Albillo

Abstract

INVERSA is a simple program written in 1980 for the SHARP PC-1210, PC-1211 and PC-1212 pocket computers and compatibles to compute the inverse of a given non-singular NxN matrix using an exchange method. Two worked examples are given.

Keywords: *matrix inversion, exchange method, SHARP pocket computers, PC-1210, PC-1211, PC-1212*

1. Introduction

INVERSA is a simple 5-line *(231-byte)* program I wrote in 1980 for the *SHARP PC-1210* / *PC-1211* / *PC-1212* pocket computers and compatible models, which can compute the inverse of a given non-singular *NxN* matrix (up to 13x13 for *PC-1211* / *PC-1212*) using an exchange method. See ***References***.

2. Program Listing

```
10: INPUT A: FOR B=7 TO 6+AA: INPUT A(B): NEXT B: FOR F=1 TO A: E=FA+F+6-A,A(E)=1/A(E)
   : FOR B=1 TO A: FOR C=1 TO A
20: IF B<>F IF C<>F LET D=AB+C+6-A,A(D)=A(D)-A(D-C+F)A(E+C-F)A(E)
30: NEXT C: NEXT B: FOR B=6+F TO B+AA-A STEP A: IF B<>E LET A(B)=A(B)A(E)
40: NEXT B: FOR B=E-F+1 TO E-F+A: IF B<>E LET A(B)=-A(B)A(E)
50: NEXT B: NEXT F: FOR B=7 TO 6+AA: PRINT A(B): NEXT B
```

3. Usage Instructions

See the worked examples to understand how to use the program.

4. Examples

The following examples can be used to check that the program was correctly entered and to understand its usage.

4.1 Example 1

Invert the ***3x3 Hilbert Matrix***:

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

In **RUN** Mode, proceed as follows:

```
RUN  [ENTER]  ?                  1/4  [ENTER]  ?    ...               [ENTER]  192.0000004  ...
3    [ENTER]  ?  (3x3 matrix)    1/3  [ENTER]  ?    ...               [ENTER] -180.0000004  ...
1    [ENTER]  ?  (1st elem.)     1/4  [ENTER]  ?  (8th elem.)         [ENTER]  30.00000009  ...
1/2  [ENTER]  ?  (2nd elem.)     1/5  [ENTER]  9.000000019 1st        [ENTER] -180.0000004  ...
1/3  [ENTER]  ?     ...               [ENTER] -36.00000009 2nd        [ENTER]  180.0000004  9th
1/2  [ENTER]  ?     ...               [ENTER]  30.00000009  ...       [ENTER]
1/3  [ENTER]  ?     ...               [ENTER] -36.00000009  ...    >
```

So, ignoring the small rounding errors, the computed inverse matrix is:

$$A^{-1} = \begin{pmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{pmatrix}$$

4.2 Example 2

Invert the following **4x4** matrix:

$$A = \begin{pmatrix} 2 & 2 & 3 & 2 \\ 11 & 5 & 4 & 6 \\ 2 & 1 & 1 & -9 \\ 2 & 2 & 3 & 1 \end{pmatrix}$$

In **RUN** Mode, proceed as follows:

```
RUN  [ENTER]  ?                   5    [ENTER]  ?     ...                  [ENTER] -251.9999989  ...
4    [ENTER]  ?  (4x4 matrix)     ...  [ENTER]  ?     ...                  [ENTER]  3.999999981  ...
2    [ENTER]  ?  (1st elem.)      3    [ENTER]  ?  (15th elem.)            [ENTER] ...           ...
2    [ENTER]  ?  (2nd elem.)      1    [ENTER]  69.99999972 1st            [ENTER]  0           15th
3    [ENTER]  ?      ...               [ENTER] -9.999999953E-01            [ENTER] -1           16th
2    [ENTER]  ?      ...               [ENTER]  6.999999971  ...           [ENTER]
11   [ENTER]  ?      ...               [ENTER] -70.99999972  ...     >
```

And again ignoring the small rounding errors, the computed inverse matrix is:

$$A^{-1} = \begin{pmatrix} 70 & -1 & 7 & -71 \\ -252 & 4 & -25 & 255 \\ 121 & -2 & 12 & -122 \\ 1 & 0 & 0 & -1 \end{pmatrix}$$

Notes

1. The inversion is done *in-place* so after a succesful inversion, the inverse matrix replaces the input matrix in array A( ).

2. If the matrix to be inverted is *singular or nearly-singular* (its determinant is either **0** or very small) the inversion process will either stop with an ***Error*** condition or give incorrect (usually extremely large) results.

3. Even if the matrix isn't singular, this program is very simple and does not check whether initially there are **0** (or very small) elements in the *main diagonal* or at some other points while computing the inverse. Should this happen, the process will either stop with an ***Error*** condition or give incorrect (usually extremely large) results. In that case re-run the program making sure there are no **0** elements in the main diagonal and/or *rearrange* rows or columns (exchange some), then *revert* the rearrangement in the resulting inverse (e.g.: if you, say, exchange *rows* 2 and 5 in the input matrix, you should afterwards exchange *columns* 2 and 5 in the resulting inverse.)

References

Francis Scheid (1988). *Schaum's Outline of Theory and Problems of Numerical Analysis, 2nd Edition*.

Copyrights

Copyright for this paper and its contents is retained by the author. Permission to use it for non-profit purposes is granted as long as the contents aren't modified in any way and the copyright is acknowledged.