Matrix49 v1.0 by Steen S. Schmidt

May 3, 2001

Contents

Disclaimer & Copyright

This program is freeware, so no registration or licensing fees are necessary. You are free to distribute this program to anyone, as long as this document is included.

I cannot take responsibility for any damage or data loss caused by this program – it is written in 100% SysRPL and there could be bugs in it. This also means that it will <u>not</u> run on the HP48 series.

If you have any suggestions, additions or find any bugs in the code, you're welcome to contact me per email at SSchmidt@nospam.dk. Please include details about ROM revision and flag settings if you're reporting a bug.

Credits

The library is coded and compiled directly on the HP49G, which has proven to be a great tool for this too. A thanks goes to ACO for making it possible – including these programming tools on the calculator itself has made the job a lot easier.

Mr. Bernard Parisse (CAS author on the HP49G) has been a big help explaining his software to me.

As always, a big thanks to the guys at the comp.sys.hp48 newsgroup – especially Carsten Dominik for his Emacs library, in which the RPLCPL command resides.

Requirements & Installation

You need to copy the library (library #1128) to the calculator (with HPComm for example) and store it in a port (any port should do).

To store it in port 2 for example, you will need to recall the library to the stack, type 2 and STO.

Hold down ON and press F3, release both keys and the calc should warm boot (warning: you'll loose all stack contents, but calc memory will remain intact). When this is done, your port 2 should contain the entry for the library (press left shift APPS to enter the Filer):

 →

Running BYTES on the library on the stack should yield #8C42h and 1232 bytes.

 →

This document shows stack syntax for RPN mode, but algebraic mode should work too. Use RPN for best results (as always ☺).

The Matrix Inverse

Most people know that it's fairly hard to calculate the inverse of a matrix – it consists of such extensive operations as finding the adjoint of the matrix and the determinant. Both can be hazardous to calculate, if the matrix has somewhat large dimensions or is ill conditioned.

The inverse of a matrix is defined for square matrices only, and can be expressed as

$$[M]^{-1} = \frac{Adj([M])}{Det([M])}$$

Here it is clearly seen, that if the determinant is zero, there exists no well-defined inverse. The matrix is then said to be singular. Another property of a singular matrix is that its rank is different from its order. The above property does not imply that a singular matrix doesn't have an adjoint – this is always the case. How is the adjoint calculated then? To understand that, we must define two additional entities: the cofactor and the minor.

The minor$_{ij}$ of an element is defined as the determinant of a matrix, when you remove the ith row and jth column. This can in nature only be done on square matrices. The reduced matrix, that you calculate the determinant of, is here called the minor matrix, and is returned by the MMINOR command.

The cofactor$_{ij}$ of an element is defined as $(-1)^{(i+j)} \cdot$ minor$_{ij}$. The cofactor matrix (returned by MCOFACTOR) is merely a matrix with all elements replaced by their cofactors. Transpose this to get the adjoint of the original matrix.

Matrix49

This library adds a couple of matrix commands to the HP49G. All commands accept both symbolic and numeric matrices as input, but will always return symbolic matrices (except for →MNUM) – even

if they're filled with reals. The reason for this is the fact that symbolic matrices (type 29) take up much less RAM than the equivalent numeric matrices (type 3 or 4). Whenever possible, commands will accept vectors as input.

The commands are listed below.

ADJOINT
Adjoint Matrix command:
The adjoint of a square matrix can be used to find its inverse. The adjoint is sometimes called the adjugate.

Level 1	→	Level 1
[matrix]	→	[matrix]$_{\text{Adjoint}}$

The ADJOINT command uses the cofactor matrix to calculate the adjoint, which will usually cause it to be a fair bit slower than doing the inverse times the determinant. The latter approach would of course fail for singular matrices, so I've chosen the certain approach.
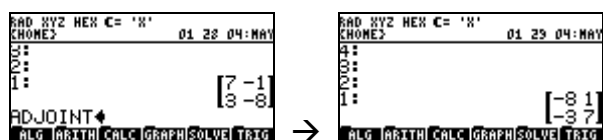
MCOFACTOR
Cofactor Matrix command:
Transpose of the adjoint of a square matrix.

Level 1	→	Level 1
[matrix]	→	[matrix]$_{\text{Cofactors}}$

The cofactor matrix is a matrix with all elements replaced by their cofactors – see the COFACTOR command for further explanation.
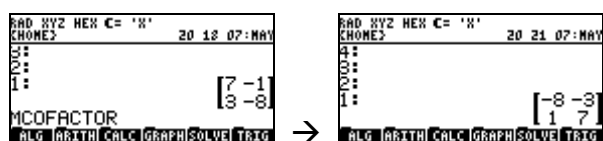
COFACTOR
Cofactor command:
Cofactor of a square matrix.

Level 2	Level 1	→	Level 1
[matrix]	{ i$_{\text{column}}$ j$_{\text{row}}$ }	→	symbolic$_{\text{Cofactor,ij}}$

The symbolic returned can also be a number or an integer. The calculation of a cofactor is explained in the section "The Matrix Inverse".
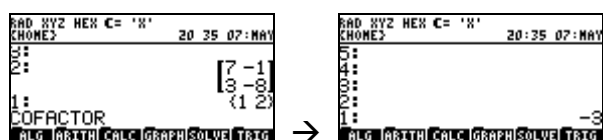
MMINOR
Minor Matrix command:
$Minor_{ij}$ matrix of matrix.

Level 2	Level 1	→	Level 1
[matrix]	{ i_{column} j_{row} }	→	[matrix]$_{Minor,ij}$

A minor$_{ij}$ matrix is a matrix where you've removed row$_i$ and column$_j$. The matrix argument doesn't have to be square, but if you'll calculate the minor$_{ij}$, it'll have to be.
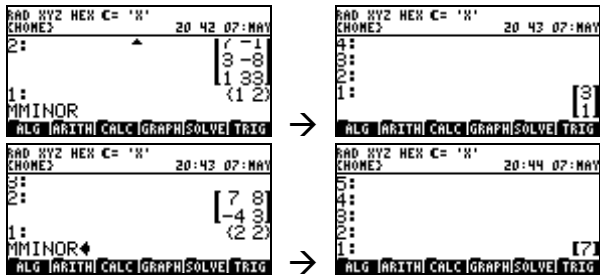
MINOR
Minor command:
$Minor_{ij}$ of a square matrix.

Level 2	Level 1	→	Level 1
[matrix]	{ i_{column} j_{row} }	→	Symbolic$_{Minor,ij}$

The symbolic returned can also be a number or an integer. The calculation of a minor is explained in the section "The Matrix Inverse".
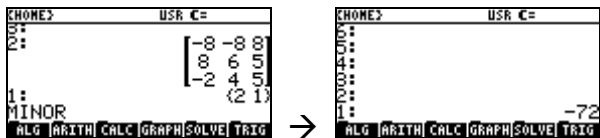
MHERMITE
Hermite Matrix command:
Calculates a Hermite matrix.

Level 1	→	Level 1
[matrix]	→	[matrix]$_{Hermite}$

A Hermite matrix is a conjugate matrix, not necessarily square, transposed. For a real matrix, the Hermite is the same as the transpose.

4
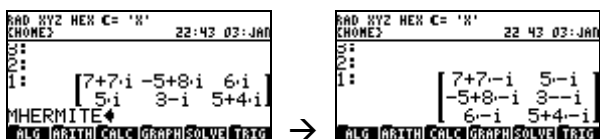
KROENECKER
Kroenecker Multiplication command:
Calculates the Kroenecker product of two square matrices of equal dimensions.

Level 2	Level 1	→	Level 1
[matrix]$_2$	[matrix]$_1$	→	[matrix]$_{Kroenecker}$

The Kroenecker product of the two matrices is the elements of each row of matrix$_2$ multiplied one by one with the whole matrix$_1$. The sub-matrices then make up the Kroenecker product. An example will illustrate this:

Matrix$_1$:   Matrix$_2$:

$$\begin{bmatrix} 1 & 5 \\ 7 & -6 \end{bmatrix} \quad \begin{bmatrix} 11 & 4 \\ -2 & 9 \end{bmatrix}$$

Split matrix$_2$ into the elements of its rows:

$$\{11 \quad 4\}$$
$$\{-2 \quad 9\}$$

The first row of the Kroenecker product then is made of these two horizontally augmented matrices:

$$11 \cdot \begin{bmatrix} 1 & 5 \\ 7 & -6 \end{bmatrix} = \begin{bmatrix} 11 & 55 \\ 77 & -66 \end{bmatrix} \& 4 \cdot \begin{bmatrix} 1 & 5 \\ 7 & -6 \end{bmatrix} = \begin{bmatrix} 4 & 20 \\ 28 & -24 \end{bmatrix} \Rightarrow \begin{bmatrix} 11 & 55 & 4 & 20 \\ 77 & -66 & 28 & -24 \end{bmatrix}$$

The second row of the Kroenecker product then is made of these two horizontally augmented matrices:

$$-2 \cdot \begin{bmatrix} 1 & 5 \\ 7 & -6 \end{bmatrix} = \begin{bmatrix} -2 & -10 \\ -14 & 12 \end{bmatrix} \& 9 \cdot \begin{bmatrix} 1 & 5 \\ 7 & -6 \end{bmatrix} = \begin{bmatrix} 9 & 45 \\ 63 & -54 \end{bmatrix} \Rightarrow \begin{bmatrix} -2 & -10 & 9 & 45 \\ -14 & 12 & 63 & -54 \end{bmatrix}$$

The two rows vertically augmented then combine the complete Kroenecker product:

$$\begin{bmatrix} 11 & 55 & 4 & 20 \\ 77 & -66 & 28 & -24 \\ -2 & -10 & 9 & 45 \\ -14 & 12 & 63 & -54 \end{bmatrix}$$

The example executed on the HP49G:

 →
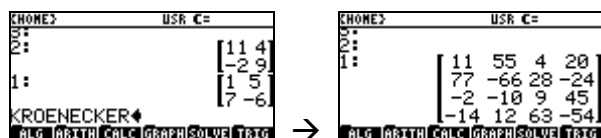
HAUGMENT
Horizontal Augmentation command:
Augments two matrices horizontally.

Level 2	Level 1	→	Level 1
[matrix	[matrix]	→	[matrix]

The built-in AUGMENT command augments two matrices vertically. The two matrices must have the same width.
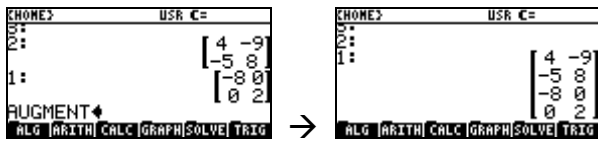
This command, HAUGMENT, augments two matrices horizontally. The two matrices must have the same height.
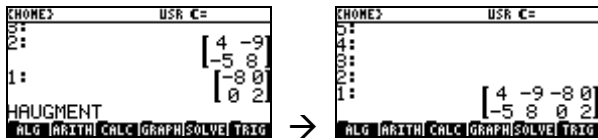
VEC
Vectorize Matrix command:
Converts a matrix into a vector of all its elements.

Level 1	→	Level 1
[matrix]	→	[vector]

The returned vector will always be type 29.
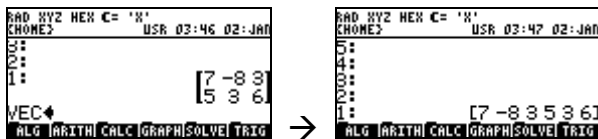
→MNUM
To Numeric Matrix command:
Attempts to convert a matrix to type 3 or 4.

Level 1	→	Level 2	Level 1
[matrix]	→	[matrix]	0. or 1.

The command can only convert a matrix to numeric type if the entire contents can be evaluated into numbers. If the conversion is successful, level 1 will contain 1, else it'll contain 0. If any element evaluates to a complex number, the matrix type will be 4, else it'll be 3 (or 29 if the conversion was unsuccessful).

 (Return type 3)

 → (Return type 4)

 → (Return type 29)

→MSYMB
To Symbolic Matrix command:
Convert a matrix to type 29.

Level 1	→	Level 1
[matrix]	→	[matrix]$_{\text{Type 29}}$

This conversion will always be successful, why →MSYMB will only return the converted matrix.

 → (Always return type 29)

Revision History
v1.0: Initial public release.

7